aws

Panduan Pengembang Database

Amazon Redshift



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Redshift: Panduan Pengembang Database

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

..... xxxiii
Pengantar 1
Prasyarat untuk menggunakan Amazon Redshift 1
Arsitektur Amazon Redshift 2
Arsitektur gudang data 2
Performa 5
Penyimpanan kolumnar 9
Manajemen beban kerja 11
Menggunakan Amazon Redshift dengan layanan lain 12
Database sampel 14
Praktik terbaik 21
Melakukan bukti konsep 22
Langkah 1: Lingkup POC Anda 22
Langkah 2: Luncurkan Amazon Redshift 24
Langkah 3: Muat data Anda 25
Langkah 4: Analisis data Anda 26
Langkah 5: Optimalkan 28
Praktik terbaik untuk merancang tabel 29
Pilih kunci pengurutan terbaik 30
Pilih gaya distribusi terbaik 30
Gunakan kompresi otomatis 32
Tentukan kendala 32
Gunakan ukuran kolom sekecil mungkin 33
Gunakan date/time tipe data untuk kolom tanggal 33
Praktik terbaik untuk memuat data 33
Pelajari cara memuat data 34
Gunakan perintah COPY untuk memuat data 34
Gunakan satu perintah COPY 34
Memuat file data 34
Mengompresi file data Anda 35
Verifikasi file data sebelum dan sesudah pemuatan 36
Gunakan sisipan multi-baris 36
Gunakan sisipan massal 36
Muat data dalam urutan kunci 37

Muat data dalam blok berurutan ... 37
Gunakan tabel deret waktu ... 38
Jadwalkan sekitar jendela pemeliharaan ... 38
Praktik terbaik untuk merancang kueri ... 39
Ikuti rekomendasi Penasihat ... 41
Bagaimana Advisor Bekerja ... 41
Mengoptimalkan Arsitektur Berbagi Data ... 42
Wilayah Amazon Redshift tempat Penasihat didukung ... 42
Melihat rekomendasi Penasihat ... 44
Rekomendasi penasihat ... 45
Tutorial ... 60
Optimasi database otomatis ... 62
Mengalokasikan sumber daya komputasi tambahan ... 64
Mengelola sumber daya komputasi tambahan untuk otonom ... 64
Kontrol biaya untuk otonom menggunakan sumber daya komputasi tambahan ... 68
Penagihan untuk operasi otonom ... 69
Penagihan otonom untuk cluster yang disediakan ... 69
Penagihan otonom untuk kelompok kerja tanpa server ... 70
Metrik penggunaan untuk operasi otonom ... 71
Pengoptimalan tabel otomatis ... 72
Mengaktifkan, menonaktifkan, dan memantau pengoptimalan tabel otomatis ... 73
Mengaktifkan pengoptimalan tabel otomatis ... 73
Menghapus pengoptimalan tabel otomatis dari tabel ... 73
Pemantauan pengoptimalan tabel otomatis ... 74
Mengelola pengecualian beban kerja dari Autonomics ... 74
Mengelola sumber daya denylist di konsol Amazon Redshift ... 75
Menambahkan sumber daya denylist ... 76
Hapus sumber daya denylist ... 76
Kompresi kolom ... 76
Pengkodean kompresi ... 78
Menguji pengkodean kompresi ... 88
Distribusi data ... 95
Konsep distribusi data ... 96
Gaya distribusi ... 97
Melihat gaya distribusi ... 99
Mengevaluasi pola kueri ... 100

Menunjuk gaya distribusi 101
Mengevaluasi rencana kueri 102
Contoh rencana kueri 105
Contoh distribusi 109
Urutkan kunci 112
Penyortiran tata letak data multidimensi 113
Kunci pengurutan majemuk 114
Kunci pengurutan yang disisipkan 114
Kendala tabel 116
Memuat data 118
Memuat tabel dengan COPY 121
Kredensi dan izin akses 123
Mempersiapkan data input Anda 125
Memuat data dari Amazon S3 126
Memuat data dari Amazon EMR 139
Memuat data dari host jarak jauh 145
Memuat dari Amazon DynamoDB 154
Memverifikasi bahwa data dimuat dengan benar 157
Memvalidasi data masukan 158
Memuat tabel dengan kompresi otomatis 158
Mengoptimalkan untuk tabel sempit 161
Memuat nilai kolom default 162
Memecahkan masalah beban data 163
Buat integrasi acara S3 untuk menyalin file secara otomatis 171
Prasyarat 171
Membuat integrasi acara S3 176
Membuat dan memantau COPY JOB 178
Pertimbangan-pertimbangan 180
Wilayah yang Didukung 180
Pertimbangan saat memulihkan target integrasi peristiwa S3 183
Memuat tabel dengan DML 184
Memperbarui dan memasukkan 184
Melakukan salinan mendalam 192
Menganalisis tabel 197
Analisis otomatis 197
Analisis data tabel baru 197

Analisis riwayat perintah 203
Meja penyedot debu 204
Pengurutan tabel otomatis 205
Hapus vakum otomatis 206
Frekuensi vakum 206
Urutkan tahap dan gabungkan tahap 207
Ambang vakum 208
Jenis vakum 208
Meminimalkan waktu vakum 208
Mengelola operasi penulisan bersamaan 217
Tingkat isolasi 219
Menulis dan men read/write goperasikan 224
Contoh penulisan bersamaan 226
Pemecahan masalah 229
Tutorial: Memuat data dari Amazon S3 232
Prasyarat 232
Gambaran umum 233
Langkah 1: Buat cluster 233
Langkah 2: Unduh file data 234
Langkah 3: Unggah file ke bucket Amazon S3 235
Langkah 4: Buat tabel sampel 237
Langkah 5: Jalankan perintah COPY 240
Langkah 6: Vakum dan analisis database 258
Langkah 7: Bersihkan sumber daya Anda 258
Ringkasan 259
Membongkar data 260
Membongkar data ke Amazon S3 260
Membongkar file data terenkripsi 264
Membongkar data dalam format lebar terbatas atau tetap 266
Memuat ulang data yang dibongkar 267
User-defined fungsi 269
Keamanan dan izin UDF 269
Mencegah konflik penamaan UDF 270
Membebani nama fungsi 270
Menghindari konflik dengan fungsi Amazon Redshift bawaan 271
UDF SQL skalar 271

Contoh 272
UDF Python skalar 273
Contoh 274
Jenis data Python UDF 274
Dukungan bahasa Python 276
Batasan 281
Kesalahan pencatatan dan peringatan 281
UDF Lambda Skalar 283
Mengelola keamanan dan izin Lambda UDF 284
Mengkonfigurasi parameter otorisasi untuk Lambda UDF 285
Menggunakan antarmuka JSON antara Amazon Redshift dan Lambda 287
Gunakan contoh kasus untuk UDF 290
Membuat prosedur tersimpan 292
Ikhtisar prosedur tersimpan 292
Penamaan prosedur yang disimpan 296
Keamanan dan hak istimewa 297
Mengembalikan set hasil 298
Mengelola transaksi 301
Kesalahan menjebak 314
Pencatatan prosedur tersimpan 322
Batasan 323
PL/pgSQL referensi bahasa 324
PL/pgSQL konvensi referensi 324
Struktur PL/pgSQL 325
PL/pgSQL Pernyataan yang didukung 331
Tampilan terwujud 348
Kueri tampilan yang terwujud 351
Penulisan ulang kueri otomatis untuk menggunakan tampilan yang terwujud 352
Catatan penggunaan 352
Cara kerja penulisan ulang kueri otomatis dari tampilan yang terwujud 353
Batasan 354
Tampilan terwujud pada tabel danau data eksternal 355
Batasan 355
Menyegarkan tampilan yang terwujud 356
Tampilan terwujud bersarang 357
Batasan 359

Menyegarkan tampilan yang terwujud secara otomatis 360
Tampilan terwujud otomatis 361
Lingkup SQL dan pertimbangan untuk tampilan terwujud otomatis 364
Batasan tampilan terwujud otomatis 364
Penagihan untuk tampilan terwujud otomatis 365
Sumber daya tambahan 365
Menggunakan fungsi yang ditentukan pengguna (UDF) dalam tampilan terwujud 365
Mereferensikan UDF dalam tampilan terwujud 365
Streaming konsumsi ke tampilan yang terwujud 367
Bagaimana data mengalir dari layanan streaming ke Redshift 368
Praktik terbaik penguraian data untuk meningkatkan kinerja 368
Perilaku penyerapan streaming dan tipe data 369
Memulai penyerapan streaming dari Amazon Kinesis Data Streams 376
Memulai dengan konsumsi streaming dari Apache Kafka 381
Tutorial penyerapan streaming data stasiun kendaraan listrik, menggunakan Kinesis 396
Tampilan Katalog Data 400
Prasyarat 401
End-to-end contoh 402
Pertimbangan-pertimbangan 403
Mengkueri data spasial 404
Tutorial: Menggunakan fungsi SQL spasial 407
Prasyarat 408
Langkah 1: Buat tabel dan muat data uji 408
Langkah 2: Kueri data spasial 411
Langkah 3: Bersihkan sumber daya Anda 415
Memuat shapefile 415
Terminologi 417
Kotak pembatas 417
Validitas geometris 418
Kesederhanaan geometris 420
H3 422
Pertimbangan-pertimbangan 422
Mengkueri data dengan kueri federasi 424
Memulai dengan menggunakan kueri federasi ke PostgreSQL 425
Memulai menggunakan kueri federasi ke PostgreSQL dengan CloudFormation 426
Meluncurkan CloudFormation tumpukan untuk kueri federasi Redshift 427

Mengkueri data dari skema eksternal ... 428
Memulai dengan menggunakan kueri federasi ke MySQL ... 429
Membuat rahasia dan peran IAM ... 431
Prasyarat ... 431
Contoh menggunakan kueri federasi ... 433
Contoh menggunakan kueri federasi dengan PostgreSQL ... 433
Contoh penggunaan nama kasus campuran ... 436
Contoh menggunakan query federasi dengan MySQL ... 437
Perbedaan jenis data ... 438
Pertimbangan-pertimbangan ... 443
Versi database federasi yang didukung ... 445
Izin federasi ... 446
Konsep Utama ... 446
Manfaat ... 447
Kasus penggunaan ... 447
Pertimbangan-pertimbangan ... 448
Prasyarat ... 451
Persyaratan kebijakan IAM ... 451
Konfigurasi aplikasi ... 452
Prasyarat Formasi Danau ... 458
Konfigurasi jaringan ... 460
Hubungkan hak istimewa ... 460
Orientasi ... 461
Pendaftaran cluster pergeseran merah ... 461
Redshift Pendaftaran namespace tanpa server ... 464
Aktifkan AWS IAM Identity Center propagasi identitas ... 466
MENGUBAH IDENTITAS GLOBAL SET PENGGUNA ... 468
Mengkueri katalog ... 470
Prasyarat ... 470
Mengkueri Database Federasi ... 471
Keamanan dan Kontrol Akses ... 471
Kontrol akses ... 472
Berikan/Cabut ... 472
Fine-grained kontrol akses ... 473
Contoh ... 476
Prasyarat ... 477

Pada instance katalog berikan sys:secadmin Peran ke Peran IAM yang sesuai 477
Menyiapkan Lingkungan 478
Menyiapkan Izin Tingkat Kolom (CLP) 479
Membuat Kebijakan Keamanan Tingkat Baris (RLS) 479
Membuat Kebijakan Dynamic Data Masking (DDM) 480
Available/Applied Kebijakan Audit 481
Menguji Pola Akses 481
Mengubah Kebijakan 482
Kebijakan Pelepasan dan Penghapusan 482
Bersihkan: Matikan kebijakan RLS dan hapus 483
Setel ulang identitas global 483
Konfigurasi pengguna federasi 484
Konfigurasi Pengguna yang konsisten di seluruh Amazon Redshift Warehouse 484
Manajemen Koneksi dan Konfigurasi 485
Menonaktifkan propagasi Pusat Identitas IAM 485
Cluster yang disediakan 486
Namespace 486
Deregister dari AWS Glue Data Catalog 487
Spektrum Pergeseran Merah Amazon 490
Ikhtisar Amazon Redshift Spectrum 491
Wilayah Spektrum Pergeseran Merah Amazon 492
Batasan Amazon Redshift Spectrum 492
Memulai dengan Amazon Redshift Spectrum 493
Prasyarat 494
CloudFormation 495
Memulai dengan Redshift Spectrum langkah demi langkah 495
Langkah 1. Membuat peran IAM 495
Langkah 2: Mengaitkan peran IAM dengan cluster Anda 500
Langkah 3: Buat skema eksternal dan tabel eksternal 500
Langkah 4: Kueri data Anda di Amazon S3 502
Luncur CloudFormation kan tumpukan Anda dan kemudian kueri data Anda 505
Tabel Kueri S3 508
Prasyarat 509
Kueri Tabel S3 dari Amazon Redshift 509
Kebijakan IAM untuk Amazon Redshift Spectrum 517
Izin Amazon S3 517

Cross-account Izin Amazon S3 519
Berikan atau batasi akses menggunakan Redshift Spectrum 519
Izin minimum 520
Merantau peran IAM 522
Mengakses AWS Glue data 523
Spektrum Pergeseran Merah dan Formasi Danau 531
Menggunakan filter data untuk keamanan tingkat baris dan tingkat sel 533
File data untuk kueri di Amazon Redshift Spectrum 534
Format data untuk Redshift Spectrum 534
Jenis kompresi untuk Redshift Spectrum 536
Enkripsi untuk Redshift Spectrum 537
Skema eksternal 537
Bekerja dengan katalog eksternal 539
Tabel eksternal 544
Pseudokolom 546
Partisi tabel eksternal Redshift Spectrum 547
Pemetaan ke kolom ORC 553
Membuat tabel eksternal untuk Hudi-managed data 556
Membuat tabel eksternal untuk data Danau Delta 557
Menggunakan tabel Apache Iceberg 560
Pertimbangan saat menggunakan tabel Apache Iceberg 561
Jenis data yang didukung 563
Mereferensikan tabel Iceberg 567
Menulis ke tabel Apache Iceberg 571
Fitur Iceberg v3 594
Performa kueri Amazon Redshift Spectrum 599
Opsi penanganan data 603
Melakukan subquery yang berkorelasi 604
Metrik-metrik 605
Catatan tentang cluster yang disediakan RG 605
Pemecahan masalah kueri 607
Percobaan ulang terlampaui 607
Akses dibatasi 608
Batas sumber daya terlampaui 609
Tidak ada baris yang dikembalikan untuk tabel yang dipartisi 609
Tidak ada kesalahan resmi 610

Format data yang tidak kompatibel 610
Kesalahan sintaks saat menggunakan Hive DDL di Amazon Redshift 611
Izin untuk membuat tabel sementara 611
Rentang tidak valid 611
Nomor versi Parket tidak valid 611
Bidang wajib hilang dari informasi partisi eksternal 612
Tutorial: Menanyakan data bersarang dengan Amazon Redshift Spectrum 613
Ikhtisar 613
Langkah 1: Buat tabel eksternal yang berisi data bersarang 614
Langkah 2: Kueri data bersarang Anda di Amazon S3 dengan ekstensi SQL 615
Kasus penggunaan data bersarang 619
Batasan data bersarang (pratinjau) 622
Serialisasi JSON bersarang kompleks 624
HyperLogLog sketsa 628
Pertimbangan-pertimbangan 629
Batasan 630
Contoh 631
Contoh: Mengembalikan kardinalitas dalam subquery 631
Contoh: Mengembalikan tipe HLLSKETCH dari sketsa gabungan dalam subquery 632
Contoh: Mengembalikan HyperLogLog sketsa dari menggabungkan beberapa sketsa 632
Contoh: Menghasilkan HyperLogLog sketsa di atas data S3 menggunakan tabel eksternal . 633
Cross-database pertanyaan 637
Pertimbangan-pertimbangan 639
Batasan 640
Contoh 640
Menggunakan kueri lintas database dengan editor kueri 646
Kompatibilitas Apache Iceberg 648
Wilayah di mana kompatibilitas Apache Iceberg tersedia 648
Pertimbangan dan batasan saat menggunakan katalog Amazon Redshift di AWS Glue Data Catalog 649
Persyaratan kebijakan IAM 650
Mendaftarkan cluster dan ruang nama 651
Menghapus pendaftaran cluster dan ruang nama 652
Kelompok kerja yang dikelola 654
Mengkueri katalog yang terdaftar di AWS Glue Data Catalog 654
Menanyakan katalog federasi Glue IRC 655

Gambaran umum arsitektur ... 656
Same-account penyiapan ... 656
Cross-account penyiapan ... 658
Praktik dan batasan terbaik ... 661
Berbagi data ... 662
Kasus penggunaan berbagi data ... 662
Multi-warehouse menulis kasus penggunaan ... 663
Berbagi data di berbagai tingkatan ... 663
Manajemen konsistensi berbagi data ... 664
Pertimbangan-pertimbangan ... 664
Pertimbangan umum untuk berbagi data ... 665
Pertimbangan untuk berbagi data membaca dan menulis ... 666
Pertimbangan untuk berbagi data dalam pemulihan tanpa server ... 668
Pertimbangan untuk berbagi data dengan tabel danau data ... 669
Pertimbangan untuk berbagi data dengan AWS Lake Formation ... 669
Pertimbangan untuk berbagi data dengan AWS Data Exchange ... 670
Izin yang dapat Anda berikan untuk berbagi data ... 671
Pernyataan SQL yang didukung untuk berbagi data menulis pada konsumen ... 672
Pernyataan SQL yang tidak didukung untuk berbagi data menulis pada konsumen ... 673
Tersedia Wilayah AWS ... 674
Memulai ... 678
Memulai berbagi data baca-saja di konsol ... 678
Memulai dengan berbagi data read-only dengan antarmuka SQL ... 695
Memulai penulisan multi-gudang ... 741
Memulai dengan CloudFormation ... 773
Jenis datashares ... 779
Berbagi data standar ... 780
AWS Data Exchange berbagi data ... 781
AWS Lake Formation-berbagi data yang dikelola ... 784
Status Data Share ... 786
Mengelola akses ke operasi API berbagi data dengan kebijakan IAM ... 787
Menghubungkan ke database konsumen ... 789
Mengakses data bersama ... 789
Mengakses metadata untuk objek bersama ... 789
Mengintegrasikan berbagi data Amazon Redshift dengan alat intelijen bisnis ... 790
Pemantauan dan audit berbagi data ... 791

Mengintegrasikan berbagi data Amazon Redshift dengan AWS CloudTrail 792
Semi-structured data dalam Amazon Redshift 793
Praktik terbaik untuk bekerja dengan data semi-terstruktur 793
Konsep untuk penggunaan tipe data SUPER 794
Pertimbangan untuk menggunakan data tipe SUPER 795
PartiQL untuk Amazon Redshift 796
Apa yang dimaksud dengan PartiQL? 796
PartiQL di Amazon Redshift 797
Memuat data semi-terstruktur 797
Menggunakan JSON_PARSE untuk menyisipkan data JSON 797
Menggunakan COPY untuk memuat data JSON 798
Membongkar data semi-terstruktur 804
Mengkueri data semi-terstruktur 805
Navigasi 806
Kueri tanpa sarang 806
Objek tidak berputar 808
Pengetikan dinamis 809
Case-insensitive menanyakan 813
Semantik yang lemah 814
Urutkan berdasarkan 814
Operator dan fungsi 814
Operator aritmatika 815
Fungsi aritmatika 815
Fungsi array 816
Perilaku pengumpulan 818
Fungsi informasi 819
Fungsi objek 819
Fungsi string 820
Konfigurasi SUPER 822
Mode longgar dan ketat untuk SUPER 822
Mengakses bidang JSON dengan huruf besar dan huruf campuran 822
Opsi penguraian 823
Batasan 824
Jenis data SUPER dan tampilan yang terwujud 827
Mempercepat kueri PartiQL 828

Menghancurkan data semi-terstruktur menjadi kolom SUPER dengan tampilan yang terwujud ... 828
Membuat kolom skalar Amazon Redshift dari data yang diparut ... 830
Contoh ... 832
Memuat data semi-terstruktur ... 832
Mengkueri data semi-terstruktur bersarang ... 839
Menggunakan enable_case_sensitive_super_atribut dengan data semi-terstruktur ... 840
Memfilter data semi-terstruktur ... 841
Data semi-terstruktur yang tidak bersarang ... 841
Array bersarang yang tidak bersarang ... 842
Menggunakan data semi-terstruktur dalam subquery ... 843
Menggabungkan kueri menggunakan data semi-terstruktur ... 843
Menggunakan data semi-terstruktur dalam tampilan terwujud ... 844
Menggunakan PIVOT dan UNPIVOT dengan data semi-terstruktur ... 844
Machine learning ... 847
Ikhtisar pembelajaran mesin ... 849
Bagaimana pembelajaran mesin dapat memecahkan masalah ... 849
Syarat dan konsep untuk Amazon Redshift ML ... 851
Pembelajaran mesin untuk pemula dan ahli ... 853
Biaya untuk menggunakan Amazon Redshift ML ... 855
Biaya untuk menggunakan Amazon Redshift ML dengan AI SageMaker ... 856
Biaya untuk menggunakan Amazon Redshift ML dengan Amazon Bedrock ... 856
Memulai: Amazon Redshift ML ... 858
Pengaturan administratif ... 858
Menggunakan penjelasan model dengan Amazon Redshift ML ... 864
Metrik probabilitas Amazon Redshift ML ... 864
Tutorial untuk Amazon Redshift ML ... 867
Tutorial: Membangun model churn pelanggan ... 868
Tutorial: Membangun model K-means pengelompokan ... 877
Tutorial: Membangun model klasifikasi multi-kelas ... 887
Tutorial: Membangun model XGBoost ... 897
Tutorial: Membangun model regresi ... 903
Tutorial: Membangun model regresi dengan pembelajar linier ... 916
Tutorial: Membangun model klasifikasi multi-kelas dengan pembelajar linier ... 924
Integrasi Amazon Redshift ML dengan Amazon Bedrock ... 947

Membuat peran IAM 947
Membuat model eksternal 948
Menggunakan model eksternal 950
Rekayasa yang cepat 952
Penyetelan kinerja kueri 954
Pemrosesan kueri 954
Perencanaan kueri dan alur kerja pelaksanaan 955
Membuat dan menafsirkan rencana kueri 957
Meninjau langkah-langkah rencana kueri 965
Faktor-faktor yang mempengaruhi kinerja kueri 968
Analisis dan peningkatan kueri 969
Alur kerja analisis kueri 970
Meninjau peringatan kueri 971
Menganalisis rencana kueri 973
Menganalisis ringkasan kueri 974
Peningkatan kueri 982
Kueri diagnostik untuk penyetelan kueri 986
Pemecahan masalah kueri 990
Koneksi gagal 991
Kueri hang 992
Kueri memakan waktu terlalu lama 993
Beban gagal 995
Beban memakan waktu terlalu lama 995
Memuat data salah 996
Mengatur parameter ukuran pengambilan JDBC 996
Manajemen beban kerja 998
Mengalihkan mode WLM 1000
Memodifikasi konfigurasi WLM 1000
Migrasi dari WLM manual ke WLM otomatis 1000
WLM otomatis 1002
Prioritas 1003
Mode penskalaan konkurensi 1003
Grup pengguna 1004
Peran pengguna 1004
Grup kueri 1004
Wildcard 1004

Aturan pemantauan kueri 1005
Memeriksa WLM otomatis 1005
Prioritas kueri 1006
WLM manual 1011
Mode penskalaan konkurensi 1012
Tingkat konkurensi 1013
Grup pengguna 1015
Peran pengguna 1015
Grup kueri 1015
Wildcard 1016
Persentase memori WLM untuk digunakan 1016
Batas waktu WLM 1016
Aturan pemantauan kueri 1017
Melompat antrian kueri WLM 1017
Tutorial: Mengkonfigurasi antrian WLM manual 1021
Penskalaan konkurensi 1040
Kemampuan penskalaan konkurensi 1040
Batasan untuk penskalaan konkurensi 1041
Wilayah untuk penskalaan konkurensi 1042
Kandidat penskalaan konkurensi 1043
Mengkonfigurasi antrian penskalaan konkurensi 1007
Memantau penskalaan konkurensi 1045
Tampilan sistem penskalaan konkurensi 1046
Akselerasi kueri singkat 1046
Waktu proses SQA maksimum 1047
Pemantauan SQA 1048
Aturan penetapan antrian WLM 1049
Contoh tugas antrian 1051
Menetapkan kueri ke antrian 1053
Menetapkan kueri ke antrian berdasarkan peran pengguna 1053
Menetapkan kueri ke antrian berdasarkan grup pengguna 1054
Menetapkan kueri ke grup kueri 1054
Menetapkan kueri ke antrian superuser 1055
Sifat dinamis dan statis 1055
Alokasi memori dinamis WLM 1057
Contoh WLM dinamis 1058

Aturan pemantauan kueri 1060
Mendefinisikan aturan monitor kueri 1061
Metrik pemantauan kueri untuk Amazon Redshift disediakan 1064
Metrik pemantauan kueri untuk Amazon Redshift Serverless 1067
Templat aturan pemantauan kueri 1069
Tabel sistem dan tampilan untuk aturan pemantauan kueri 1071
Tabel dan tampilan sistem WLM 1071
ID kelas layanan WLM 1073
Keamanan database 1075
Ikhtisar keamanan Amazon Redshift 1076
Izin pengguna basis data default 1077
Superuser 1078
Pengguna 1079
Membuat, mengubah, dan menghapus pengguna 1079
Penguncian pengguna 1080
Grup 1082
Membuat, mengubah, dan menghapus grup 1082
Contoh untuk mengontrol akses pengguna dan grup 1082
Skema 1084
Jalur pencarian 1085
Membuat, mengubah, dan menghapus skema 1085
Perizinan 1086
Role-based kontrol akses 1086
Hirarki peran 1087
Penetapan peran 1088
Peran yang ditentukan sistem Amazon Redshift 1088
Izin sistem 1093
Izin objek database 1104
UBAH HAK ISTIMEWA STANDAR untuk RBAC 1104
Pertimbangan untuk penggunaan peran 1105
Mengelola peran 1106
Tutorial: Membuat peran dan menanyakan dengan RBAC 1106
Row-level keamanan 1125
Menggunakan kebijakan RLS dalam pernyataan SQL 1126
Menggabungkan beberapa kebijakan per pengguna 1127
Kepemilikan dan manajemen kebijakan RLS 1129

Policy-dependent objek dan prinsip 1130
Pertimbangan dan batasan 1132
Praktik terbaik 1136
End-to-end contoh 1140
Keamanan metadata 1144
Penyembunyian data dinamis 1146
Perintah SQL untuk kebijakan DDM 1147
Hirarki kebijakan DDM 1148
Menggunakan DDM dengan jalur tipe SUPER 1150
Penyembunyian data dinamis bersyarat 1155
Tampilan sistem DDM 1156
Pertimbangan-pertimbangan 1158
End-to-end contoh 1163
Izin terlingkup 1166
Pertimbangan-pertimbangan 1167
Referensi SQL 1168
Amazon Redshift SQL 1168
Fungsi SQL didukung pada node pemimpin 1169
Amazon Redshift dan PostgreSQL 1171
Menggunakan SQL 1179
Konvensi referensi SQL 1180
Elemen dasar 1181
Ekspresi 1238
Ketentuan 1243
Perintah SQL 1271
MEMBATALKAN 1276
ALTER DATABASE 1277
ALTER DATA 1284
UBAH HAK ISTIMEWA DEFAULT 1288
MENGUBAH SKEMA EKSTERNAL 1292
UBAH TAMPILAN EKSTERNAL 1294
ALTER FUNCTION 1295
MENGUBAH KELOMPOK 1297
UBAH PENYEDIA IDENTITAS 1298
MENGUBAH KEBIJAKAN MASKING 1301
MENGUBAH TAMPILAN YANG TERWUJUD 1302

MENGUBAH KEBIJAKAN RLS 1309
MENGUBAH PERAN 1311
MENGUBAH PROSEDUR 1312
ALTER SCHEMA 1314
MENGUBAH SISTEM 1315
ALTER TABLE 1317
UBAH TABEL DITAMBAHKAN 1344
UBAH TEMPLAT 1350
ALTER USER 1353
MENGANALISA 1360
MENGANALISIS KOMPRESI 1363
LAMPIRKAN KEBIJAKAN MASKING 1366
LAMPIRKAN KEBIJAKAN RLS 1368
MULAI 1370
PANGGILAN 1372
CANCEL (BATALKAN) 1376
TUTUP 1378
KOMENTAR 1379
COMMIT 1382
MENYALIN 1382
BUAT BASIS DATA 1491
BUAT DATASHARE 1506
BUAT FUNGSI EKSTERNAL 1508
BUAT MODEL EKSTERNAL 1519
BUAT SKEMA EKSTERNAL 1524
CREATE EXTERNAL TABLE 1535
BUAT TAMPILAN EKSTERNAL 1565
CREATE FUNCTION 1566
BUAT GRUP 1573
BUAT PENYEDIA IDENTITAS 1573
BUAT PERPUSTAKAAN 1577
BUAT KEBIJAKAN MASKING 1580
BUAT TAMPILAN YANG TERWUJUD 1581
BUAT MODEL 1588
BUAT PROSEDUR 1618
BUAT KEBIJAKAN RLS 1624

CREATE ROLE 1626
BUAT SKEMA 1627
CREATE TABLE 1630
CREATE TABLE AS 1655
BUAT TEMPLAT 1667
BUAT PENGGUNA 1671
CREATE VIEW 1678
DEALOKASI 1684
MENYATAKAN 1685
DELETE 1690
DESKRIPSI DATA 1694
PENYEDIA IDENTITAS DESC 1695
LEPASKAN KEBIJAKAN MASKING 1696
LEPASKAN KEBIJAKAN RLS 1698
DROP DATABASE 1699
DROP DATA 1700
JATUHKAN TAMPILAN EKSTERNAL 1702
FUNGSI DROP 1703
KELOMPOK DROP 1705
JATUHKAN PENYEDIA IDENTITAS 1706
JATUHKAN PERPUSTAKAAN 1707
KEBIJAKAN DROP MASKING 1707
MODEL JATUHKAN 1708
JATUHKAN TAMPILAN TERWUJUD 1709
PROSEDUR DROP 1710
KEBIJAKAN DROP RLS 1712
DROP ROLE 1713
DROP SCHEMA 1714
MEJA DROP 1717
MENJATUHKAN TEMPLAT 1721
JATUHKAN PENGGUNA 1722
MENJATUHKAN TAMPILAN 1724
AKHIR 1726
EXECUTE 1727
EXPLAIN 1728
AMBIL 1737

HIBAH 1739
INSERT 1768
INSERT (tabel eksternal) 1775
KUNCI 1778
MERGE 1779
MEMPERSIAPKAN 1786
MENYEGARKAN TAMPILAN YANG TERWUJUD 1788
MENGATUR ULANG 1793
MENCABUT 1794
ROLLBACK 1814
SELECT 1816
PILIH KE 1896
SET 1897
MENGATUR OTORISASI SESI 1903
MENGATUR KARAKTERISTIK SESI 1904
MENUNJUKKAN 1904
TAMPILKAN PENGUNCIAN PENGGUNA 1905
TAMPILKAN HIBAH KOLOM 1907
TAMPILKAN KOLOM 1909
TUNJUKKAN KENDALA 1912
TAMPILKAN TABEL EKSTERNAL 1914
TAMPILKAN DATABASE 1918
TAMPILKAN FUNGSI 1922
TUNJUKKAN HIBAH 1924
TUNJUKKAN MODEL 1931
TAMPILKAN DATA SHARE 1934
TAMPILKAN PARAMETER 1935
TAMPILKAN KEBIJAKAN 1938
TAMPILKAN PROSEDUR 1942
TUNJUKKAN PROSEDUR 1943
TAMPILKAN SKEMA 1945
TAMPILKAN TABEL 1947
TAMPILKAN TABEL 1950
TAMPILKAN TEMPLATE 1952
TAMPILKAN TEMPLAT 1954
TAMPILKAN TAMPILAN 1956

START TRANSACTION 1957
MEMOTONG 1957
MEMBONGKAR 1959
UPDATE 1993
USE 2002
VAKUM 2012
Referensi fungsi SQL 2020
Node—hanya fungsi utama 2021
Fungsi agregat 2023
Fungsi array 2052
Bit-wise fungsi agregat 2076
Ekspresi bersyarat 2084
Fungsi pemformatan tipe data 2099
Fungsi tanggal dan waktu 2138
Fungsi hash 2211
HyperLogLog fungsi 2221
Fungsi JSON 2227
Fungsi pembelajaran mesin 2249
Fungsi matematika 2252
Fungsi objek 2291
Fungsi spasial 2302
Fungsi string 2453
Fungsi informasi tipe SUPER 2533
Fungsi VARBYTE 2548
Fungsi jendela 2558
Fungsi administrasi sistem 2625
Fungsi informasi sistem 2636
Kata yang dicadangkan 2670
Tabel sistem dan referensi tampilan 2675
Jenis tabel dan tampilan sistem 2676
Visibilitas data dalam tabel dan tampilan sistem 2677
Memfilter kueri yang dihasilkan sistem 2678
Memigrasi kueri khusus yang disediakan ke kueri tampilan pemantauan SYS 2678
Bermigrasi dari cluster yang disediakan ke Amazon Redshift Tanpa Server 2678
Memperbarui kueri sambil tetap berada di klaster yang disediakan 2679
Meningkatkan pelacakan pengenal kueri menggunakan tampilan pemantauan SYS 2679

Contoh 2680
Kueri tabel sistem, proses, dan id sesi 2687
Tampilan metadata SVV 2687
SVV_ACTIVE_KURSOR 2690
SVV_ALL_KOLOM 2691
SVV_ALL_SCHEMAS 2693
SVV_ALL_TABEL 2695
SVV_ALTER_TABLE_REKOMENDASI 2697
KEBIJAKAN PENYAMARAN SVV_TERLAMPIR 2698
SVV_KOLOM 2701
SVV_COLUMN_PRIVILEGIA 2703
SVV_COPY_JOB_INTEGRASI 2705
SVV_DATABASE_HAK ISTIMEWA 2706
HAK ISTIMEWA SVV_DATASHARE_ 2708
SVV_DATASHARES 2710
SVV_DATASHARE_KONSUMEN 2713
SVV_DATASHARE_OBJECT 2714
SVV_DEFAULT_HAK ISTIMEWA 2716
SVV_DISKUSI 2718
SVV_KOLON_EKSTERNAL_ 2721
SVV_EXTERNAL_DATABASE 2722
SVV_PARTISI_EKSTERNAL 2723
SVV_EXTERNAL_SCHEMAS 2724
SVV_EXTERNAL_TABEL 2726
SVV_FUNGSI_HAK ISTIMEWA 2728
SVV_GEOGRAPHY_COLUMN_ 2729
SVV_GEOMETRI_KOLOM 2731
SVV_IAM_HAK ISTIMEWA 2732
SVV_IDENTITY_PROVIDER 2733
SVV_INTEGRASI 2735
SVV_INTEGRASI_TABEL_PEMETAAN 2737
SVV_INTEGRASI_TABLE_STATE 2740
SVV_KOLUMN_INTERLEAVED_ 2742
SVV_LANGUAGE_KEISTIMEWAAN 2744
SVV_MASKING_POLICY 2745
SVV_ML_MODEL_INFO 2746

SVV_ML_MODEL_HAK ISTIMEWA 2747
SVV_MV_DEPENDENSI 2748
SVV_MV_INFO 2750
SVV_QUERY_DALAM PENERBANGAN 2752
SVV_QUERY_STATE 2754
SVV_REDSHIFT_KOLOM 2757
SVV_REDSHIFT_DATABASE 2760
SVV_REDSHIFT_FUNCTION 2762
SVV_REDSHIFT_SCHEMA_KUOTA 2763
SVV_REDSHIFT_SKEMA 2764
SVV_REDSHIFT_TABEL 2766
HAK ISTIMEWA SVV_RELASI 2768
SVV_RLS_APPLICED_POLICY 2770
SVV_RLS_TERLAMPIR_KEBIJAKAN 2771
SVV_RLS_POLICY 2773
SVV_RLS_RELASI 2774
SVV_ROLE_GRANT 2776
SVV_ROLE 2777
SVV_SCHEMA_HAK ISTIMEWA 2778
SVV_SCHEMA_KUOTA_STATUS 2780
SVV_SYSTEM_HAK ISTIMEWA 2781
SVV_TABLE_INFO 2782
SVV_TABEL 2787
SVV_TRANSAKSI 2788
SVV_USER_GRANT 2791
SVV_USERINFO 2792
SVV_VACUUM_PROGRES 2793
SVV_VACUUM_RINGKASAN 2796
Tampilan pemantauan SYS 2798
SYS_ANALIS_RIWAYAT KOMPRESI 2800
SYS_ANALIS_RIWAYAT 2802
LOG KEBIJAKAN SYS_APLIKASI_MASKING_MASKING_ 2804
SYS_OTOMATIS_OPTIMASI 2806
SYS_AUTO_TABLE_OPTIMASI 2810
SYS_CHILD_QUERY_TEXT 2812
SYS_CONNECTION_LOG 2813

SYS_COPY_JOB 2817
SYS_COPY_JOB_DETAIL 2818
SYS_COPY_JOB_INFO 2820
SYS_COPY_PENGGANTIAN 2820
SYS_DATASHARE_CHANGE_LOG 2822
SYS_DATASHARE_CROSS_REGION_PENGGUNAAN 2825
SYS_DATASHARE_PENGGUNAAN_KONSUMEN 2826
SYS_DATASHARE_PENGGUNAAN_PRODUSEN 2828
SYS_EXTERNAL_QUERY_DETAIL 2829
SYS_EXTERNAL_QUERY_ERROR 2833
SYS_EXTRA_COMPUTE_UNTUK_OPTIMASI_OTOMATIS 2835
SYS_INTEGRASI_AKTIVITAS 2837
SYS_INTEGRASI_TABEL_AKTIVITAS 2838
SYS_INTEGRASI_TABEL_STATE_PERUBAHAN 2840
SYS_LOAD_DETAIL 2841
SYS_LUDF_DETAIL 2844
SYS_LOAD_ERROR_DETAIL 2847
SYS_LOAD_HISTORY 2849
SYS_MV_REFRESH RIWAYAT 2853
SYS_MV_STATUS 2857
SYS_PROSEDUR_PANGGILAN 2860
SYS_PROSEDUR_PESAN-PESAN 2863
SYS_QUERY_DETAIL 2864
SYS_QUERY_JELASKAN 2873
SYS_QUERY_HISTORY 2877
SYS_QUERY_TEKS 2889
SYS_REDSHIFT_TEMPLATE 2891
SYS_RESTORE_LOG 2893
SYS_RESTORE_STATE 2896
SYS_SCHEMA_KUOTA_PELANGGARAN 2899
SYS_USUS_PENGGUNAAN_SERVER 2900
SYS_SESSION_RIWAYAT 2903
SYS_SPATIAL_SIMPLIFIKASI 2905
KESALAHAN SYS_STREAM_SCAN_ 2907
SYS_STREAM_SCAN_STATE 2908
SYS_RIWAYAT TRANSAKSI_ 2910

SYS_UDF_LOG 2912
SYS_UNLOAD_DETAIL 2914
SYS_UNLOAD_HISTORY 2916
SYS_USERLOG 2919
SYS_VACUUM_HISTORY 2921
Pemetaan tampilan sistem untuk bermigrasi ke tampilan pemantauan SYS 2925
SYS_QUERY_HISTORY 2926
SYS_QUERY_DETAIL 2926
SYS_RESTORE_LOG 2928
SYS_RESTORE_STATE 2928
SYS_TRANSACTION_HISTORY 2928
SYS_QUERY_TEXT 2928
SYS_CONNECTION_LOG 2928
SYS_SESSION_HISTORY 2929
SYS_LOAD_DETAIL 2929
SYS_LOAD_HISTORY 2929
SYS_LOAD_ERROR_DETAIL 2929
SYS_UNLOAD_HISTORY 2929
SYS_UNLOAD_DETAIL 2929
SYS_COPY_REPLACEMENTS 2930
SYS_DATASHARE_USAGE_CONSUMER 2930
SYS_DATASHARE_USAGE_PRODUCER 2930
SYS_DATASHARE_CROSS_REGION_USAGE 2930
SYS_DATASHARE_CHANGE_LOG 2930
SYS_EXTERNAL_QUERY_DETAIL 2931
SYS_EXTERNAL_QUERY_ERROR 2931
SYS_VACUUM_HISTORY 2931
SYS_ANALYZE_HISTORY 2931
SYS_ANALYZE_COMPRESSION_HISTORY 2931
SYS_MV_REFRESH_HISTORY 2932
SYS_MV_STATE 2932
SYS_PROCEDURE _CALL 2932
SYS_PROCEDURE _MESSAGES 2932
SYS_UDF_LOG 2932
SYS_USERLOG 2933
SYS_SCHEMA_QUOTA_VIOLATIONS 2933

SYS_SPATIAL_MENYEDERHANAKAN ... 2933
Pemantauan sistem (hanya disediakan) ... 2933
Tampilan STL untuk logging ... 2934
Tabel STV untuk data snapshot ... 3078
Tampilan SVCS untuk cluster penskalaan utama dan konkurensi ... 3135
Tampilan SVL untuk cluster utama ... 3165
Tabel katalog sistem ... 3244
PG_ATTRIBUTE_INFO ... 3245
PG_CLASS_INFO ... 3246
PG_DATABASE_INFO ... 3248
PG_DEFAULT_ACL ... 3248
PG_EXTERNAL_SCHEMA ... 3251
PG_LIBRARY ... 3252
PG_PROC_INFO ... 3253
PG_STATISTIC_INDICATOR ... 3254
PG_TABLE_DEF ... 3255
PG_USER_INFO ... 3258
Menanyakan tabel katalog ... 3259
Referensi konfigurasi ... 3266
Memodifikasi konfigurasi server ... 3267
analisis_ambang_persen ... 3268
Nilai (default dalam huruf tebal) ... 3268
Deskripsi ... 3268
Contoh ... 3269
cast_super_null_on_error ... 3269
Nilai (default dalam huruf tebal) ... 3269
Deskripsi ... 3269
datashare_break_glass_session_var ... 3269
Nilai (default dalam huruf tebal) ... 3269
Deskripsi ... 3269
Contoh ... 3270
datestyle ... 3270
Nilai (default dalam huruf tebal) ... 3270
Deskripsi ... 3269
Contoh ... 3270
default_array_search_null_handling ... 3271

Nilai (default dalam huruf tebal) 3271
Deskripsi 3269
Contoh 3271
default_geometri_encoding 3272
Nilai (default dalam huruf tebal) 3272
Deskripsi 3269
deskripsi_field_name_in_uppercase 3272
Nilai (default dalam huruf tebal) 3272
Deskripsi 3269
Contoh 3270
downcase_delimited_identifier 3273
Nilai (default dalam huruf tebal) 3273
Deskripsi 3269
Catatan Penggunaan 3273
enable_case_sensitive_identifier 3274
Nilai (default dalam huruf tebal) 3274
Deskripsi 3275
Contoh 3275
Catatan Penggunaan 3276
enable_case_sensitive_super_atribut 3278
Nilai (default dalam huruf tebal) 3278
Deskripsi 3278
Contoh 3278
Catatan Penggunaan 3279
mengaktifkan_pembulatan numerik 3280
Nilai (default dalam huruf tebal) 3280
Deskripsi 3280
Contoh 3280
mengaktifkan_hasil_cache_untuk_sesi 3282
Nilai (default dalam huruf tebal) 3282
Deskripsi 3282
Contoh 3282
enable_vacuum_boost 3282
Nilai (default dalam huruf tebal) 3282
Deskripsi 3269
error_pada_pembaruan_nondeterministik 3283

Nilai (default dalam huruf tebal) 3283
Deskripsi 3269
Contoh 3270
extra_float_digits 3283
Nilai (default dalam huruf tebal) 3283
Deskripsi 3283
Contoh 3284
interval_terban_composite_literal 3285
Nilai (default dalam huruf tebal) 3285
Deskripsi 3269
json_serialization_enable 3285
Nilai (default dalam huruf tebal) 3285
Deskripsi 3269
json_serialization_parse_bersarang_string 3286
Nilai (default dalam huruf tebal) 3286
Deskripsi 3269
klaster penskalaan maksimum 3286
Nilai (default dalam huruf tebal) 3286
Deskripsi 3286
maksimum_kursor_hasil_set_ukuran_hasil_ 3287
Nilai (default dalam huruf tebal) 3287
Deskripsi 3287
upayan_login_gagal maksimal 3287
Nilai (default dalam huruf tebal) 3287
Deskripsi 3287
Contoh 3288
mv_enable_aqmv_untuk_sesi 3288
Nilai (default dalam huruf tebal) 3288
Deskripsi 3288
navigasi_super_null_on_error 3288
Nilai (default dalam huruf tebal) 3288
Deskripsi 3269
parse_super_null_on_error 3288
Nilai (default dalam huruf tebal) 3288
Deskripsi 3269
pg_federation_berulang_baca 3289

Nilai (default dalam huruf tebal) 3289
Deskripsi 3269
Contoh 3289
Query_group 3290
Nilai (default dalam huruf tebal) 3290
Deskripsi 3290
search_path 3291
Nilai (default dalam huruf tebal) 3291
Deskripsi 3291
Contoh 3291
Spektrum_enable_pseudo_column 3293
Nilai (default dalam huruf tebal) 3293
Deskripsi 3293
Contoh 3293
enable_spectrum_oid 3293
Nilai (default dalam huruf tebal) 3293
Deskripsi 3293
Contoh 3293
spectrum_query_maxerror 3294
Nilai (default dalam huruf tebal) 3294
Deskripsi 3294
Contoh 3294
statement_timeout 3294
Nilai (default dalam huruf tebal) 3294
Deskripsi 3294
Contoh 3295
menyimpan_proc_log_min_pesan-pesan 3295
Nilai (default dalam huruf tebal) 3295
Deskripsi 3269
timezone 3296
Nilai (default dalam huruf tebal) 3296
Sintaks 3296
Deskripsi 3296
Format zona waktu 3297
Contoh 3299
gunakan_fips_ssl 3299

Nilai (default dalam huruf tebal) 3299
Deskripsi 3269
wlm_query_slot_count 3300
Nilai (default dalam huruf tebal) 3300
Deskripsi 3300
Contoh 3301
Riwayat dokumen 3302
Pembaruan lebih awal 3318

Amazon Redshift tidak akan lagi mendukung penggunaan Python UDF setelah 30 Juni 2026. Kami akan mulai menegakkannya secara bertahap. Untuk informasi lebih lanjut tentang detail opsi akhir masa pakai dan migrasi Python, lihat posting blog yang diterbitkan pada 30 Juni 2025.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Pengantar Amazon Redshift

Selamat datang di Panduan Pengembang Database Amazon Redshift. Panduan ini berfokus untuk membantu Anda memahami cara menggunakan Amazon Redshift untuk membuat dan mengelola gudang data. Jika Anda bekerja dengan database sebagai desainer, pengembang perangkat lunak, atau administrator, panduan ini memberi Anda informasi yang Anda butuhkan untuk merancang, membangun, menanyakan, dan memelihara gudang data Anda.

Amazon Redshift adalah layanan gudang data dengan skala petabyte yang terkelola penuh di cloud. Dengan Amazon Redshift Serverless, Anda dapat mengakses dan menganalisis data tanpa konfigurasi biasa dari gudang data yang disediakan. Sumber daya disediakan secara otomatis dan kapasitas gudang data diskalakan secara cerdas untuk memberikan kinerja yang cepat bahkan untuk beban kerja yang paling menuntut dan tidak dapat diprediksi sekalipun. Anda tidak dikenakan biaya saat gudang data idle, jadi Anda hanya membayar untuk apa yang Anda gunakan. Terlepas dari ukuran dataset, Anda dapat memuat data dan mulai menanyakan langsung di editor kueri Amazon Redshift v2 atau di alat intelijen bisnis (BI) favorit Anda. Nikmati kinerja harga terbaik dan fitur SQL yang sudah dikenal dalam lingkungan tanpa administrasi yang mudah digunakan.

Topik

- Prasyarat untuk menggunakan Amazon Redshift
- Arsitektur Amazon Redshift
- Database sampel

Prasyarat untuk menggunakan Amazon Redshift

Topik ini menjelaskan prasyarat yang Anda perlukan untuk menggunakan Amazon Redshift.

Sebelum Anda menggunakan panduan ini, Anda harus membaca Memulai dengan gudang data Redshift Serverless, yang membahas cara menyelesaikan tugas-tugas berikut.

- Buat gudang data dengan Amazon Redshift Serverless.
- Memuat data sampel dengan editor kueri Amazon Redshift v2
- Memuat data dari Amazon S3.

Anda juga harus tahu cara menggunakan klien SQL Anda dan harus memiliki pemahaman mendasar tentang bahasa SQL.

Arsitektur Amazon Redshift

Topik ini membantu Anda memahami komponen yang membentuk Amazon Redshift.

Gudang data Amazon Redshift adalah kueri dan sistem manajemen basis data relasional kelas perusahaan.

Amazon Redshift mendukung koneksi klien dengan berbagai jenis aplikasi, termasuk intelijen bisnis (BI), pelaporan, data, dan alat analitik.

Saat menjalankan kueri analitik, Anda mengambil, membandingkan, dan mengevaluasi sejumlah besar data dalam operasi multi-tahap untuk menghasilkan hasil akhir.

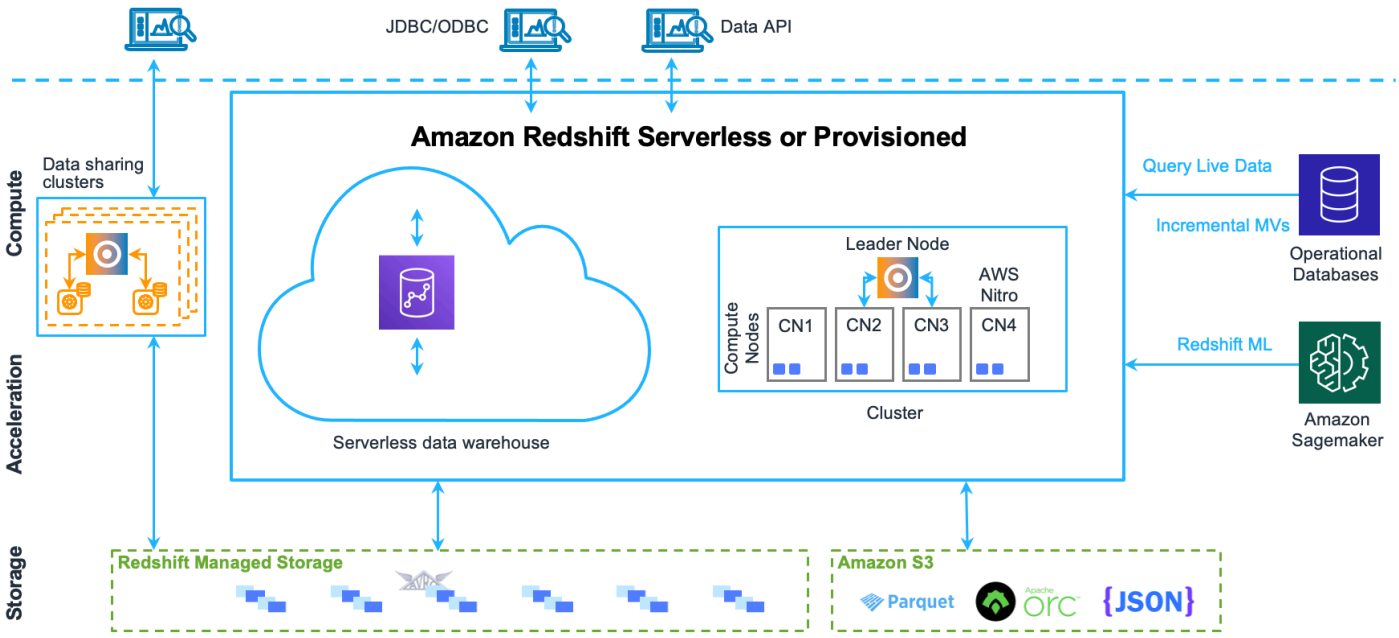
Amazon Redshift mencapai penyimpanan yang efisien dan kinerja kueri yang optimal melalui kombinasi pemrosesan paralel besar-besaran, penyimpanan data kolumnar, dan skema pengkodean kompresi data yang sangat efisien dan ditargetkan. Bagian ini menyajikan pengantar arsitektur sistem Amazon Redshift.

Topik

- Arsitektur sistem gudang data
- Kinerja Amazon Redshift
- Penyimpanan kolumnar
- Manajemen beban kerja
- Menggunakan Amazon Redshift dengan layanan lain

Arsitektur sistem gudang data

Bagian ini menjelaskan komponen yang membentuk arsitektur gudang data Amazon Redshift, seperti yang ditunjukkan pada gambar berikut.



Aplikasi klien

Amazon Redshift terintegrasi dengan berbagai alat pemuatan data dan ETL (ekstrak, transformasi, dan muat) serta pelaporan intelijen bisnis (BI), penambangan data, dan alat analitik. Amazon Redshift didasarkan pada PostgreSQL standar terbuka, sehingga sebagian besar aplikasi klien SQL yang ada akan bekerja hanya dengan sedikit perubahan. Untuk informasi tentang perbedaan penting antara Amazon Redshift SQL dan PostgreSQL, lihat. Amazon Redshift dan PostgreSQL

Cluster

Komponen infrastruktur inti dari gudang data Amazon Redshift adalah cluster.

Cluster terdiri dari satu atau lebih node komputasi. Jika cluster disediakan dengan dua atau lebih node komputasi, node pemimpin tambahan mengoordinasikan node komputasi dan menangani komunikasi eksternal. Aplikasi klien Anda berinteraksi langsung hanya dengan node pemimpin. Node komputasi transparan untuk aplikasi eksternal.

Node pemimpin

Node pemimpin mengelola komunikasi dengan program klien dan semua komunikasi dengan node komputasi. Ini mengurai dan mengembangkan rencana eksekusi untuk melakukan operasi database, khususnya, serangkaian langkah yang diperlukan untuk mendapatkan hasil untuk kueri kompleks. Berdasarkan rencana eksekusi, node pemimpin mengkompilasi kode, mendistribusikan kode yang dikompilasi ke node komputasi, dan menetapkan sebagian data ke setiap node komputasi.

Node pemimpin mendistribusikan pernyataan SQL ke node komputasi hanya ketika kueri mereferensikan tabel yang disimpan pada node komputasi. Semua kueri lain berjalan secara eksklusif pada node pemimpin. Amazon Redshift dirancang untuk mengimplementasikan fungsi SQL tertentu hanya pada node pemimpin. Kueri yang menggunakan salah satu fungsi ini akan mengembalikan kesalahan jika mereferensikan tabel yang berada di node komputasi. Untuk informasi selengkapnya, lihat Fungsi SQL didukung pada node pemimpin.

Hitung node

Node pemimpin mengkompilasi kode untuk elemen individu dari rencana eksekusi dan menetapkan kode ke node komputasi individu. Node komputasi menjalankan kode yang dikompilasi dan mengirim hasil perantara kembali ke node pemimpin untuk agregasi akhir.

Setiap node komputasi memiliki CPU dan memori khusus sendiri, yang ditentukan oleh jenis node. Seiring bertambahnya beban kerja, Anda dapat meningkatkan kapasitas komputasi cluster dengan meningkatkan jumlah node, meningkatkan jenis node, atau keduanya.

Amazon Redshift menyediakan beberapa jenis node untuk kebutuhan komputasi Anda. Untuk detail setiap jenis node, lihat cluster Amazon Redshift di Panduan Manajemen Amazon Redshift.

Penyimpanan Terkelola Redshift

Data gudang data disimpan dalam tingkat penyimpanan terpisah Redshift Managed Storage (RMS). RMS menyediakan kemampuan untuk menskalakan penyimpanan Anda menjadi petabyte menggunakan penyimpanan Amazon S3. Dengan RMS, Anda dapat menskalakan dan membayar komputasi dan penyimpanan secara mandiri, sehingga Anda dapat mengukur cluster hanya berdasarkan kebutuhan komputasi Anda. Secara otomatis menggunakan penyimpanan SSD-based lokal berkinerja tinggi sebagai cache tier-1. Ini juga memanfaatkan pengoptimalan, seperti suhu blok data, usia blok data, dan pola beban kerja untuk memberikan kinerja tinggi sambil menskalakan penyimpanan secara otomatis ke Amazon S3 bila diperlukan tanpa memerlukan tindakan apa pun.

Irisan simpul

Node komputasi dipartisi menjadi irisan. Setiap irisan dialokasikan sebagian dari memori node dan ruang disk, di mana ia memproses sebagian dari beban kerja yang ditetapkan ke node. Node pemimpin mengelola distribusi data ke irisan dan membagi beban kerja untuk setiap kueri atau operasi database lainnya ke irisan. Irisan kemudian bekerja secara paralel untuk menyelesaikan operasi.

Jumlah irisan per node ditentukan oleh ukuran node cluster. Untuk informasi selengkapnya tentang jumlah irisan untuk setiap ukuran node, buka Tentang cluster dan node di Panduan Manajemen Amazon Redshift.

Saat Anda membuat tabel, Anda dapat secara opsional menentukan satu kolom sebagai kunci distribusi. Ketika tabel dimuat dengan data, baris didistribusikan ke irisan node sesuai dengan kunci distribusi yang ditentukan untuk tabel. Memilih kunci distribusi yang baik memungkinkan Amazon Redshift menggunakan pemrosesan paralel untuk memuat data dan menjalankan kueri secara efisien. Untuk informasi tentang memilih kunci distribusi, lihatPilih gaya distribusi terbaik.

Jaringan internal

Amazon Redshift memanfaatkan koneksi bandwidth tinggi, jarak dekat, dan protokol komunikasi khusus untuk menyediakan komunikasi jaringan pribadi berkecepatan sangat tinggi antara node pemimpin dan node komputasi. Node komputasi berjalan pada jaringan terpisah dan terisolasi yang aplikasi klien tidak pernah mengakses secara langsung.

Basis Data

Sebuah cluster berisi satu atau lebih database. Data pengguna disimpan di node komputasi. Klien SQL Anda berkomunikasi dengan node pemimpin, yang pada gilirannya mengoordinasikan query yang dijalankan dengan node komputasi.

Amazon Redshift adalah sistem manajemen basis data relasional (RDBMS), sehingga kompatibel dengan aplikasi RDBMS lainnya. Meskipun menyediakan fungsionalitas yang sama dengan RDBMS biasa, termasuk fungsi pemrosesan transaksi online (OLTP) seperti memasukkan dan menghapus data, Amazon Redshift dioptimalkan untuk analisis kinerja tinggi dan pelaporan kumpulan data yang sangat besar.

Amazon Redshift didasarkan pada PostgreSQL. Amazon Redshift dan PostgreSQL memiliki sejumlah perbedaan yang sangat penting yang perlu Anda perhitungkan saat Anda merancang dan mengembangkan aplikasi gudang data Anda. Untuk informasi tentang perbedaan Amazon Redshift SQL dari PostgreSQL, lihat. Amazon Redshift dan PostgreSQL

Kinerja Amazon Redshift

Topik ini menjelaskan komponen Amazon Redshift yang mendorong kinerja. Memahami komponen ini akan membantu Anda menyesuaikan kinerja dan memecahkan masalah kinerja yang buruk dengan Amazon Redshift.

Amazon Redshift mencapai kueri yang dijalankan sangat cepat dengan menggunakan fitur kinerja ini.

Topik

- Pemrosesan paralel besar-besaran
- Penyimpanan data kolumnar
- Kompresi data
- Pengoptimal kueri
- Caching hasil
- Kode yang dikompilasi

Pemrosesan paralel besar-besaran

Pemrosesan paralel besar-besaran (MPP) memungkinkan menjalankan cepat kueri paling kompleks yang beroperasi pada sejumlah besar data. Beberapa node komputasi menangani semua pemrosesan kueri yang mengarah ke agregasi hasil akhir, dengan setiap inti dari setiap node menjalankan segmen kueri terkompilasi yang sama pada bagian dari seluruh data.

Amazon Redshift mendistribusikan baris tabel ke node komputasi sehingga data dapat diproses secara paralel. Dengan memilih kunci distribusi yang sesuai untuk setiap tabel, Anda dapat mengoptimalkan distribusi data untuk menyeimbangkan beban kerja dan meminimalkan pergerakan data dari node ke node. Untuk informasi selengkapnya, lihat Pilih gaya distribusi terbaik.

Memuat data dari file datar memanfaatkan pemrosesan paralel dengan menyebarkan beban kerja di beberapa node sambil membaca dari beberapa file secara bersamaan. Untuk informasi selengkapnya tentang cara memuat data ke dalam tabel, lihatPraktik terbaik Amazon Redshift untuk memuat data.

Penyimpanan data kolumnar

Penyimpanan kolumnar untuk tabel database secara drastis mengurangi I/O kebutuhan disk secara keseluruhan dan merupakan faktor penting dalam mengoptimalkan kinerja kueri analitik. Untuk informasi selengkapnya, lihat Penyimpanan kolumnar.

Ketika kolom diurutkan dengan tepat, prosesor kueri dapat dengan cepat menyaring sebagian besar blok data. Untuk informasi selengkapnya, lihat Pilih kunci pengurutan terbaik.

Kompresi data

Kompresi data mengurangi kebutuhan penyimpanan, sehingga mengurangi disk I/O, yang meningkatkan kinerja kueri. Ketika Anda menjalankan kueri, data terkompresi dibaca ke dalam

memori, kemudian dikompresi selama menjalankan kueri. Memuat lebih sedikit data ke dalam memori memungkinkan Amazon Redshift mengalokasikan lebih banyak memori untuk menganalisis data. Karena penyimpanan kolumnar menyimpan data serupa secara berurutan, Amazon Redshift dapat menerapkan pengkodean kompresi adaptif yang secara khusus terkait dengan tipe data kolumnar. Cara terbaik untuk mengaktifkan kompresi data pada kolom tabel adalah dengan mengizinkan Amazon Redshift menerapkan pengkodean kompresi optimal saat Anda memuat tabel dengan data. Untuk mempelajari lebih lanjut tentang menggunakan kompresi data otomatis, lihatMemuat tabel dengan kompresi otomatis.

Pengoptimal kueri

Mesin run kueri Amazon Redshift menggabungkan pengoptimal kueri yang juga MPP-aware memanfaatkan penyimpanan data berorientasi kolom. Pengoptimal kueri Amazon Redshift menerapkan peningkatan dan ekstensi signifikan untuk memproses kueri analitik kompleks yang sering menyertakan gabungan multi-tabel, subkueri, dan agregasi. Untuk mempelajari lebih lanjut tentang mengoptimalkan kueri, lihatPenyetelan kinerja kueri.

Caching hasil

Untuk mengurangi runtime kueri dan meningkatkan kinerja sistem, Amazon Redshift menyimpan hasil jenis kueri tertentu dalam memori pada node pemimpin. Saat pengguna mengirimkan kueri, Amazon Redshift memeriksa cache hasil untuk salinan hasil kueri yang valid dan di-cache. Jika kecocokan ditemukan di cache hasil, Amazon Redshift menggunakan hasil yang di-cache dan tidak menjalankan kueri. Caching hasil transparan bagi pengguna.

Caching hasil diaktifkan secara default. Untuk mematikan cache hasil untuk sesi saat ini, atur mengaktifkan_hasil_cache_untuk_sesi parameter `keoff`.

Amazon Redshift menggunakan hasil cache untuk kueri baru jika semua hal berikut benar:

- Pengguna yang mengirimkan kueri memiliki izin akses ke objek yang digunakan dalam kueri.
- Tabel atau tampilan dalam kueri belum dimodifikasi.
- Kueri tidak menggunakan fungsi yang harus dievaluasi setiap kali dijalankan, seperti GETDATE.
- Kueri tidak mereferensikan tabel eksternal Amazon Redshift Spectrum.
- Parameter konfigurasi yang mungkin mempengaruhi hasil kueri tidak berubah.
- Kueri secara sintaksis cocok dengan kueri yang di-cache.

Untuk memaksimalkan efektivitas cache dan penggunaan sumber daya yang efisien, Amazon Redshift tidak menyimpan beberapa set hasil kueri yang besar. Amazon Redshift menentukan apakah hasil kueri akan di-cache berdasarkan sejumlah faktor. Faktor-faktor ini termasuk jumlah entri dalam cache dan jenis instans cluster Amazon Redshift Anda.

Untuk menentukan apakah kueri menggunakan cache hasil, kueri SVL_QLOG tampilan sistem. Jika kueri menggunakan cache hasil, kolom source_query mengembalikan ID kueri dari kueri sumber. Jika hasil caching tidak digunakan, nilai kolom source_query adalah NULL.

Contoh berikut menunjukkan bahwa kueri yang dikirimkan oleh userid 104 dan userid 102 menggunakan cache hasil dari kueri yang dijalankan oleh userid 100.

```
select userid, query, elapsed, source_query from svl_qlog
where userid > 1
order by query desc;

userid | query  | elapsed  | source_query
-------+--------+----------+-------------
   104 | 629035 |       27 |       628919
   104 | 629034 |       60 |       628900
   104 | 629033 |       23 |       628891
   102 | 629017 |  1229393 |
   102 | 628942 |       28 |       628919
   102 | 628941 |       57 |       628900
   102 | 628940 |       26 |       628891
   100 | 628919 | 84295686 |
   100 | 628900 | 87015637 |
   100 | 628891 | 58808694 |
```

Kode yang dikompilasi

Kompilasi kode — Amazon Redshift menghasilkan dan mengkompilasi kode yang dioptimalkan untuk setiap rencana eksekusi kueri. Kode yang dikompilasi berjalan lebih cepat karena menghilangkan overhead menggunakan penerjemah. Untuk meminimalkan latensi untuk kueri baru sambil mempertahankan manfaat kinerja kode yang dikompilasi, Amazon Redshift menggunakan teknik yang disebut komposisi. Komposisi menghasilkan pengaturan ringan dari logika yang sudah ada sebelumnya untuk memproses kueri baru segera, sementara secara bersamaan mengkompilasi kode khusus kueri yang sangat dioptimalkan di latar belakang. Ini menghapus kompilasi dari jalur kritis eksekusi kueri, sehingga kueri baru dimulai lebih cepat dan memberikan kinerja yang konsisten dengan proses berikutnya.

Amazon Redshift juga menggunakan layanan kompilasi tanpa server untuk menskalakan kompilasi kueri di luar sumber daya komputasi cluster Amazon Redshift. Segmen kode yang dikompilasi di-cache baik secara lokal di cluster dan dalam cache jarak jauh yang hampir tidak terbatas yang tetap ada setelah reboot cluster. Eksekusi berikutnya dari kueri yang sama berjalan lebih cepat karena mereka dapat melewati fase kompilasi. Dengan menggunakan layanan kompilasi yang dapat diskalakan, Amazon Redshift mengkompilasi kode secara paralel untuk memberikan kinerja cepat secara konsisten.

Penyimpanan kolumnar

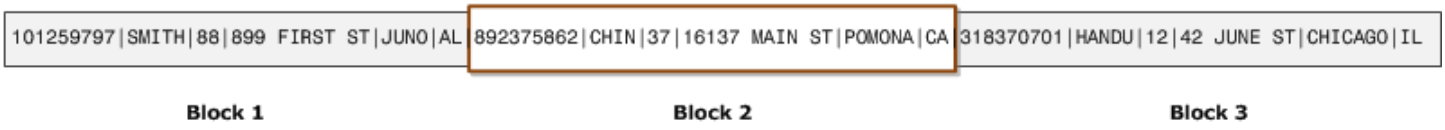
Bagian ini menjelaskan penyimpanan kolumnar, yang merupakan metode yang digunakan Amazon Redshift untuk menyimpan data tabular secara efisien.

Penyimpanan kolom untuk tabel database merupakan faktor penting dalam mengoptimalkan kinerja kueri analitik, karena secara drastis mengurangi I/O kebutuhan disk secara keseluruhan. Ini mengurangi jumlah data yang perlu Anda muat dari disk.

Rangkaian ilustrasi berikut menjelaskan bagaimana penyimpanan data kolumnar mengimplementasikan efisiensi, dan bagaimana hal itu diterjemahkan menjadi efisiensi saat mengambil data ke dalam memori.

Ilustrasi pertama ini menunjukkan bagaimana catatan dari tabel database biasanya disimpan ke dalam blok disk demi baris.

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

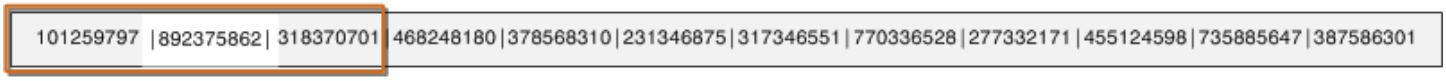


Dalam tabel database relasional tipikal, setiap baris berisi nilai bidang untuk satu catatan. Dalam penyimpanan database baris, blok data menyimpan nilai secara berurutan untuk setiap kolom berturut-turut yang membentuk seluruh baris. Jika ukuran blok lebih kecil dari ukuran catatan, penyimpanan untuk seluruh catatan mungkin membutuhkan lebih dari satu blok. Jika ukuran blok lebih besar dari ukuran catatan, penyimpanan untuk seluruh catatan mungkin memakan waktu kurang dari satu blok, yang mengakibatkan penggunaan ruang disk yang tidak efisien. Dalam aplikasi

pemrosesan transaksi online (OLTP), sebagian besar transaksi melibatkan sering membaca dan menulis semua nilai untuk seluruh catatan, biasanya satu catatan atau sejumlah kecil catatan pada satu waktu. Akibatnya, penyimpanan baris optimal untuk database OLTP.

Ilustrasi berikutnya menunjukkan bagaimana dengan penyimpanan kolumnar, nilai untuk setiap kolom disimpan secara berurutan ke dalam blok disk.

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL



Block 1

Menggunakan penyimpanan kolumnar, setiap blok data menyimpan nilai dari satu kolom untuk beberapa baris. Saat catatan masuk ke sistem, Amazon Redshift secara transparan mengonversi data ke penyimpanan kolom untuk setiap kolom.

Dalam contoh yang disederhanakan ini, menggunakan penyimpanan kolumnar, setiap blok data menyimpan nilai bidang kolom sebanyak tiga kali lebih banyak rekaman daripada penyimpanan berbasis baris. Ini berarti bahwa membaca jumlah nilai kolom yang sama untuk jumlah catatan yang sama memerlukan sepertiga I/O operasi dibandingkan dengan penyimpanan baris. Dalam praktiknya, menggunakan tabel dengan jumlah kolom yang sangat besar dan jumlah baris yang sangat besar, efisiensi penyimpanan bahkan lebih besar.

Keuntungan tambahan adalah bahwa, karena setiap blok menyimpan jenis data yang sama, data blok dapat menggunakan skema kompresi yang dipilih secara khusus untuk tipe data kolom, selanjutnya mengurangi ruang disk dan I/O. Untuk informasi selengkapnya tentang pengkodean kompresi berdasarkan tipe data, lihatPengkodean kompresi.

Penghematan ruang untuk menyimpan data pada disk juga dibawa ke pengambilan dan kemudian menyimpan data itu dalam memori. Karena banyak operasi database hanya perlu mengakses atau beroperasi pada satu atau sejumlah kecil kolom pada satu waktu, Anda dapat menghemat ruang memori dengan hanya mengambil blok untuk kolom yang sebenarnya Anda butuhkan untuk kueri. Di mana transaksi OLTP biasanya melibatkan sebagian besar atau semua kolom dalam satu baris untuk sejumlah kecil catatan, kueri gudang data biasanya hanya membaca beberapa kolom untuk sejumlah besar baris. Ini berarti bahwa membaca jumlah nilai bidang kolom yang sama untuk jumlah

baris yang sama membutuhkan sebagian kecil dari I/O operasi. Ini menggunakan sebagian kecil dari memori yang akan diperlukan untuk memproses blok baris. Dalam praktiknya, menggunakan tabel dengan jumlah kolom yang sangat besar dan jumlah baris yang sangat besar, keuntungan efisiensi secara proporsional lebih besar. Misalnya, sebuah tabel berisi 100 kolom. Kueri yang menggunakan lima kolom hanya perlu membaca sekitar lima persen dari data yang terkandung dalam tabel. Penghematan ini diulang untuk kemungkinan miliaran atau bahkan triliunan catatan untuk database besar. Sebaliknya, database baris akan membaca blok yang berisi 95 kolom yang tidak diperlukan juga.

Ukuran blok database khas berkisar dari 2 KB hingga 32 KB. Amazon Redshift menggunakan ukuran blok 1 MB, yang lebih efisien dan selanjutnya mengurangi jumlah I/O permintaan yang diperlukan untuk melakukan pemuatan database atau operasi lain yang merupakan bagian dari menjalankan kueri.

Manajemen beban kerja

Bagian ini menjelaskan manajemen beban kerja (WLM), yang membantu Anda memahami cara Amazon Redshift menyiapkan dan menjalankan kueri.

Manajemen beban kerja Amazon Redshift (WLM) memungkinkan prioritas manajemen yang fleksibel dalam beban kerja sehingga kueri singkat yang berjalan cepat tidak terjebak dalam antrian di balik kueri yang berjalan lama. Amazon Redshift membuat antrian kueri saat runtime sesuai dengan kelas layanan, yang menentukan parameter konfigurasi untuk berbagai jenis antrian, termasuk antrian sistem internal dan antrian yang dapat diakses pengguna. Dari perspektif pengguna, kelas layanan yang dapat diakses pengguna dan antrian secara fungsional setara. Untuk konsistensi, dokumentasi ini menggunakan istilah antrean untuk berarti kelas layanan yang dapat diakses pengguna serta antrian runtime.

Redshift menawarkan manajemen beban kerja otomatis, yang disebut WLM otomatis, yang disetel untuk menangani berbagai beban kerja dan merupakan default yang disarankan. Dengan WLM otomatis, Redshift menentukan pemanfaatan sumber daya saat kueri tiba dan secara dinamis menentukan apakah akan menjalankannya di cluster utama, pada cluster penskalaan konkurensi, atau mengirim masing-masing ke antrian. (Saat kueri diantri, WLM otomatis memprioritaskan kueri durasi pendek.) WLM otomatis memaksimalkan throughput total, sehingga Anda dapat mempertahankan sumber daya gudang data yang efisien. Anda menjalankan beban kerja tanpa harus khawatir dengan ukurannya atau bagaimana mereka dijadwalkan. WLM otomatis adalah default untuk cluster yang disediakan. Untuk informasi selengkapnya, lihat Mener apkan WLM otomatis.

> **Note**
>
> Kelompok kerja Amazon Redshift Serverless selalu menggunakan WLM otomatis.

Pada saat banyak kueri atau kueri intensif sumber daya sedang berjalan, manajemen beban kerja dapat menskalakan ke sumber daya komputasi tambahan saat beban kerja mengantri pada sumber daya lokal. Penskalaan konkurensi dengan WLM otomatis mendukung kinerja yang konsisten untuk pengguna dan kueri bersamaan yang hampir tidak terbatas.

Cluster yang disediakan Redshift menawarkan WLM manual jika Anda memerlukan pengoptimalan manual yang halus. Di sini, pelanggan mengelola alokasi sumber daya, konkurensi kueri, dan antrian. Ketika kueri berjalan, WLM menetapkan kueri ke antrian sesuai dengan grup pengguna pengguna atau dengan mencocokkan grup kueri yang tercantum dalam konfigurasi antrian. Ini dikonfigurasi dengan label grup kueri yang ditetapkan pengguna. Untuk informasi selengkapnya, lihat Mener apkan manual WLM.

Meskipun Manual WLM dapat disetel dengan baik dari waktu ke waktu agar sesuai dengan pola beban kerja Anda, dalam banyak kasus kami tidak menyarankan penggunaannya karena sifatnya yang statis dapat membuat Anda lebih sulit untuk beradaptasi dengan perubahan beban kerja selama sehari atau selama periode yang lama. Ini membutuhkan lebih banyak pemantauan dan penyetelan berkelanjutan. Selain itu, Manual WLM dalam banyak kasus tidak menggunakan sumber daya komputasi seefisien WLM otomatis, seperti misalnya jika antrian diatur secara manual untuk membatasi memori yang dialokasikan untuk mereka.

Metrik penting untuk mengukur keberhasilan konfigurasi manajemen beban kerja adalah throughput sistem, yang dengan kata lain adalah berapa banyak kueri yang berhasil diselesaikan. Throughput sistem diukur dalam kueri per detik. Untuk informasi selengkapnya tentang metrik sistem, lihat Mem antau kinerja cluster Amazon Redshift.

Cara termudah untuk mengelola konfigurasi WLM Anda adalah dengan menggunakan konsol Amazon Redshift Management. Anda juga dapat menggunakan antarmuka baris perintah Amazon Redshift (CLI) atau Amazon Redshift API. Untuk informasi selengkapnya tentang menerapkan dan menggunakan manajemen beban kerja, lihat Mener apkan manajemen beban kerja.

Menggunakan Amazon Redshift dengan layanan lain

Bagian ini menjelaskan cara menggunakan layanan lain sebagai sumber dan tujuan untuk data Amazon Redshift.

Amazon Redshift terintegrasi dengan AWS layanan lain sehingga Anda dapat memindahkan, mengubah, dan memuat data dengan cepat dan andal, menggunakan fitur keamanan data.

S3

Amazon Simple Storage Service (Amazon S3) adalah layanan web yang menyimpan data di cloud. Amazon Redshift memanfaatkan pemrosesan paralel untuk membaca dan memuat data dari beberapa file data yang disimpan di bucket Amazon S3. Untuk informasi selengkapnya, lihat Memuat data dari Amazon S3.

Anda juga dapat menggunakan pemrosesan paralel untuk mengekspor data dari gudang data Amazon Redshift Anda ke beberapa file data di Amazon S3. Untuk informasi selengkapnya, lihat Membongkar data di Amazon Redshift.

DynamoDB

Amazon DynamoDB adalah layanan database NoSQL yang dikelola sepenuhnya. Anda dapat menggunakan perintah COPY untuk memuat tabel Amazon Redshift dengan data dari satu tabel Amazon DynamoDB. Untuk informasi selengkapnya, lihat Memuat data dari tabel Amazon DynamoDB.

SSH

Anda dapat menggunakan perintah COPY di Amazon Redshift untuk memuat data dari satu atau beberapa host jarak jauh, seperti cluster Amazon EMR, instans Amazon EC2, atau komputer lain. COPY terhubung ke host jarak jauh menggunakan SSH dan menjalankan perintah pada host jarak jauh untuk menghasilkan data. Amazon Redshift mendukung beberapa koneksi simultan. Perintah COPY membaca dan memuat output dari beberapa sumber host secara paralel. Untuk informasi selengkapnya, lihat Memuat data dari host jarak jauh.

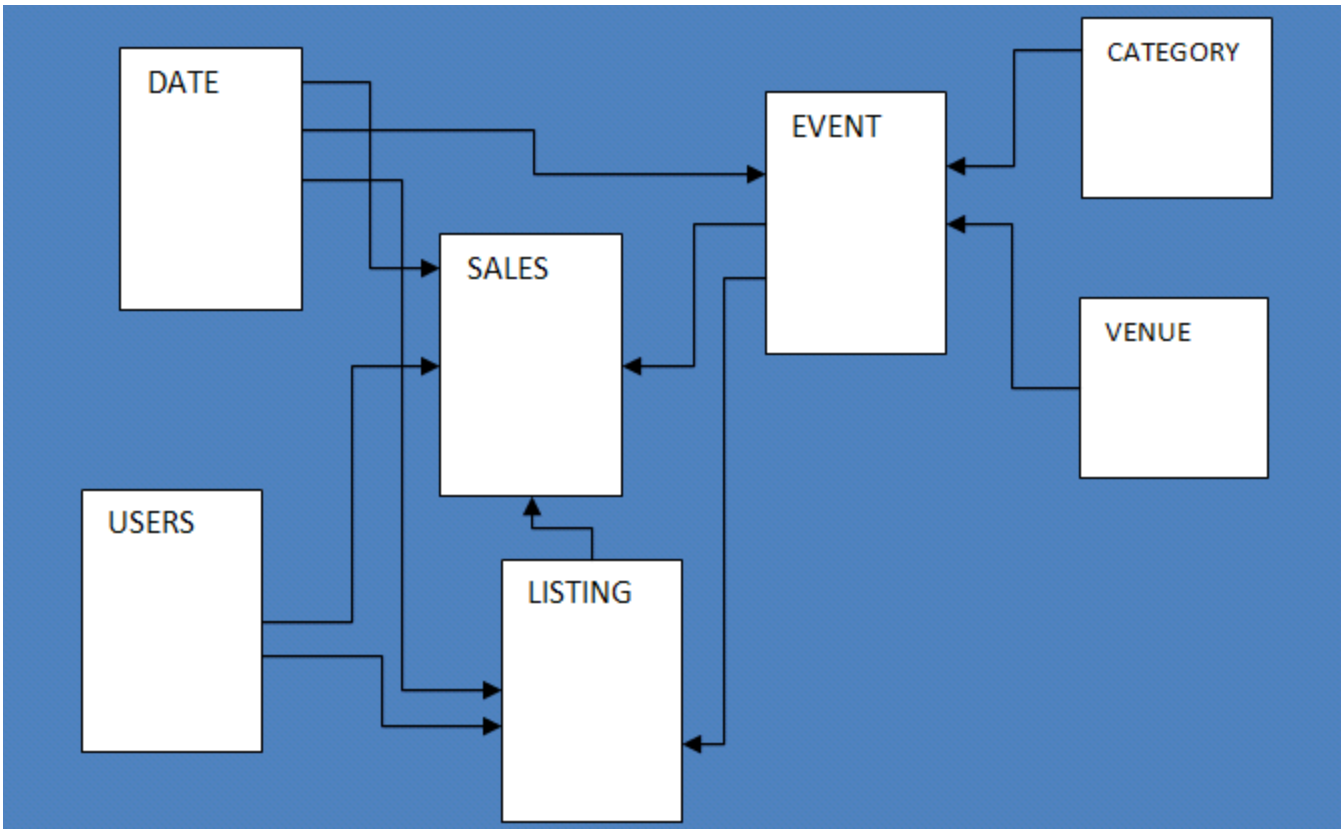
AWS DMS

Anda dapat memigrasikan data ke Amazon Redshift menggunakan AWS Database Migration Service. AWS DMS dapat memigrasikan data Anda ke dan dari database komersial dan sumber terbuka yang paling banyak digunakan seperti Oracle, PostgreSQL, Microsoft SQL Server, Amazon Redshift, cluster Aurora DB, DynamoDB, Amazon S3, MariaDB, dan MySQL. Untuk informasi selengkapnya, lihat Menggunakan database Amazon Redshift sebagai target untuk AWS Database Migration Service.

Database sampel

Bagian ini menjelaskan TICKIT, database contoh yang digunakan contoh dokumentasi Amazon Redshift.

Database kecil ini terdiri dari tujuh tabel: dua tabel fakta dan lima dimensi. Anda dapat memuat kumpulan data TICKIT dengan mengikuti langkah-langkah di Langkah 4: Muat data dari Amazon S3 ke Amazon Redshift di Panduan Memulai Pergeseran Merah Amazon.



Aplikasi database sampel ini membantu analis melacak aktivitas penjualan untuk situs web TICKIT fiksi, tempat pengguna membeli dan menjual tiket secara online untuk acara olahraga, pertunjukan, dan konser. Secara khusus, analis dapat mengidentifikasi pergerakan tiket dari waktu ke waktu, tingkat keberhasilan untuk penjual, dan acara, tempat, dan musim terlaris. Analis dapat menggunakan informasi ini untuk memberikan insentif kepada pembeli dan penjual yang sering mengunjungi situs, untuk menarik pengguna baru, dan untuk mendorong iklan dan promosi.

Misalnya, kueri berikut menemukan lima penjual teratas di San Diego, berdasarkan jumlah tiket yang terjual pada tahun 2008:

```
select sellerid, username, (firstname ||' '|| lastname) as name,
```

```
city, sum(qtysold)
from sales, date, users
where sales.sellerid = users.userid
and sales.dateid = date.dateid
and year = 2008
and city = 'San Diego'
group by sellerid, username, name, city
order by 5 desc
limit 5;

sellerid | username |       name        |   city    | sum
----------+----------+-------------------+-----------+-----
49977 | JJK84WTE | Julie Hanson      | San Diego |  22
19750 | AAS23BDR | Charity Zimmerman | San Diego |  21
29069 | SVL81MEQ | Axel Grant        | San Diego |  17
43632 | VAG08HKW | Griffin Dodson    | San Diego |  16
36712 | RXT40MKU | Hiram Turner      | San Diego |  14
(5 rows)
```

Database yang digunakan untuk contoh dalam panduan ini berisi kumpulan data kecil; dua tabel fakta masing-masing berisi kurang dari 200.000 baris, dan dimensi berkisar dari 11 baris dalam tabel KATEGORI hingga sekitar 50.000 baris dalam tabel USERS.

Secara khusus, contoh database dalam panduan ini menunjukkan fitur utama dari desain tabel Amazon Redshift:

- Distribusi data
- Urutkan data
- Kompresi kolumnar

Untuk informasi tentang skema tabel dalam database TICKIT, pilih tab berikut:

CATEGORY table

Nama kolom	Jenis data	Deskripsi
CATID	SMALLINT	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili jenis acara tertentu di mana tiket dibeli dan dijual.

Nama kolom	Jenis data	Deskripsi
KELOMPOK KUCING	VARCHAR(10)	Nama deskriptif untuk sekelompok acara, seperti **`Shows`** dan**`Sports`**.
NAMA KUCING	VARCHAR(10)	Nama deskriptif singkat untuk jenis acara dalam grup, seperti **`Opera`** dan**`Musicals`**.
CATDESC	VARCHAR (50)	Nama deskriptif yang lebih panjang untuk jenis acara, seperti**`Musical theatre`**.

DATE table

Nama kolom	Jenis data	Deskripsi
DATEID	SMALLINT	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili satu hari dalam tahun kalender.
KALDAT	DATE	Tanggal kalender, seperti**`2008-06-24`** .
DAY	ARANG (3)	Hari dalam seminggu (bentuk pendek), seperti**`SA`**.
MINGGU	SMALLINT	Nomor minggu, seperti**`26`**.
MONTH	ARANG (5)	Nama bulan (bentuk pendek), seperti**`JUN`**.
QTR	ARANG (5)	Nomor seperempat (**`1`**melalui**`4`**).
YEAR	SMALLINT	Four-digit tahun (**`2008`**).
LIBURAN	BOOLEAN	Bendera yang menunjukkan apakah hari itu adalah hari libur umum (U.S.).

EVENT table

Nama kolom	Jenis data	Deskripsi
EVENTID	INTEGER	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili acara terpisah yang berlangsung di tempat tertentu pada waktu tertentu.
VENUEID	SMALLINT	Foreign-key referensi ke tabel VENUE.
CATID	SMALLINT	Foreign-key referensi ke tabel KATEGORI.
DATEID	SMALLINT	Foreign-key referensi ke tabel DATE.
NAMA ACARA	VARCHAR (200)	Nama acara, seperti **`Hamlet`** atau**`La Traviata`**.
WAKTU MULAI	TIMESTAMP	Tanggal penuh dan waktu mulai untuk acara tersebut, seperti**`2008-10-10 19:30:00`** .

VENUE table

Nama kolom	Jenis data	Deskripsi
VENUEID	SMALLINT	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili tempat tertentu di mana acara berlangsung.
NAMA VENUENAME	VARCHAR (100)	Nama tempat yang tepat, seperti**`Cleveland Browns Stadium`**.
KELIMPAHAN	VARCHAR (30)	Nama kota, seperti **`Cleveland`**
VENUESTATE	ARANG (2)	Two-letter singkatan negara bagian atau provinsi (Amerika Serikat dan Kanada), seperti**`OH`**.
KURSI TEMPAT DUDUK	INTEGER	Jumlah maksimum kursi yang tersedia di venue, jika diketahui, seperti**`73200`**. Untuk tujuan

Nama kolom	Jenis data	Deskripsi
		demonstrasi, kolom ini berisi beberapa nilai nol dan nol.

USERS table

Nama kolom	Jenis data	Deskripsi
USERID	INTEGER	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili pengguna terdaftar (pembeli atau penjual atau keduanya) yang telah mendaftarkan atau membeli tiket untuk setidaknya satu acara.
NAMA PENGGUNA	ARANG (8)	Nama pengguna alfanumerik 8 karakter, seperti. **`PGL08LJI`**
NAMA DEPAN	VARCHAR (30)	Nama depan pengguna, seperti**`Victor`**.
NAMA BELAKANG	VARCHAR (30)	Nama belakang pengguna, seperti**`Hernandez`** .
KOTA	VARCHAR (30)	Kota asal pengguna, seperti**`Naperville`** .
STATE	ARANG (2)	Negara asal pengguna, seperti**`GA`**.
Email	VARCHAR (100)	Alamat email pengguna; kolom ini berisi nilai-nil ai Latin acak, seperti**`turpis@accumsanlaoreet.org`** .
TELEPON	ARANG (14)	Nomor telepon 14 karakter pengguna, seperti**`(818) 765-4255`**.
LIKESPORTS,...	BOOLEAN	Serangkaian 10 kolom berbeda yang mengident ifikasi suka dan tidak suka **`true`** dan **`false`** nilai pengguna.

LISTING table

Nama kolom	Jenis data	Deskripsi
LISTID	INTEGER	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili daftar batch tiket untuk acara tertentu.
SELLERID	INTEGER	Foreign-key referensi ke tabel USERS, mengident ifikasi pengguna yang menjual tiket.
EVENTID	INTEGER	Foreign-key referensi ke tabel EVENT.
DATEID	SMALLINT	Foreign-key referensi ke tabel DATE.
NUMTICKETS	SMALLINT	Jumlah tiket yang tersedia untuk dijual, seperti **2** atau**20**.
PRICEPERT ICKET	DESIMAL (8,2)	Harga tetap dari tiket individu, seperti **27.00** atau**206.00**.
TOTALHARGA	DESIMAL (8,2)	Total harga untuk listing ini (NUMTICKETS*PRICEP ERTICKET).
DAFTAR WAKTU	TIMESTAMP	Tanggal dan waktu penuh ketika daftar diposting, seperti**2008-03-18 07:19:35** .

SALES table

Nama kolom	Jenis data	Deskripsi
SALESID	INTEGER	Kunci primer, nilai ID unik untuk setiap baris. Setiap baris mewakili penjualan satu atau lebih tiket untuk acara tertentu, seperti yang ditawarkan dalam daftar tertentu.
LISTID	INTEGER	Foreign-key referensi ke tabel LISTING.

Nama kolom	Jenis data	Deskripsi
SELLERID	INTEGER	Foreign-key referensi ke tabel USERS (pengguna yang menjual tiket).
PEMBELI	INTEGER	Foreign-key referensi ke tabel USERS (pengguna yang membeli tiket).
EVENTID	INTEGER	Foreign-key referensi ke tabel EVENT.
DATEID	SMALLINT	Foreign-key referensi ke tabel DATE.
QTYSOLD	SMALLINT	Jumlah tiket yang terjual, dari **1** ke**8**. (Maksimal 8 tiket dapat dijual dalam satu transaksi.)
PRICEPAID	DESIMAL (8,2)	Total harga yang dibayarkan untuk tiket, seperti **75.00** atau**488.00**. Harga individual tiket adalah PRICEPAID /QTYSOLD.
KOMISI	DESIMAL (8,2)	Komisi 15% yang dikumpulkan bisnis dari penjualan, seperti **11.25** atau**73.20**. Penjual menerima 85% dari nilai PRICEPAID.
WAKTU PENJUALAN	TIMESTAMP	Tanggal dan waktu penuh ketika penjualan selesai, seperti**2008-05-24 06:21:47** .

Praktik terbaik Amazon Redshift

Berikut ini, Anda dapat menemukan praktik terbaik untuk merencanakan bukti konsep, merancang tabel, memuat data ke dalam tabel, dan menulis kueri untuk Amazon Redshift, dan juga diskusi tentang bekerja dengan Amazon Redshift Advisor.

Amazon Redshift tidak sama dengan sistem database SQL lainnya. Untuk sepenuhnya menyadari manfaat arsitektur Amazon Redshift, Anda harus secara khusus merancang, membangun, dan memuat tabel Anda untuk menggunakan pemrosesan paralel besar-besaran, penyimpanan data kolumnar, dan kompresi data kolumnar. Jika waktu pemuatan data dan eksekusi kueri lebih lama dari yang Anda harapkan, atau lebih lama dari yang Anda inginkan, Anda mungkin mengabaikan informasi utama.

Jika Anda adalah pengembang database SQL yang berpengalaman, kami sangat menyarankan Anda meninjau topik ini sebelum Anda mulai mengembangkan gudang data Amazon Redshift Anda.

Jika Anda baru mengembangkan database SQL, topik ini bukan tempat terbaik untuk memulai. Kami menyarankan Anda memulai dengan membaca perintah Jalankan untuk menentukan dan menggunakan database di gudang data Anda di Panduan Mem ulai Amazon Redshift, dan mencoba contohnya sendiri.

Dalam topik ini, Anda dapat menemukan ikhtisar prinsip-prinsip pengembangan yang paling penting, bersama dengan tips, contoh, dan praktik terbaik untuk menerapkan prinsip-prinsip tersebut. Tidak ada praktik tunggal yang dapat diterapkan untuk setiap aplikasi. Evaluasi semua opsi Anda sebelum menyelesaikan desain database. Untuk informasi selengkapnyaPengoptimalan tabel otomatis, lihatMemuat data di Amazon Redshift,Penyetelan kinerja kueri,, dan bagian referensi.

Topik

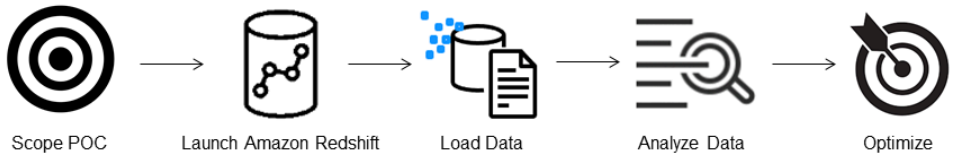
- Melakukan bukti konsep (POC) untuk Amazon Redshift
- Praktik terbaik Amazon Redshift untuk merancang tabel
- Praktik terbaik Amazon Redshift untuk memuat data
- Praktik terbaik Amazon Redshift untuk merancang kueri
- Ikuti rekomendasi dari Amazon Redshift Advisor

Melakukan bukti konsep (POC) untuk Amazon Redshift

Amazon Redshift adalah gudang data cloud populer, yang menawarkan layanan berbasis cloud yang dikelola sepenuhnya yang terintegrasi dengan danau data Amazon Simple Storage Service organisasi, aliran real-time, alur kerja pembelajaran mesin (ML), alur kerja transaksional, dan banyak lagi. Bagian berikut memandu Anda melalui proses melakukan bukti konsep (POC) di Amazon Redshift. Informasi di sini membantu Anda menetapkan tujuan untuk POC Anda, dan memanfaatkan alat yang dapat mengotomatiskan penyediaan dan konfigurasi layanan untuk POC Anda.

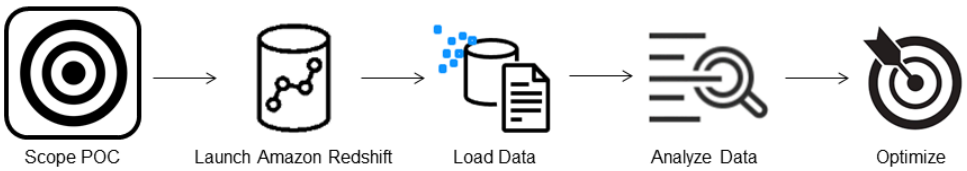
> **Note**
>
> Untuk salinan informasi ini sebagai PDF, pilih tautan J alankan POC Redshift Anda sendiri di halaman sumber daya Amazon Redshift.

Saat melakukan POC Amazon Redshift, Anda menguji, membuktikan, dan mengadopsi fitur mulai dari kemampuan keamanan terbaik di kelasnya, penskalaan elastis, integrasi dan penyerapan yang mudah, dan opsi arsitektur data terdesentralisasi yang fleksibel.



Ikuti langkah-langkah ini untuk melakukan POC yang sukses.

Langkah 1: Lingkup POC Anda



Saat melakukan POC, Anda dapat memilih untuk menggunakan data Anda sendiri, atau Anda dapat memilih untuk menggunakan kumpulan data pembandingan. Ketika Anda memilih data Anda

sendiri, Anda menjalankan kueri Anda sendiri terhadap data tersebut. Dengan data benchmarking, kueri sampel disediakan dengan tolok ukur. Lihat Gunakan kumpulan data sampel untuk detail selengkapnya jika Anda belum siap melakukan POC dengan data Anda sendiri.

Secara umum, kami sarankan menggunakan data dua minggu untuk Amazon Redshift POC.

Mulailah dengan melakukan hal berikut:

1. Identifikasi kebutuhan bisnis dan fungsional Anda, lalu kerjakan mundur. Contoh umum adalah: kinerja yang lebih cepat, biaya lebih rendah, menguji beban kerja atau fitur baru, atau perbandingan antara Amazon Redshift dan gudang data lainnya.
2. Tetapkan target spesifik yang menjadi kriteria keberhasilan POC. Misalnya, dari kinerja yang lebih cepat, buat daftar lima proses teratas yang ingin Anda akselerasi, dan sertakan waktu berjalan saat ini bersama dengan waktu berjalan yang Anda butuhkan. Ini bisa berupa laporan, kueri, proses ETL, konsumsi data, atau apa pun masalah Anda saat ini.
3. Identifikasi ruang lingkup spesifik dan artefak yang diperlukan untuk menjalankan pengujian. Kumpulan data apa yang Anda perlukan untuk bermigrasi atau terus diserap ke Amazon Redshift, dan kueri serta proses apa yang diperlukan untuk menjalankan pengujian untuk mengukur dengan kriteria keberhasilan? Ada dua cara untuk melakukan hal ini:

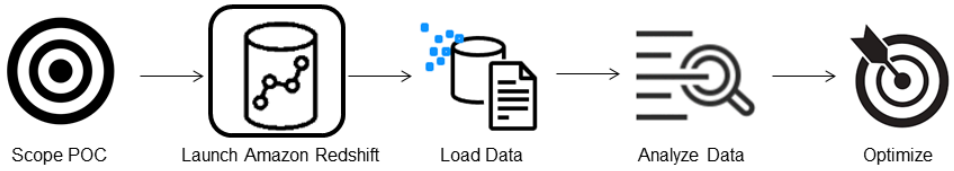
   Bawa data Anda sendiri

   - Untuk menguji data Anda sendiri, buat daftar artefak data minimum yang layak yang diperlukan untuk menguji kriteria keberhasilan Anda. Misalnya, jika gudang data Anda saat ini memiliki 200 tabel, tetapi laporan yang ingin Anda uji hanya membutuhkan 20, POC Anda dapat dijalankan lebih cepat dengan hanya menggunakan subset tabel yang lebih kecil.

   Gunakan kumpulan data sampel

   - Jika Anda belum menyiapkan kumpulan data sendiri, Anda masih dapat memulai melakukan POC di Amazon Redshift dengan menggunakan kumpulan data benchmark standar industri seperti TPC-DS atau TPC-H dan menjalankan kueri pembandingan sampel untuk memanfaatkan kekuatan Amazon Redshift. Kumpulan data ini dapat diakses dari dalam gudang data Amazon Redshift Anda setelah dibuat. Untuk petunjuk terperinci tentang cara mengakses kumpulan data ini dan kueri sampel, lihatLangkah 2: Luncurkan Amazon Redshift.

Langkah 2: Luncurkan Amazon Redshift



Amazon Redshift mempercepat waktu Anda untuk mendapatkan wawasan dengan pergudangan data cloud yang cepat, mudah, dan aman dalam skala besar. Anda dapat memulai dengan cepat dengan meluncurkan gudang Anda di konsol Redshift Serverless dan beralih dari data ke wawasan dalam hitungan detik. Dengan Redshift Serverless, Anda dapat fokus memberikan hasil bisnis Anda tanpa khawatir mengelola gudang data Anda.

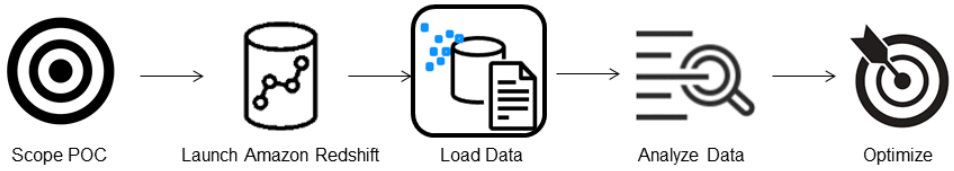
Siapkan Amazon Redshift Serverless

Pertama kali Anda menggunakan Redshift Serverless, konsol mengarahkan Anda melalui langkah-langkah yang diperlukan untuk meluncurkan gudang Anda. Anda mungkin juga memenuhi syarat untuk mendapatkan kredit untuk penggunaan Redshift Serverless di akun Anda. Untuk informasi selengkapnya tentang memilih uji coba gratis, lihat uji coba gratis Amazon Redshift. Ikuti langkah-langkah di Membuat gudang data dengan Redshift Serverless di Panduan Memulai Amazon Redshift untuk membuat gudang data dengan Redshift Serverless. Jika Anda tidak memiliki dataset yang ingin Anda muat, panduan ini juga berisi langkah-langkah tentang cara memuat kumpulan data sampel.

Jika sebelumnya Anda telah meluncurkan Redshift Serverless di akun Anda, ikuti langkah-langkah dalam Membuat grup kerja dengan namespace di Panduan Manajemen Amazon Redshift. Setelah gudang Anda tersedia, Anda dapat memilih untuk memuat data sampel yang tersedia di Amazon Redshift. Untuk informasi tentang menggunakan editor kueri Amazon Redshift v2 untuk memuat data, lihat Memuat data sampel di Panduan Manajemen Amazon Redshift.

Jika Anda membawa data Anda sendiri alih-alih memuat kumpulan data sampel, lihatLangkah 3: Muat data Anda.

Langkah 3: Muat data Anda



Setelah meluncurkan Redshift Serverless, langkah selanjutnya adalah memuat data Anda untuk POC. Baik Anda mengunggah file CSV sederhana, mengambil data semi-terstruktur dari S3, atau streaming data secara langsung, Amazon Redshift memberikan fleksibilitas untuk memindahkan data ke tabel Amazon Redshift dengan cepat dan mudah dari sumbernya.

Pilih salah satu metode berikut untuk memuat data Anda.

Unggah file lokal

Untuk konsumsi dan analisis yang cepat, Anda dapat menggunakan editor kueri Amazon Redshift v2 untuk memuat file data dengan mudah dari desktop lokal Anda. Ini memiliki kemampuan untuk memproses file dalam berbagai format seperti CSV, JSON, AVRO, PARQUET, ORC, dan banyak lagi. Untuk mengaktifkan pengguna Anda, sebagai administrator, untuk memuat data dari desktop lokal menggunakan editor kueri v2, Anda harus menentukan bucket Amazon S3 umum, dan akun pengguna harus dikonfigurasi dengan izin yang tepat. Anda dapat mengikuti Pem uatan data dengan mudah dan aman di Amazon Redshift menggunakan Query Editor V2 untuk panduan langkah demi langkah.

Memuat file Amazon S3

Untuk memuat data dari bucket Amazon S3 ke Amazon Redshift, mulailah dengan menggunakan perintah COPY, tentukan lokasi sumber Amazon S3 dan target tabel Amazon Redshift. Pastikan peran dan izin IAM dikonfigurasi dengan benar untuk memungkinkan akses Amazon Redshift ke bucket Amazon S3 yang ditunjuk. Ikuti Tutorial: Memuat data dari Amazon S3 untuk panduan langkah demi langkah. Anda juga dapat memilih opsi Load data di query editor v2 untuk langsung memuat data dari bucket S3 Anda.

Konsumsi data terus menerus

Autocopy (dalam pratinjau) adalah ekstensi dari perintah COPY dan mengotomatiskan pemuatan data berkelanjutan dari bucket Amazon S3. Saat Anda membuat pekerjaan penyalinan, Amazon

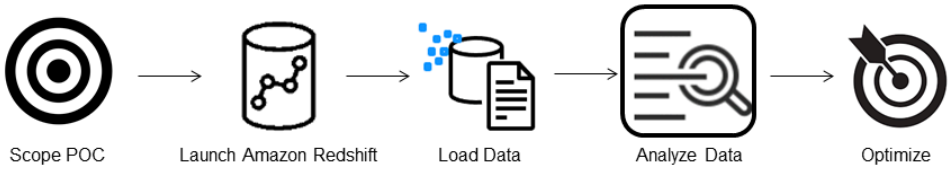
Redshift mendeteksi kapan file Amazon S3 baru dibuat di jalur tertentu, dan kemudian memuatnya secara otomatis tanpa campur tangan Anda. Amazon Redshift melacak file yang dimuat untuk memverifikasi bahwa file tersebut dimuat hanya satu kali. Untuk petunjuk tentang cara membuat pekerjaan penyalinan, lihat SALIN PEKERJAAN

> **Note**
>
> Autocopy saat ini dalam pratinjau dan hanya didukung di cluster yang disediakan secara spesifik. Wilayah AWS Untuk membuat cluster pratinjau untuk autokopi, lihatBuat integrasi acara S3 untuk menyalin file secara otomatis dari bucket Amazon S3.

Muat data streaming Anda

Penyerapan streaming menyediakan penyerapan data aliran dengan latensi rendah dan kecepatan tinggi dari Amazon Kinesis Data Streams dan Amazon Managed Streaming untuk Apache Kafka ke Amazon Redshift. Konsumsi streaming Amazon Redshift menggunakan tampilan terwujud, yang diperbarui langsung dari aliran menggunakan penyegaran otomatis. Tampilan terwujud memetakan ke sumber data aliran. Anda dapat melakukan pemfilteran dan agregasi pada data aliran sebagai bagian dari definisi tampilan yang terwujud. Untuk panduan langkah demi langkah untuk memuat data dari aliran, lihat Mem ulai dengan Amazon Kinesis Data Streaming atau Mem ulai dengan Amazon Managed Streaming untuk Apache Kaf ka.

Langkah 4: Analisis data Anda



Setelah membuat grup kerja dan namespace Redshift Serverless Anda, dan memuat data Anda, Anda dapat segera menjalankan kueri dengan membuka editor Ku eri v2 dari panel navigasi konsol Redshift Serverless. https://console.aws.amazon.com/redshiftv2/home?#serverless-dashboard Anda dapat menggunakan editor kueri v2 untuk menguji fungsionalitas kueri atau kinerja kueri terhadap kumpulan data Anda sendiri.

Kueri menggunakan editor kueri Amazon Redshift v2

Anda dapat mengakses editor kueri v2 dari konsol Amazon Redshift. Lihat Menyederhanakan analisis data Anda dengan editor kueri Amazon Redshift v2 untuk panduan lengkap tentang cara mengonfigurasi, menghubungkan, dan menjalankan kueri dengan editor kueri v2.

Atau, jika Anda ingin menjalankan uji beban sebagai bagian dari POC Anda, Anda dapat melakukannya dengan langkah-langkah berikut untuk menginstal dan menjalankan Apache JMeter.

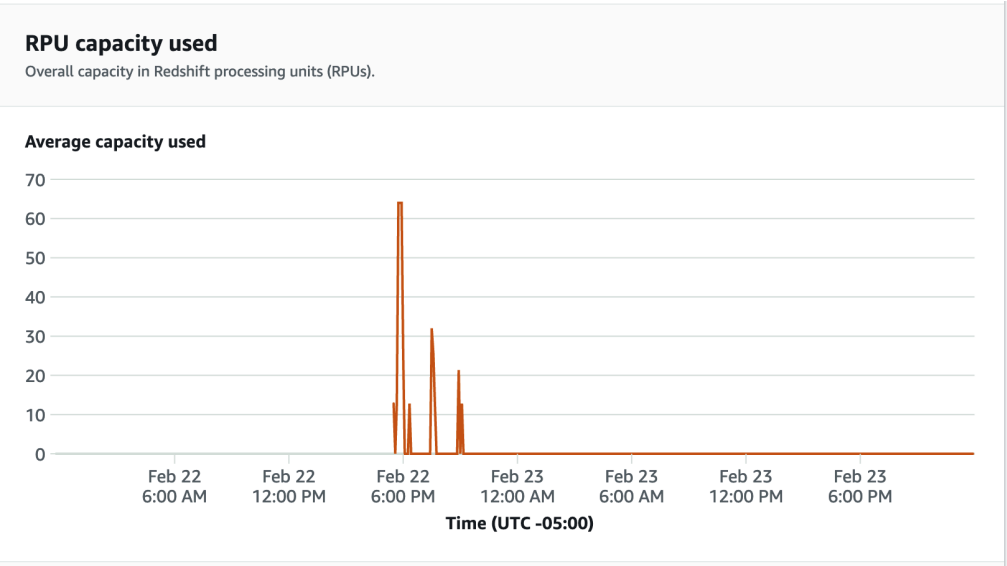
Jalankan uji beban menggunakan Apache JMeter

Untuk melakukan uji beban untuk mensimulasikan pengguna “N” yang mengirimkan kueri secara bersamaan ke Amazon Redshift, Anda dapat menggunakan Apache JMeter, alat berbasis Java open-source.

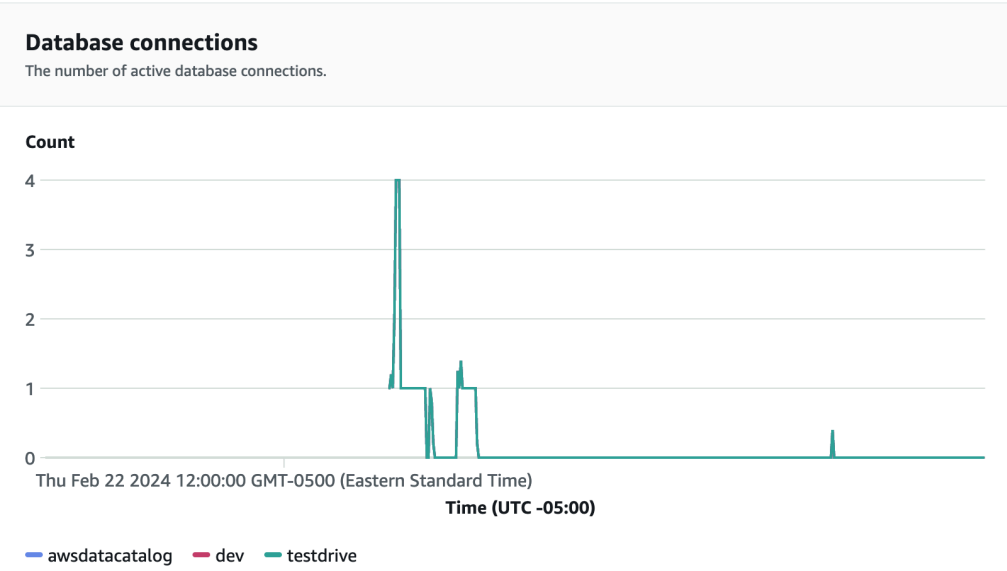
Untuk menginstal dan mengonfigurasi Apache JMeter agar dijalankan terhadap workgroup Redshift Serverless Anda, ikuti petunjuk di Otom atisasi pengujian beban Amazon Redshift dengan Analytics Automation Toolkit. AWS Ini menggunakan AWS Analytics Automation toolkit (AAA), utilitas sumber terbuka untuk menerapkan solusi Redshift secara dinamis, untuk meluncurkan sumber daya ini secara otomatis. Jika Anda telah memuat data Anda sendiri ke Amazon Redshift, pastikan untuk melakukan opsi Langkah #5 - Kustomisasi SQL, untuk memastikan Anda memberikan pernyataan SQL yang sesuai yang ingin Anda uji terhadap tabel Anda. Uji masing-masing pernyataan SQL ini satu kali menggunakan query editor v2 untuk memastikan mereka berjalan tanpa kesalahan.

Setelah Anda menyelesaikan penyesuaian pernyataan SQL dan menyelesaikan rencana pengujian, simpan dan jalankan rencana pengujian terhadap kelompok kerja Redshift Serverless Anda. Untuk memantau kemajuan pengujian Anda, buka konsol Redshift Serverless, navigasikan ke Query and database monitoring, pilih tab Riwayat kueri dan lihat informasi tentang kueri Anda.

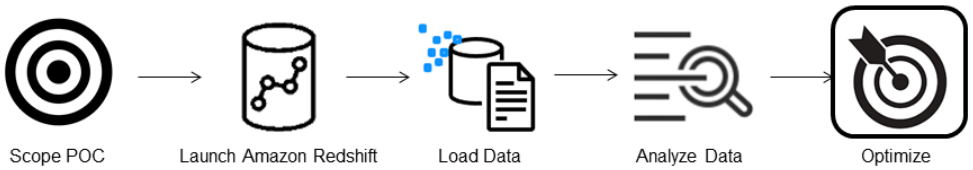
Untuk metrik kinerja, pilih tab Kin erja database di konsol Redshift Serverless, untuk memantau metrik seperti Koneksi Database dan pemanfaatan CPU. Di sini Anda dapat melihat grafik untuk memantau kapasitas RPU yang digunakan dan mengamati bagaimana Redshift Serverless secara otomatis menskalakan untuk memenuhi tuntutan beban kerja bersamaan saat uji beban berjalan pada kelompok kerja Anda.



Koneksi database adalah metrik lain yang berguna untuk dipantau saat menjalankan uji beban untuk melihat bagaimana kelompok kerja Anda menangani banyak koneksi bersamaan pada waktu tertentu untuk memenuhi tuntutan beban kerja yang meningkat.



Langkah 5: Optimalkan



Amazon Redshift memberdayakan puluhan ribu pengguna untuk memproses data exabyte setiap hari dan memperkuat beban kerja analitik mereka dengan menawarkan berbagai konfigurasi dan fitur untuk mendukung kasus penggunaan individu. Saat memilih di antara opsi ini, pelanggan mencari alat yang membantu mereka menentukan konfigurasi gudang data yang paling optimal untuk mendukung beban kerja Amazon Redshift mereka.

Uji coba

Anda dapat menggunakan Test Drive untuk secara otomatis memutar ulang beban kerja yang ada pada konfigurasi potensial dan menganalisis output yang sesuai untuk mengevaluasi target optimal untuk memigrasikan beban kerja Anda. Lihat Menemukan konfigurasi Amazon Redshift terbaik untuk beban kerja Anda menggunakan Redshift Test Drive untuk informasi tentang menggunakan Test Drive untuk mengevaluasi konfigurasi Amazon Redshift yang berbeda.

Praktik terbaik Amazon Redshift untuk merancang tabel

Saat Anda merencanakan database Anda, keputusan desain tabel kunci tertentu sangat mempengaruhi kinerja kueri secara keseluruhan. Pilihan desain ini juga memiliki efek signifikan pada persyaratan penyimpanan, yang pada gilirannya mempengaruhi kinerja kueri dengan mengurangi jumlah I/O operasi dan meminimalkan memori yang diperlukan untuk memproses kueri.

Di bagian ini, Anda dapat menemukan ringkasan keputusan desain yang paling penting dan praktik terbaik untuk mengoptimalkan kinerja kueri. Pengoptimalan tabel otomatismemberikan penjelasan yang lebih rinci dan contoh opsi desain tabel.

Topik

- Pilih kunci pengurutan terbaik
- Pilih gaya distribusi terbaik
- Biarkan COPY memilih pengkodean kompresi
- Tentukan kunci primer dan kendala kunci asing
- Gunakan ukuran kolom sekecil mungkin
- Gunakan date/time tipe data untuk kolom tanggal

Pilih kunci pengurutan terbaik

Amazon Redshift menyimpan data Anda di disk dalam urutan yang diurutkan sesuai dengan kunci pengurutan. Pengoptimal kueri Amazon Redshift menggunakan urutan saat menentukan rencana kueri yang optimal.

> **Note**
>
> Saat Anda menggunakan pengoptimalan tabel otomatis, Anda tidak perlu memilih kunci pengurutan tabel Anda. Untuk informasi selengkapnya, lihat Pengoptimalan tabel otomatis.

Beberapa saran untuk pendekatan terbaik berikut:

- Agar Amazon Redshift memilih urutan yang sesuai, tentukan AUTO tombol pengurutan.
- Jika data terbaru paling sering ditanyakan, tentukan kolom stempel waktu sebagai kolom utama untuk kunci pengurutan.

  Kueri lebih efisien karena mereka dapat melewati seluruh blok yang berada di luar rentang waktu.
- Jika Anda melakukan pemfilteran rentang sering atau pemfilteran kesetaraan pada satu kolom, tentukan kolom itu sebagai kunci pengurutan.

  Amazon Redshift dapat melewatkan membaca seluruh blok data untuk kolom itu. Itu dapat melakukannya karena melacak nilai kolom minimum dan maksimum yang disimpan di setiap blok dan dapat melewati blok yang tidak berlaku untuk rentang predikat.
- Jika Anda sering bergabung dengan tabel, tentukan kolom gabungan sebagai kunci pengurutan dan kunci distribusi.

  Melakukan hal ini memungkinkan pengoptimal kueri untuk memilih penggabungan sortir alih-alih gabungan hash yang lebih lambat. Karena data sudah diurutkan pada kunci gabungan, pengoptimal kueri dapat melewati fase pengurutan penggabungan pengurutan.

Pilih gaya distribusi terbaik

Saat Anda menjalankan kueri, pengoptimal kueri mendistribusikan ulang baris ke node komputasi sesuai kebutuhan untuk melakukan gabungan dan agregasi apa pun. Tujuan dalam memilih gaya distribusi tabel adalah untuk meminimalkan dampak dari langkah redistribusi dengan menemukan data di tempat yang diperlukan sebelum query dijalankan.

> **Note**
>
> Saat Anda menggunakan pengoptimalan tabel otomatis, Anda tidak perlu memilih gaya distribusi tabel Anda. Untuk informasi selengkapnya, lihat Pengoptimalan tabel otomatis.

Beberapa saran untuk pendekatan terbaik berikut:

1. Bagikan tabel fakta dan satu tabel dimensi pada kolom umum mereka.

   Tabel fakta Anda hanya dapat memiliki satu kunci distribusi. Setiap tabel yang bergabung pada kunci lain tidak dikolokasikan dengan tabel fakta. Pilih satu dimensi untuk dikolokasikan berdasarkan seberapa sering digabungkan dan ukuran baris penggabungan. Tentukan kunci utama tabel dimensi dan kunci asing yang sesuai tabel fakta sebagai DISTKEY.
2. Pilih dimensi terbesar berdasarkan ukuran dataset yang difilter.

   Hanya baris yang digunakan dalam gabungan yang harus didistribusikan, jadi pertimbangkan ukuran dataset setelah pemfilteran, bukan ukuran tabel.
3. Pilih kolom dengan kardinalitas tinggi dalam kumpulan hasil yang difilter.

   Jika Anda mendistribusikan tabel penjualan pada kolom tanggal, misalnya, Anda mungkin harus mendapatkan distribusi data yang cukup merata, kecuali sebagian besar penjualan Anda bersifat musiman. Namun, jika Anda biasanya menggunakan predikat terbatas rentang untuk memfilter periode tanggal yang sempit, sebagian besar baris yang difilter terjadi pada kumpulan irisan terbatas dan beban kerja kueri miring.
4. Ubah beberapa tabel dimensi untuk menggunakan distribusi SEMUA.

   Jika tabel dimensi tidak dapat dikolokasikan dengan tabel fakta atau tabel penggabungan penting lainnya, Anda dapat meningkatkan kinerja kueri secara signifikan dengan mendistribusikan seluruh tabel ke semua node. Menggunakan distribusi ALL melipatgandakan kebutuhan ruang penyimpanan dan meningkatkan waktu pemuatan dan operasi pemeliharaan, jadi Anda harus mempertimbangkan semua faktor sebelum memilih distribusi SEMUA.

Agar Amazon Redshift memilih gaya distribusi yang sesuai, tent AUT0 ukan gaya distribusi.

Untuk informasi selengkapnya tentang memilih gaya distribusi, lihatDistribusi data untuk optimasi kueri.

Biarkan COPY memilih pengkodean kompresi

Anda dapat menentukan pengkodean kompresi saat membuat tabel, tetapi dalam kebanyakan kasus, kompresi otomatis menghasilkan hasil terbaik.

ENCODE AUTO adalah default untuk tabel. Saat tabel disetel ke ENCODE AUTO, Amazon Redshift secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel. Untuk informasi selengkapnya, lihat CREATE TABLE dan ALTER TABLE.

Perintah COPY menganalisis data Anda dan menerapkan pengkodean kompresi ke tabel kosong secara otomatis sebagai bagian dari operasi pemuatan.

Kompresi otomatis menyeimbangkan kinerja keseluruhan saat memilih pengkodean kompresi. Range-restricted pemindaian mungkin berkinerja buruk jika kolom kunci pengurutan dikompresi jauh lebih tinggi daripada kolom lain dalam kueri yang sama. Akibatnya, kompresi otomatis memilih pengkodean kompresi yang kurang efisien untuk menjaga kolom kunci pengurutan seimbang dengan kolom lain.

Misalkan kunci pengurutan tabel Anda adalah tanggal atau stempel waktu dan tabel menggunakan banyak kolom varchar besar. Dalam hal ini, Anda mungkin mendapatkan kinerja yang lebih baik dengan tidak mengompresi kolom kunci pengurutan sama sekali. Jalan MENGANALISIS KOMPRESI kan perintah pada tabel, lalu gunakan pengkodean untuk membuat tabel baru, tetapi tinggalkan pengkodean kompresi untuk kunci pengurutan.

Ada biaya kinerja untuk pengkodean kompresi otomatis, tetapi hanya jika tabel kosong dan belum memiliki pengkodean kompresi. Untuk tabel dan tabel berumur pendek yang sering Anda buat, seperti tabel pementasan, muat tabel sekali dengan kompresi otomatis atau jalankan perintah ANALYZE COMPRESSION. Kemudian gunakan pengkodean tersebut untuk membuat tabel baru. Anda dapat menambahkan pengkodean ke pernyataan CREATE TABLE, atau gunakan CREATE TABLE LIKE untuk membuat tabel baru dengan encoding yang sama.

Untuk informasi selengkapnya, lihat Memuat tabel dengan kompresi otomatis.

Tentukan kunci primer dan kendala kunci asing

Tentukan kunci primer dan kendala kunci asing antara tabel jika sesuai. Meskipun hanya bersifat informasi, pengoptimal kueri menggunakan batasan tersebut untuk menghasilkan rencana kueri yang lebih efisien.

Jangan mendefinisikan kendala kunci utama dan kunci asing kecuali aplikasi Anda memberlakukan batasan tersebut. Amazon Redshift tidak menerapkan batasan unik, kunci utama, dan kunci asing.

Lihat Kendala tabel untuk informasi tambahan tentang cara Amazon Redshift menggunakan batasan.

Gunakan ukuran kolom sekecil mungkin

Jangan menjadikannya praktik untuk menggunakan ukuran kolom maksimum untuk kenyamanan.

Sebagai gantinya, pertimbangkan nilai terbesar yang mungkin Anda simpan di kolom Anda dan ukurlah sesuai ukuran. Misalnya, kolom CHAR untuk menyimpan U.S. singkatan negara bagian dan wilayah yang digunakan oleh kantor pos hanya perlu CHAR (2).

Gunakan date/time tipe data untuk kolom tanggal

Amazon Redshift menyimpan data DATE dan TIMESTAMP lebih efisien daripada CHAR atau VARCHAR, yang menghasilkan kinerja kueri yang lebih baik. Gunakan tipe data DATE atau TIMESTAMP, tergantung pada resolusi yang Anda butuhkan, bukan tipe karakter saat menyimpan date/time informasi. Untuk informasi selengkapnya, lihat Jenis tanggal-waktu.

Praktik terbaik Amazon Redshift untuk memuat data

Memuat kumpulan data yang sangat besar dapat memakan waktu lama dan menghabiskan banyak sumber daya komputasi. Cara data Anda dimuat juga dapat memengaruhi kinerja kueri. Bagian ini menyajikan praktik terbaik untuk memuat data secara efisien menggunakan perintah COPY, sisipan massal, dan tabel pementasan.

Topik

- Pelajari cara memuat data dengan tutorial
- Gunakan perintah COPY untuk memuat data
- Gunakan satu perintah COPY untuk memuat dari beberapa file
- Memuat file data
- Mengompresi file data Anda
- Verifikasi file data sebelum dan sesudah pemuatan
- Gunakan sisipan multi-baris
- Gunakan sisipan massal

- Muat data dalam urutan kunci
- Muat data dalam blok berurutan
- Gunakan tabel deret waktu
- Jadwalkan sekitar jendela pemeliharaan

Pelajari cara memuat data dengan tutorial

Tutorial: Memuat data dari Amazon S3memandu Anda mulai sampai akhir melalui langkah-langkah untuk mengunggah data ke bucket Amazon S3 dan kemudian menggunakan perintah COPY untuk memuat data ke tabel Anda. Tutorial ini mencakup bantuan dengan pemecahan masalah kesalahan beban dan membandingkan perbedaan kinerja antara memuat dari satu file dan memuat dari beberapa file.

Gunakan perintah COPY untuk memuat data

Perintah COPY memuat data secara paralel dari Amazon S3, Amazon EMR, Amazon DynamoDB, atau beberapa sumber data di host jarak jauh. COPY memuat sejumlah besar data jauh lebih efisien daripada menggunakan pernyataan INSERT, dan menyimpan data dengan lebih efektif juga.

Untuk informasi selengkapnya tentang menggunakan perintah COPY, lihat Memuat data dari Amazon S3 danMemuat data dari tabel Amazon DynamoDB.

Gunakan satu perintah COPY untuk memuat dari beberapa file

Amazon Redshift dapat secara otomatis memuat secara paralel dari beberapa file data terkompresi. Anda dapat menentukan file yang akan dimuat dengan menggunakan awalan objek Amazon S3 atau dengan menggunakan file manifes.

Namun, jika Anda menggunakan beberapa perintah COPY bersamaan untuk memuat satu tabel dari beberapa file, Amazon Redshift dipaksa untuk melakukan pemuatan serial. Jenis beban ini jauh lebih lambat dan membutuhkan proses VACUUM di akhir jika tabel memiliki kolom pengurutan yang ditentukan. Untuk informasi selengkapnya tentang menggunakan COPY untuk memuat data secara paralel, lihatMemuat data dari Amazon S3.

Memuat file data

Source-data file datang dalam format yang berbeda dan menggunakan algoritma kompresi yang bervariasi. Saat memuat data dengan perintah COPY, Amazon Redshift memuat semua file yang

direferensikan oleh awalan bucket Amazon S3. (Awalan adalah string karakter di awal nama kunci objek.) Jika awalan mengacu pada beberapa file atau file yang dapat dibagi, Amazon Redshift memuat data secara paralel, memanfaatkan arsitektur MPP Amazon Redshift. Ini membagi beban kerja di antara node dalam cluster. Sebaliknya, ketika Anda memuat data dari file yang tidak dapat dibagi, Amazon Redshift dipaksa untuk melakukan pemuatan serial, yang jauh lebih lambat. Bagian berikut menjelaskan cara yang disarankan untuk memuat berbagai jenis file ke Amazon Redshift, tergantung pada format dan kompresinya.

Memuat data dari file yang dapat dibagi

File-file berikut dapat secara otomatis dibagi ketika data mereka dimuat:

- file CSV yang tidak terkompresi
- file kolumnar () Parquet/ORC

Amazon Redshift secara otomatis membagi file 128MB atau lebih besar menjadi beberapa bagian. File kolumnar, khususnya Parquet dan ORC, tidak dibagi jika ukurannya kurang dari 128MB. Redshift menggunakan irisan yang bekerja secara paralel untuk memuat data. Ini memberikan kinerja beban yang cepat.

Memuat data dari file yang tidak dapat dibagi

Jenis file seperti JSON, atau CSV, ketika dikompresi dengan algoritma kompresi lain, seperti GZIP, tidak secara otomatis dibagi. Untuk ini, kami sarankan membagi data secara manual menjadi beberapa file yang lebih kecil yang ukurannya dekat, dari 1 MB hingga 1 GB setelah kompresi. Selain itu, buat jumlah file menjadi kelipatan dari jumlah irisan di cluster Anda. Untuk informasi selengkapnya tentang cara membagi data menjadi beberapa file dan contoh pemuatan data menggunakan COPY, lihat Mem uat data dari Amazon S3.

Mengompresi file data Anda

Saat Anda ingin mengompres file beban besar, kami sarankan Anda menggunakan gzip, lzop, bzip2, atau Zstandard untuk mengompresnya dan membagi data menjadi beberapa file yang lebih kecil.

Tentukan opsi GZIP, LZOP, BZIP2, atau ZSTD dengan perintah COPY. Contoh ini memuat tabel TIME dari file lzop yang dibatasi pipa.

```
copy time
```

```
from 's3://amzn-s3-demo-bucket/data/timerows.lzo'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
lzop
delimiter '|';
```

Ada beberapa contoh ketika Anda tidak perlu membagi file data yang tidak terkompresi. Untuk informasi selengkapnya tentang pemisahan data Anda dan contoh penggunaan COPY untuk memuat data, lihatMemuat data dari Amazon S3.

Verifikasi file data sebelum dan sesudah pemuatan

Sebelum memuat data dari Amazon S3, verifikasi terlebih dahulu apakah bucket Amazon S3 berisi semua file yang benar, dan hanya file tersebut. Untuk informasi selengkapnya, lihat Memverifikasi bahwa file yang benar ada di bucket Anda.

Setelah operasi pemuatan selesai, kueri tabel STL_LOAD_COMMIT sistem untuk memverifikasi bahwa file yang diharapkan telah dimuat. Untuk informasi selengkapnya, lihat Memverifikasi bahwa data dimuat dengan benar.

Gunakan sisipan multi-baris

Jika perintah COPY bukan pilihan dan Anda memerlukan sisipan SQL, gunakan sisipan multi-baris bila memungkinkan. Kompresi data tidak efisien ketika Anda menambahkan data hanya satu baris atau beberapa baris sekaligus.

Multi-row sisipan meningkatkan kinerja dengan menggabungkan serangkaian sisipan. Contoh berikut menyisipkan tiga baris ke dalam tabel empat kolom menggunakan pernyataan INSERT tunggal. Ini masih sisipan kecil, ditampilkan hanya untuk menggambarkan sintaks sisipan multi-baris.

```
insert into category_stage values
(default, default, default, default),
(20, default, 'Country', default),
(21, 'Concerts', 'Rock', default);
```

Untuk detail dan contoh lebih lanjut, lihatINSERT.

Gunakan sisipan massal

Gunakan operasi penyisipan massal dengan klausa SELECT untuk penyisipan data berkinerja tinggi.

Gunakan CREATE TABLE AS perintah INSERT dan ketika Anda perlu memindahkan data atau subset data dari satu tabel ke tabel lainnya.

Misalnya, pernyataan INSERT berikut memilih semua baris dari tabel CATEGORY dan menyisipkannya ke dalam tabel CATEGORY_STAGE.

```
insert into category_stage
(select * from category);
```

Contoh berikut membuat CATEGORY_STAGE sebagai salinan CATEGORY dan menyisipkan semua baris dalam CATEGORY ke CATEGORY_STAGE.

```
create table category_stage as
select * from category;
```

Muat data dalam urutan kunci

Muat data Anda dalam urutan kunci agar tidak perlu menyedot debu.

Jika setiap batch data baru mengikuti baris yang ada di tabel Anda dan operasi COPY tidak cukup besar untuk memicu pengoptimalan beban tertentu, data Anda disimpan dengan benar dalam urutan urutan, dan Anda tidak perlu menjalankan vakum. Anda tidak perlu mengurutkan baris di setiap pemuatan karena COPY mengurutkan setiap batch data yang masuk saat dimuat.

Misalnya, Anda memuat data setiap hari berdasarkan aktivitas hari saat ini. Jika kunci pengurutan Anda adalah kolom stempel waktu dan operasi COPY tidak cukup besar untuk memicu pengoptimalan beban tertentu, data Anda disimpan dalam urutan pengurutan. Urutan ini terjadi karena data hari ini selalu ditambahkan pada akhir data hari sebelumnya. Untuk informasi selengkapnya, lihat Muat data Anda dalam urutan kunci. Untuk informasi selengkapnya tentang operasi vakum, lihat tabel meny edot debu.

Muat data dalam blok berurutan

Jika Anda perlu menambahkan sejumlah besar data, muat data dalam blok berurutan sesuai urutan untuk menghilangkan kebutuhan untuk menyedot debu.

Misalnya, Anda perlu memuat tabel dengan acara dari Januari 2017 hingga Desember 2017. Dengan asumsi setiap bulan dalam satu file, muat baris untuk Januari, lalu Februari, dan seterusnya. Tabel

Anda sepenuhnya diurutkan ketika beban Anda selesai, dan Anda tidak perlu menjalankan penyedot debu. Untuk informasi selengkapnya, lihat Gunakan tabel deret waktu.

Saat memuat kumpulan data yang sangat besar, ruang yang diperlukan untuk mengurutkan mungkin melebihi total ruang yang tersedia. Dengan memuat data dalam blok yang lebih kecil, Anda menggunakan ruang pengurutan menengah yang jauh lebih sedikit selama setiap pemuatan. Selain itu, memuat blok yang lebih kecil memudahkan untuk memulai ulang jika COPY gagal dan diputar kembali.

Gunakan tabel deret waktu

Jika data Anda memiliki periode retensi tetap, Anda dapat mengatur data Anda sebagai urutan tabel deret waktu. Dalam urutan seperti itu, setiap tabel identik tetapi berisi data untuk rentang waktu yang berbeda.

Anda dapat dengan mudah menghapus data lama hanya dengan menjalankan perintah DROP TABLE pada tabel yang sesuai. Pendekatan ini jauh lebih cepat daripada menjalankan proses DELETE skala besar dan menyelamatkan Anda dari keharusan menjalankan proses VACUUM berikutnya untuk merebut kembali ruang. Untuk menyembunyikan fakta bahwa data disimpan dalam tabel yang berbeda, Anda dapat membuat tampilan UNION ALL. Saat Anda menghapus data lama, perbaiki tampilan UNION ALL Anda untuk menghapus tabel yang dihapus. Demikian pula, saat Anda memuat periode waktu baru ke tabel baru, tambahkan tabel baru ke tampilan. Untuk memberi sinyal pengoptimal untuk melewati pemindaian pada tabel yang tidak cocok dengan filter kueri, definisi tampilan Anda memfilter rentang tanggal yang sesuai dengan setiap tabel.

Hindari memiliki terlalu banyak tabel di tampilan UNION ALL. Setiap tabel tambahan menambahkan waktu pemrosesan kecil ke kueri. Tabel tidak perlu menggunakan kerangka waktu yang sama. Misalnya, Anda mungkin memiliki tabel untuk periode waktu yang berbeda, seperti harian, bulanan, dan tahunan.

Jika Anda menggunakan tabel deret waktu dengan kolom stempel waktu untuk kunci pengurutan, Anda secara efektif memuat data Anda dalam urutan kunci pengurutan. Melakukan hal ini menghilangkan kebutuhan untuk menyedot debu untuk mengurutkan ulang data. Untuk informasi selengkapnya, lihat Muat data Anda dalam urutan kunci.

Jadwalkan sekitar jendela pemeliharaan

Jika pemeliharaan terjadwal terjadi saat kueri sedang berjalan, kueri dihentikan dan diputar kembali dan Anda perlu memulai ulang. Jadwalkan operasi yang berjalan lama, seperti beban data besar atau operasi VACUUM, untuk menghindari jendela pemeliharaan. Anda juga dapat meminimalkan

risiko, dan membuat restart lebih mudah saat diperlukan, dengan melakukan pemuatan data dengan penambahan yang lebih kecil dan mengelola ukuran operasi VACUUM Anda. Untuk informasi selengkapnya, lihat Muat data dalam blok berurutan dan Meja penyedot debu.

Praktik terbaik Amazon Redshift untuk merancang kueri

Untuk memaksimalkan kinerja kueri, ikuti rekomendasi ini saat membuat kueri:

- Desain tabel sesuai dengan praktik terbaik untuk memberikan dasar yang kuat untuk kinerja kueri. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang tabel.
- Hindari menggunakan`select *`. Sertakan hanya kolom yang Anda butuhkan secara khusus.
- Gunakan a Ekspresi bersyarat CASE untuk melakukan agregasi kompleks alih-alih memilih dari tabel yang sama beberapa kali.
- Jangan gunakan cross-join kecuali benar-benar diperlukan. Gabungan ini tanpa kondisi gabungan menghasilkan produk Cartesian dari dua tabel. Cross-joins biasanya dijalankan sebagai gabungan loop bersarang, yang merupakan jenis gabungan yang paling lambat dari kemungkinan jenis gabungan.
- Gunakan subquery dalam kasus di mana satu tabel dalam kueri digunakan hanya untuk kondisi predikat dan subquery mengembalikan sejumlah kecil baris (kurang dari sekitar 200). Contoh berikut menggunakan subquery untuk menghindari bergabung dengan tabel LISTING.

  ```
  select sum(sales.qtysold)
  from sales
  where salesid in (select listid from listing where listtime > '2008-12-26');
  ```

- Gunakan predikat untuk membatasi dataset sebanyak mungkin.
- Dalam predikat, gunakan operator paling murah yang Anda bisa. Kondisi perbandinganOperator lebih disukai daripada SUKA operator. Operator LIKE masih lebih disukai daripada SIMILAR TO atauOperator POSIX.
- Hindari menggunakan fungsi dalam predikat kueri. Menggunakannya dapat menaikkan biaya kueri dengan membutuhkan sejumlah besar baris untuk menyelesaikan langkah-langkah perantara kueri.
- Jika memungkinkan, gunakan klausa WHERE untuk membatasi dataset. Perencana kueri kemudian dapat menggunakan urutan baris untuk membantu menentukan catatan mana yang cocok dengan kriteria, sehingga dapat melewati pemindaian sejumlah besar blok disk. Tanpa ini, mesin eksekusi kueri harus memindai kolom yang berpartisipasi sepenuhnya.

- Tambahkan predikat ke tabel filter yang berpartisipasi dalam gabungan, bahkan jika predikat menerapkan filter yang sama. Kueri mengembalikan kumpulan hasil yang sama, tetapi Amazon Redshift dapat memfilter tabel gabungan sebelum langkah pemindaian dan kemudian dapat secara efisien melewati blok pemindaian dari tabel tersebut. Filter redundan tidak diperlukan jika Anda memfilter pada kolom yang digunakan dalam kondisi gabungan.

  Misalnya, Anda ingin bergabung SALES dan menemukan penjualan tiket LISTING untuk tiket yang terdaftar setelah Desember, dikelompokkan berdasarkan penjual. Kedua tabel diurutkan berdasarkan tanggal. Kueri berikut menggabungkan tabel pada kunci umum mereka dan menyaring `listing.listtime` nilai yang lebih besar dari 1 Desember.

  ```
  select listing.sellerid, sum(sales.qtysold)
  from sales, listing
  where sales.salesid = listing.listid
  and listing.listtime > '2008-12-01'
  group by 1 order by 1;
  ```

  Klausa WHERE tidak menyertakan predikat untuk`sales.saletime`, sehingga mesin eksekusi dipaksa untuk memindai seluruh SALES tabel. Jika Anda tahu filter akan menghasilkan lebih sedikit baris yang berpartisipasi dalam gabungan, tambahkan filter itu juga. Contoh berikut memotong waktu eksekusi secara signifikan.

  ```
  select listing.sellerid, sum(sales.qtysold)
  from sales, listing
  where sales.salesid = listing.listid
  and listing.listtime > '2008-12-01'
  and sales.saletime > '2008-12-01'
  group by 1 order by 1;
  ```

- Gunakan kunci pengurutan dalam klausa GROUP BY sehingga perencana kueri dapat menggunakan agregasi yang lebih efisien. Kueri mungkin memenuhi syarat untuk agregasi satu fase ketika daftar GROUP BY hanya berisi kolom kunci pengurutan, salah satunya juga kunci distribusi. Kolom kunci pengurutan dalam daftar GROUP BY harus menyertakan kunci pengurutan pertama, kemudian kunci pengurutan lain yang ingin Anda gunakan dalam urutan kunci pengurutan. Misalnya, valid untuk menggunakan kunci pengurutan pertama, kunci pengurutan pertama dan kedua, kunci pengurutan pertama, kedua, dan ketiga, dan seterusnya. Tidak valid untuk menggunakan kunci pengurutan pertama dan ketiga.

Anda dapat mengonfirmasi penggunaan agregasi satu fase dengan menjalankan EXPLAIN perintah dan mencari `XN GroupAggregate` di langkah agregasi kueri.

- Jika Anda menggunakan klausa GROUP BY dan ORDER BY, pastikan Anda menempatkan kolom dalam urutan yang sama di keduanya. Artinya, gunakan pendekatan yang hanya mengikuti.

```
group by a, b, c
order by a, b, c
```

Jangan gunakan pendekatan berikut.

```
group by b, c, a
order by a, b, c
```

Ikuti rekomendasi dari Amazon Redshift Advisor

Untuk membantu Anda meningkatkan kinerja dan mengurangi biaya operasi untuk cluster Amazon Redshift Anda, Amazon Redshift Advisor menawarkan rekomendasi khusus tentang perubahan yang harus dilakukan. Advisor mengembangkan rekomendasi kustomisasi dengan menganalisis metrik kinerja dan penggunaan untuk cluster Anda. Rekomendasi yang disesuaikan ini terkait dengan operasi dan pengaturan cluster. Untuk membantu Anda memprioritaskan pengoptimalan Anda, Advisor memberi peringkat rekomendasi berdasarkan urutan dampak.

Bagaimana Advisor Bekerja

Penasihat mendasarkan rekomendasinya pada pengamatan mengenai statistik kinerja atau data operasi. Advisor mengembangkan pengamatan dengan menjalankan tes pada Anda clusters/workgroups untuk menentukan apakah nilai pengujian berada dalam kisaran tertentu. Jika hasil tes berada di luar kisaran itu, Advisor menghasilkan pengamatan untuk cluster Anda. Pada saat yang sama, Advisor membuat rekomendasi tentang bagaimana mengembalikan nilai yang diamati ke kisaran praktik terbaik.

Untuk arsitektur multi cluster yang menggunakan Amazon Redshift Data Sharing, Advisor kini menyediakan pengoptimalan yang ditingkatkan dengan menganalisis pola beban kerja di semua jaringan data Anda, termasuk clusters/workgroups di clusters/workgroups seluruh Wilayah yang berbeda. Saat Anda berbagi tabel antara produsen dan konsumen clusters/workgroups, Advisor secara otomatis mengumpulkan pola kueri dari semua titik akhir konsumen di mesh data, kecuali

mereka secara eksplisit dicantumkan, dan menggabungkannya dengan beban kerja produsen untuk menghasilkan rekomendasi yang lebih efektif. Ini berarti pengoptimalan tabel Anda—termasuk kunci pengurutan, kunci distribusi, dan kompresi didasarkan pada bagaimana data Anda sebenarnya digunakan di seluruh organisasi Anda, bukan hanya pada satu cluster. Advisor juga mendukung Amazon Redshift Serverless, secara otomatis mempertahankan kontinuitas pengoptimalan di seluruh siklus jeda dan melanjutkan.

Misalnya, data warehouse Anda berisi tabel dengan kunci distribusi suboptimal yang menyebabkan kemiringan data di seluruh node komputasi. Dalam hal ini, Advisor secara otomatis merekomendasikan mendistribusikan ulang tabel menggunakan parameter DISTKEY untuk menentukan kolom yang mendistribusikan data secara merata. Dalam contoh lain, misalkan Advisor mengamati bahwa cluster Anda memiliki tabel tanpa kunci pengurutan atau dengan definisi kunci pengurutan yang tidak efisien yang mengakibatkan kinerja kueri yang buruk. Dalam hal ini, Advisor secara otomatis memberikan rekomendasi untuk kolom kunci pengurutan yang sesuai berdasarkan pola kueri Anda untuk meningkatkan penyaringan data dan mengurangi disk I/O.

Mengoptimalkan Arsitektur Berbagi Data

Saat Anda menggunakan Amazon Redshift Data Sharing untuk mendistribusikan beban kerja ke beberapa beban kerja clusters/workgroups, Advisor membantu Anda mengoptimalkan kinerja di seluruh jaring data Anda. Advisor secara otomatis menganalisis bagaimana tabel bersama ditanyakan di semua konsumen. clusters/workgroups Ini termasuk memahami kolom mana yang sering difilter, tabel mana yang biasanya digabungkan bersama, dan bagaimana data dipindai. Dengan mempertimbangkan gambaran lengkap tentang penggunaan data, Advisor menghasilkan rekomendasi yang meningkatkan kinerja untuk semua pengguna data bersama Anda.

Dengan mengoptimalkan tabel berdasarkan pola penggunaan di seluruh organisasi Anda daripada satu cluster, Anda dapat:

- Buat keputusan pengoptimalan berbasis data berdasarkan pola akses data di semua clusters/workgroups dalam mesh
- Menurunkan biaya penyimpanan melalui strategi kompresi yang lebih efektif
- Meningkatkan pemanfaatan sumber daya di seluruh jaring data Anda

Wilayah Amazon Redshift tempat Penasihat didukung

Fitur Amazon Redshift Advisor hanya tersedia di Wil AWS ayah berikut:

- Wilayah AS Timur (Virginia Utara) (us-timur-1)
- Wilayah AS Timur (Ohio) (us-east-2)
- Wilayah AS Barat (California Utara) (us-barat-1)
- Wilayah AS Barat (Oregon) (us-barat-2)
- Wilayah Afrika (Cape Town) (af-south-1)
- Wilayah Asia Pasifik (Hong Kong) (ap-east-1)
- Wilayah Asia Pasifik (Hyderabad) (ap-south-2)
- Wilayah Asia Pasifik (Jakarta) (ap-southeast-3)
- Wilayah Asia Pasifik (Melbourne) (ap-southeast-4)
- Wilayah Asia Pasifik (Malaysia) (ap-southeast-5)
- Wilayah Asia Pasifik (Mumbai) (ap-south-1)
- Wilayah Asia Pasifik (Osaka) (ap-northeast-3)
- Wilayah Asia Pasifik (Seoul) (ap-northeast-2)
- Wilayah Asia Pasifik (Singapura) (ap-southeast-1)
- Wilayah Asia Pasifik (Sydney) (ap-southeast-2)
- Wilayah Asia Pasifik (Tokyo) (ap-northeast-1)
- Wilayah Kanada (Tengah) (ca-central-1)
- Wilayah Kanada Barat (Calgary) (ca-west-1)
- Wilayah China (Beijing) (cn-north-1)
- Wilayah China (Ningxia) (cn-northwest-1)
- Wilayah Eropa (Frankfurt) (eu-central-1)
- Wilayah Eropa (Irlandia) (eu-west-1)
- Wilayah Eropa (London) (eu-west-2)
- Wilayah Eropa (Milan) (eu-south-1)
- Wilayah Eropa (Paris) (eu-west-3)
- Eropa (Spanyol) Wilayah (eu-south-2)
- Wilayah Eropa (Stockholm) (eu-utara-1)
- Wilayah Eropa (Zurich) (eu-central-2)
- Wilayah Israel (Tel Aviv) (il-sentral-1)

- Wilayah Timur Tengah (Bahrain) (me-south-1)
- Wilayah Timur Tengah (UEA) (me-central-1)
- Wilayah Amerika Selatan (São Paulo) (sa-east-1)

Topik

- Melihat rekomendasi Amazon Redshift Advisor
- Rekomendasi Amazon Redshift Advisor

Melihat rekomendasi Amazon Redshift Advisor

Anda dapat mengakses rekomendasi Amazon Redshift Advisor menggunakan konsol Amazon Redshift, Amazon Redshift API, atau. AWS CLI Untuk mengakses rekomendasi, Anda harus memiliki izin yang dil `redshift:ListRecommendations` ampirkan ke peran atau identitas IAM Anda.

Melihat rekomendasi Amazon Redshift Advisor di konsol yang disediakan Amazon Redshift

Anda dapat melihat rekomendasi Amazon Redshift Advisor di Konsol Manajemen AWS.

Untuk melihat rekomendasi Amazon Redshift Advisor untuk cluster Amazon Redshift di konsol

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Advisor.
3. Perluas setiap rekomendasi untuk melihat detail lebih lanjut. Di halaman ini, Anda dapat mengurutkan dan mengelompokkan rekomendasi.

Melihat rekomendasi Amazon Redshift Advisor menggunakan operasi Amazon Redshift API

Anda dapat mencantumkan rekomendasi Amazon Redshift Advisor untuk cluster Amazon Redshift menggunakan Amazon Redshift API. Biasanya, Anda mengembangkan dan menerapkan dalam bahasa pemrograman pilihan Anda untuk memang `redshift:ListRecommendations` gil API menggunakan AWS SDK. Untuk informasi selengkapnya, lihat ListRecommendations Refer ensi API Amazon Redshift.

Melihat rekomendasi Amazon Redshift Advisor menggunakan AWS Command Line Interface operasi

Anda dapat membuat daftar rekomendasi Amazon Redshift Advisor untuk cluster Amazon Redshift menggunakan. AWS Command Line Interface Untuk informasi selengkapnya, lihat daftar rekomendasi di Referensi Per AWS CLI intah.

Rekomendasi Amazon Redshift Advisor

Amazon Redshift Advisor menawarkan rekomendasi tentang cara mengoptimalkan cluster Amazon Redshift Anda untuk meningkatkan kinerja dan menghemat biaya pengoperasian. Anda dapat menemukan penjelasan untuk setiap rekomendasi di konsol, seperti yang dijelaskan sebelumnya. Anda dapat menemukan detail lebih lanjut tentang rekomendasi ini di bagian berikut.

Topik

- Kompres objek file Amazon S3 yang dimuat oleh COPY
- Isolasi beberapa database aktif
- Mengalokasikan kembali memori manajemen beban kerja (WLM)
- Lewati analisis kompresi selama COPY
- Pisahkan objek Amazon S3 yang dimuat oleh COPY
- Perbarui statistik tabel
- Aktifkan akselerasi kueri singkat
- Mengubah kunci distribusi pada tabel
- Ubah kunci pengurutan pada tabel
- Ubah pengkodean kompresi pada kolom
- Rekomendasi tipe data

Kompres objek file Amazon S3 yang dimuat oleh COPY

Perintah COPY memanfaatkan arsitektur pemrosesan paralel massif (MPP) di Amazon Redshift untuk membaca dan memuat data secara paralel. Ini dapat membaca file dari Amazon S3, tabel DynamoDB, dan output teks dari satu atau lebih host jarak jauh.

Saat memuat data dalam jumlah besar, kami sangat menyarankan menggunakan perintah COPY untuk memuat file data terkompresi dari S3. Mengompresi kumpulan data besar menghemat

waktu mengunggah file ke Amazon S3. COPY juga dapat mempercepat proses pemuatan dengan membuka kompresi file saat dibaca.

Analisis

Long-running Perintah COPY yang memuat kumpulan data besar yang tidak terkompresi sering kali memiliki peluang untuk peningkatan kinerja yang cukup besar. Analisis Advisor mengidentifikasi perintah COPY yang memuat kumpulan data besar yang tidak terkompresi. Dalam kasus seperti itu, Advisor menghasilkan rekomendasi untuk menerapkan kompresi pada file sumber di Amazon S3.

Rekomendasi

Pastikan bahwa setiap COPY yang memuat sejumlah besar data, atau berjalan untuk durasi yang signifikan, menyerap objek data terkompresi dari Amazon S3. Anda dapat mengidentifikasi perintah COPY yang memuat kumpulan data besar yang tidak terkompresi dari Amazon S3 dengan menjalankan perintah SQL berikut sebagai superuser.

```
SELECT
    wq.userid, query, exec_start_time AS starttime, COUNT(*) num_files,
    ROUND(MAX(wq.total_exec_time/1000000.0),2) execution_secs,
    ROUND(SUM(transfer_size)/(1024.0*1024.0),2) total_mb,
    SUBSTRING(querytxt,1,60) copy_sql
FROM stl_s3client s
JOIN stl_query q USING (query)
JOIN stl_wlm_query wq USING (query)
WHERE s.userid>1 AND http_method = 'GET'
    AND POSITION('COPY ANALYZE' IN querytxt) = 0
    AND aborted = 0 AND final_state='Completed'
GROUP BY 1, 2, 3, 7
HAVING SUM(transfer_size) = SUM(data_size)
AND SUM(transfer_size)/(1024*1024) >= 5
ORDER BY 6 DESC, 5 DESC;
```

Jika data bertahap tetap ada di Amazon S3 setelah Anda memuatnya, yang biasa terjadi pada arsitektur data lake, menyimpan data ini dalam bentuk terkompresi dapat mengurangi biaya penyimpanan Anda.

Kiat implementasi

- Ukuran objek ideal adalah 1-128 MB setelah kompresi.

- Anda dapat mengompres file dengan format gzip, lzop, atau bzip2.

Isolasi beberapa database aktif

Sebagai praktik terbaik, kami sarankan untuk mengisolasi database di Amazon Redshift satu sama lain. Kueri berjalan di database tertentu dan tidak dapat mengakses data dari database lain di cluster. Namun, kueri yang Anda jalankan di semua database cluster berbagi ruang penyimpanan cluster yang mendasari dan sumber daya komputasi yang sama. Ketika satu cluster berisi beberapa database aktif, beban kerja mereka biasanya tidak terkait.

Analisis

Analisis Advisor meninjau semua database di cluster untuk beban kerja aktif yang berjalan pada saat yang sama. Jika ada beban kerja aktif yang berjalan pada saat yang sama, Advisor membuat rekomendasi untuk mempertimbangkan migrasi database ke cluster Amazon Redshift yang terpisah.

Rekomendasi

Pertimbangkan untuk memindahkan setiap database yang ditanyakan secara aktif ke cluster khusus yang terpisah. Menggunakan cluster terpisah dapat mengurangi perselisihan sumber daya dan meningkatkan kinerja kueri. Hal ini dapat dilakukan karena Anda dapat mengatur ukuran untuk setiap cluster untuk kebutuhan penyimpanan, biaya, dan kinerja setiap beban kerja. Selain itu, beban kerja yang tidak terkait sering mendapat manfaat dari konfigurasi manajemen beban kerja yang berbeda.

Untuk mengidentifikasi database mana yang digunakan secara aktif, Anda dapat menjalankan perintah SQL ini sebagai superuser.

```
SELECT database,
  COUNT(*) as num_queries,
  AVG(DATEDIFF(sec,starttime,endtime)) avg_duration,
  MIN(starttime) as oldest_ts,
  MAX(endtime) as latest_ts
FROM stl_query
WHERE userid > 1
GROUP BY database;
```

Kiat implementasi

- Karena pengguna harus terhubung ke setiap database secara khusus, dan kueri hanya dapat mengakses satu database, memindahkan database ke cluster terpisah memiliki dampak minimal bagi pengguna.
- Salah satu opsi untuk memindahkan database adalah dengan mengambil langkah-langkah berikut:
  1. Kembalikan sementara snapshot dari cluster saat ini ke cluster dengan ukuran yang sama.
  2. Hapus semua database dari cluster baru kecuali database target yang akan dipindahkan.
  3. Ubah ukuran cluster ke jenis node yang sesuai dan hitung untuk beban kerja database.

Mengalokasikan kembali memori manajemen beban kerja (WLM)

Amazon Redshift merutekan kueri pengguna ke Menerapkan manual WLM untuk diproses. Manajemen beban kerja (WLM) mendefinisikan bagaimana kueri tersebut dialihkan ke antrian. Amazon Redshift mengalokasikan setiap antrian sebagian dari memori cluster yang tersedia. Memori antrian dibagi di antara slot query antrian.

Ketika antrian dikonfigurasi dengan lebih banyak slot daripada yang dibutuhkan beban kerja, memori yang dialokasikan untuk slot yang tidak digunakan ini kurang dimanfaatkan. Mengurangi slot yang dikonfigurasi agar sesuai dengan persyaratan beban kerja puncak mendistribusikan kembali memori yang kurang dimanfaatkan ke slot aktif, dan dapat menghasilkan peningkatan kinerja kueri.

Analisis

Analisis Advisor meninjau persyaratan konkurensi beban kerja untuk mengidentifikasi antrian kueri dengan slot yang tidak digunakan. Penasihat menghasilkan rekomendasi untuk mengurangi jumlah slot dalam antrian ketika menemukan yang berikut:

- Antrian dengan slot yang sama sekali tidak aktif selama analisis.
- Antrian dengan lebih dari empat slot yang memiliki setidaknya dua slot tidak aktif selama analisis.

Rekomendasi

Mengurangi slot yang dikonfigurasi agar sesuai dengan persyaratan beban kerja puncak mendistribusikan kembali memori yang kurang dimanfaatkan ke slot aktif. Pertimbangkan untuk mengurangi jumlah slot yang dikonfigurasi untuk antrian di mana slot belum pernah sepenuhnya digunakan. Untuk mengidentifikasi antrian ini, Anda dapat membandingkan persyaratan slot puncak per jam untuk setiap antrian dengan menjalankan perintah SQL berikut sebagai superuser.

```
WITH
 generate_dt_series AS (select sysdate - (n * interval '5 second') as dt from (select
 row_number() over () as n from stl_scan limit 17280)),
 apex AS (
     SELECT iq.dt, iq.service_class, iq.num_query_tasks, count(iq.slot_count) as
 service_class_queries, sum(iq.slot_count) as service_class_slots
     FROM
         (select gds.dt, wq.service_class, wscc.num_query_tasks, wq.slot_count
         FROM stl_wlm_query wq
         JOIN stv_wlm_service_class_config wscc ON (wscc.service_class =
 wq.service_class AND wscc.service_class > 5)
         JOIN generate_dt_series gds ON (wq.service_class_start_time <= gds.dt AND
 wq.service_class_end_time > gds.dt)
         WHERE wq.userid > 1 AND wq.service_class > 5) iq
     GROUP BY iq.dt, iq.service_class, iq.num_query_tasks),
     maxes as (SELECT apex.service_class, trunc(apex.dt) as d, date_part(h,apex.dt) as
 dt_h, max(service_class_slots) max_service_class_slots
                     from apex group by apex.service_class, apex.dt,
 date_part(h,apex.dt))
SELECT apex.service_class - 5 AS queue, apex.service_class, apex.num_query_tasks AS
 max_wlm_concurrency, maxes.d AS day, maxes.dt_h || ':00 - ' || maxes.dt_h || ':59' as
 hour, MAX(apex.service_class_slots) as max_service_class_slots
FROM apex
JOIN maxes ON (apex.service_class = maxes.service_class AND apex.service_class_slots =
 maxes.max_service_class_slots)
GROUP BY  apex.service_class, apex.num_query_tasks, maxes.d, maxes.dt_h
ORDER BY apex.service_class, maxes.d, maxes.dt_h;
```

`max_service_class_slots`Kolom mewakili jumlah maksimum slot kueri WLM dalam antrian kueri untuk jam tersebut. Jika ada antrian yang kurang dimanfaatkan, terapkan pengoptimalan pengurangan slot dengan memodifikasi grup parameter, seperti yang dijelaskan dalam Panduan Manajemen Amazon Redshift.

Kiat implementasi

- Jika volume beban kerja Anda sangat bervariasi, pastikan analisis menangkap periode pemanfaatan puncak. Jika tidak, jalankan SQL sebelumnya berulang kali untuk memantau persyaratan konkurensi puncak.
- Untuk detail selengkapnya tentang menafsirkan hasil kueri dari kode SQL sebelumnya, lihat skrip wlm_apex_hourly.sql pada GitHub.

Lewati analisis kompresi selama COPY

Saat Anda memuat data ke tabel kosong dengan pengkodean kompresi yang dideklarasikan dengan perintah COPY, Amazon Redshift menerapkan kompresi penyimpanan. Pengoptimalan ini memastikan bahwa data di cluster Anda disimpan secara efisien bahkan ketika dimuat oleh pengguna akhir. Analisis yang diperlukan untuk menerapkan kompresi dapat membutuhkan waktu yang signifikan.

Analisis

Analisis Advisor memeriksa operasi COPY yang tertunda oleh analisis kompresi otomatis. Analisis menentukan pengkodean kompresi dengan mengambil sampel data saat sedang dimuat. Pengambilan sampel ini mirip dengan yang dilakukan oleh MENGANALISIS KOMPRESI perintah.

Ketika Anda memuat data sebagai bagian dari proses terstruktur, seperti dalam batch ekstrak semalam, transform, load (ETL), Anda dapat menentukan kompresi sebelumnya. Anda juga dapat mengoptimalkan definisi tabel Anda untuk melewati fase ini secara permanen tanpa dampak negatif.

Rekomendasi

Untuk meningkatkan responsivitas COPY dengan melewatkan fase analisis kompresi, terapkan salah satu dari dua opsi berikut:

- Gunakan `ENCODE` parameter kolom saat membuat tabel apa pun yang Anda muat menggunakan perintah COPY.
- Matikan kompresi sama sekali dengan menyediakan `COMPUPDATE OFF` parameter dalam perintah COPY.

Solusi terbaik umumnya menggunakan pengkodean kolom selama pembuatan tabel, karena pendekatan ini juga mempertahankan manfaat menyimpan data terkompresi pada disk. Anda dapat menggunakan perintah ANALYZE COMPRESSION untuk menyarankan pengkodean kompresi, tetapi Anda harus membuat ulang tabel untuk menerapkan pengkodean ini. Untuk mengotomatiskan proses ini, Anda dapat menggunakan AWS ColumnEncodingUtility, ditemukan di GitHub.

Untuk mengidentifikasi operasi COPY terbaru yang memicu analisis kompresi otomatis, jalankan perintah SQL berikut.

```
  WITH xids AS (
    SELECT xid FROM stl_query WHERE userid>1 AND aborted=0
    AND querytxt = 'analyze compression phase 1' GROUP BY xid
    INTERSECT SELECT xid FROM stl_commit_stats WHERE node=-1)
SELECT a.userid, a.query, a.xid, a.starttime, b.complyze_sec,
    a.copy_sec, a.copy_sql
FROM (SELECT q.userid, q.query, q.xid, date_trunc('s',q.starttime)
    starttime, substring(querytxt,1,100) as copy_sql,
    ROUND(datediff(ms,starttime,endtime)::numeric / 1000.0, 2) copy_sec
    FROM stl_query q JOIN xids USING (xid)
    WHERE (querytxt ilike 'copy %from%' OR querytxt ilike '% copy %from%')
    AND querytxt not like 'COPY ANALYZE %') a
LEFT JOIN (SELECT xid,
    ROUND(sum(datediff(ms,starttime,endtime))::numeric / 1000.0,2) complyze_sec
    FROM stl_query q JOIN xids USING (xid)
    WHERE (querytxt like 'COPY ANALYZE %'
    OR querytxt like 'analyze compression phase %')
    GROUP BY xid ) b ON a.xid = b.xid
WHERE b.complyze_sec IS NOT NULL ORDER BY a.copy_sql, a.starttime;
```

Kiat implementasi

- Pastikan bahwa semua tabel dengan ukuran signifikan yang dibuat selama proses ETL Anda (misalnya, tabel pementasan dan tabel sementara) mendeklarasikan pengkodean kompresi untuk semua kolom kecuali kunci pengurutan pertama.
- Perkirakan ukuran masa pakai yang diharapkan dari tabel yang dimuat untuk setiap perintah COPY yang diidentifikasi oleh perintah SQL sebelumnya. Jika Anda yakin bahwa tabel akan tetap sangat kecil, matikan kompresi sama sekali dengan `COMPUPDATE OFF` parameter. Jika tidak, buat tabel dengan kompresi eksplisit sebelum memuatnya dengan perintah COPY.

Pisahkan objek Amazon S3 yang dimuat oleh COPY

Perintah COPY memanfaatkan arsitektur pemrosesan paralel massif (MPP) di Amazon Redshift untuk membaca dan memuat data dari file di Amazon S3. Perintah COPY memuat data secara paralel dari beberapa file, membagi beban kerja di antara node di cluster Anda. Untuk mencapai throughput yang optimal, kami sangat menyarankan Anda membagi data menjadi beberapa file untuk memanfaatkan pemrosesan paralel.

Analisis

Analisis Advisor mengidentifikasi perintah COPY yang memuat kumpulan data besar yang terkandung dalam sejumlah kecil file yang dipentaskan di Amazon S3. Long-running Perintah COPY yang memuat kumpulan data besar dari beberapa file sering memiliki peluang untuk peningkatan kinerja yang cukup besar. Ketika Advisor mengidentifikasi bahwa perintah COPY ini membutuhkan waktu yang signifikan, itu membuat rekomendasi untuk meningkatkan paralelisme dengan membagi data menjadi file tambahan di Amazon S3.

Rekomendasi

Dalam hal ini, kami merekomendasikan tindakan berikut, tercantum dalam urutan prioritas:

1. Optimalkan perintah COPY yang memuat lebih sedikit file daripada jumlah node cluster.
2. Optimalkan perintah COPY yang memuat lebih sedikit file daripada jumlah irisan cluster.
3. Optimalkan perintah COPY di mana jumlah file tidak kelipatan dari jumlah irisan cluster.

Perintah COPY tertentu memuat sejumlah besar data atau dijalankan untuk durasi yang signifikan. Untuk perintah ini, sebaiknya Anda memuat sejumlah objek data dari Amazon S3 yang setara dengan kelipatan jumlah irisan dalam cluster. Untuk mengidentifikasi berapa banyak objek S3 yang telah dimuat setiap perintah COPY, jalankan kode SQL berikut sebagai superuser.

```
SELECT
    query, COUNT(*) num_files,
    ROUND(MAX(wq.total_exec_time/1000000.0),2) execution_secs,
    ROUND(SUM(transfer_size)/(1024.0*1024.0),2) total_mb,
    SUBSTRING(querytxt,1,60) copy_sql
FROM stl_s3client s
JOIN stl_query q USING (query)
JOIN stl_wlm_query wq USING (query)
WHERE s.userid>1 AND http_method = 'GET'
    AND POSITION('COPY ANALYZE' IN querytxt) = 0
    AND aborted = 0 AND final_state='Completed'
GROUP BY query, querytxt
HAVING (SUM(transfer_size)/(1024*1024))/COUNT(*) >= 2
ORDER BY CASE
WHEN COUNT(*) < (SELECT max(node)+1 FROM stv_slices) THEN 1
WHEN COUNT(*) < (SELECT COUNT(*) FROM stv_slices WHERE node=0) THEN 2
ELSE 2+((COUNT(*) % (SELECT COUNT(*) FROM stv_slices))/(SELECT COUNT(*)::DECIMAL FROM
 stv_slices))
END, (SUM(transfer_size)/(1024.0*1024.0))/COUNT(*) DESC;
```

Kiat implementasi

- Jumlah irisan dalam node tergantung pada ukuran node cluster. Untuk informasi selengkapnya tentang jumlah irisan di berbagai jenis node, lihat Cluster dan Node di Amazon Redshift di Panduan Manajemen Amazon Redshift.
- Anda dapat memuat beberapa file dengan menentukan awalan umum, atau kunci awalan, untuk set, atau dengan mencantumkan file secara eksplisit dalam file manifes. Untuk informasi selengkapnya tentang memuat file, lihatMemuat data dari file terkompresi dan tidak terkompresi.
- Amazon Redshift tidak memperhitungkan ukuran file saat membagi beban kerja. Pisahkan file data muat Anda sehingga file berukuran hampir sama, antara 1 MB dan 1 GB setelah kompresi.

Perbarui statistik tabel

Amazon Redshift menggunakan pengoptimal kueri berbasis biaya untuk memilih rencana eksekusi optimal untuk kueri. Perkiraan biaya didasarkan pada statistik tabel yang dikumpulkan menggunakan perintah ANALYZE. Ketika statistik sudah ketinggalan zaman atau hilang, database mungkin memilih rencana yang kurang efisien untuk eksekusi kueri, terutama untuk kueri kompleks. Mempertahankan statistik saat ini membantu kueri kompleks berjalan dalam waktu sesingkat mungkin.

Analisis

Analisis Advisor melacak tabel yang statistiknya kedaluwarsa atau hilang. Ini meninjau metadata akses tabel yang terkait dengan kueri kompleks. Jika tabel yang sering diakses dengan pola kompleks tidak memiliki statistik, Advisor membuat rekomendasi penting untuk menjalankan ANALYZE. Jika tabel yang sering diakses dengan pola kompleks memiliki statistik yang ketinggalan zaman, Advisor membuat rekomendasi yang disarankan untuk menjalankan ANALYZE.

Rekomendasi

Setiap kali konten tabel berubah secara signifikan, perbarui statistik dengan ANALYZE. Sebaiknya jalankan ANALYZE setiap kali sejumlah besar baris data baru dimuat ke tabel yang ada dengan perintah COPY atau INSERT. Kami juga merekomendasikan menjalankan ANALYZE setiap kali sejumlah besar baris dimodifikasi menggunakan perintah UPDATE atau DELETE. Untuk mengidentifikasi tabel dengan statistik yang hilang atau kedaluwarsa, jalankan perintah SQL berikut sebagai superuser. Hasilnya diurutkan dari tabel terbesar ke terkecil.

Untuk mengidentifikasi tabel dengan statistik yang hilang atau kedaluwarsa, jalankan perintah SQL berikut sebagai superuser. Hasilnya diurutkan dari tabel terbesar ke terkecil.

```
SELECT
   ti.schema||'.'||ti."table" tablename,
   ti.size table_size_mb,
   ti.stats_off statistics_accuracy
 FROM svv_table_info ti
 WHERE ti.stats_off > 5.00
 ORDER BY ti.size DESC;
```

Kiat implementasi

Ambang batas ANALISIS default adalah 10 persen. Default ini berarti bahwa perintah ANALYZE melewatkan tabel yang diberikan jika kurang dari 10 persen baris tabel telah berubah sejak ANALYZE terakhir. Akibatnya, Anda dapat memilih untuk mengeluarkan perintah ANALYZE di akhir setiap proses ETL. Mengambil pendekatan ini berarti bahwa ANALYZE sering dilewati tetapi juga memastikan bahwa ANALYZE berjalan saat diperlukan.

Statistik ANALISIS memiliki dampak paling besar untuk kolom yang digunakan dalam gabungan (misalnya,`JOIN tbl_a ON col_b`) atau sebagai predikat (misalnya,`WHERE col_b = 'xyz'`). Secara default, ANALYZE mengumpulkan statistik untuk semua kolom dalam tabel yang ditentukan. Jika diperlukan, Anda dapat mengurangi waktu yang diperlukan untuk menjalankan ANALYZE dengan menjalankan ANALYZE hanya untuk kolom yang memiliki dampak paling besar. Anda dapat menjalankan perintah SQL berikut untuk mengidentifikasi kolom yang digunakan sebagai predikat. Anda juga dapat membiarkan Amazon Redshift memilih kolom mana yang akan dianalisis dengan menentukan`ANALYZE PREDICATE COLUMNS`.

```
WITH predicate_column_info as (
SELECT ns.nspname AS schema_name, c.relname AS table_name, a.attnum as col_num,
 a.attname as col_name,
        CASE
            WHEN 10002 = s.stakind1 THEN array_to_string(stavalues1, '||')
            WHEN 10002 = s.stakind2 THEN array_to_string(stavalues2, '||')
            WHEN 10002 = s.stakind3 THEN array_to_string(stavalues3, '||')
            WHEN 10002 = s.stakind4 THEN array_to_string(stavalues4, '||')
            ELSE NULL::varchar
        END AS pred_ts
   FROM pg_statistic s
```

```
   JOIN pg_class c ON c.oid = s.starelid
   JOIN pg_namespace ns ON c.relnamespace = ns.oid
   JOIN pg_attribute a ON c.oid = a.attrelid AND a.attnum = s.staattnum)
SELECT schema_name, table_name, col_num, col_name,
       pred_ts NOT LIKE '2000-01-01%' AS is_predicate,
       CASE WHEN pred_ts NOT LIKE '2000-01-01%' THEN (split_part(pred_ts,
 '||',1))::timestamp ELSE NULL::timestamp END as first_predicate_use,
       CASE WHEN pred_ts NOT LIKE '%||2000-01-01%' THEN (split_part(pred_ts,
 '||',2))::timestamp ELSE NULL::timestamp END as last_analyze
FROM predicate_column_info;
```

Untuk informasi selengkapnya, lihat Menganalisis tabel.

Aktifkan akselerasi kueri singkat

Akselerasi kueri pendek (SQA) memprioritaskan kueri berjalan pendek yang dipilih di depan kueri yang berjalan lebih lama. SQA menjalankan kueri yang berjalan singkat di ruang khusus, sehingga kueri SQA tidak dipaksa menunggu dalam antrian di belakang kueri yang lebih lama. SQA hanya memprioritaskan kueri yang berjalan singkat dan berada dalam antrian yang ditentukan pengguna. Dengan SQA, kueri yang berjalan pendek mulai berjalan lebih cepat dan pengguna melihat hasilnya lebih cepat.

Jika Anda mengaktifkan SQA, Anda dapat mengurangi atau menghilangkan antrian manajemen beban kerja (WLM) yang didedikasikan untuk menjalankan kueri pendek. Selain itu, kueri yang berjalan lama tidak perlu bersaing dengan kueri pendek untuk slot dalam antrian, sehingga Anda dapat mengonfigurasi antrian WLM Anda untuk menggunakan slot kueri yang lebih sedikit. Bila Anda menggunakan konkurensi yang lebih rendah, throughput kueri meningkat dan kinerja sistem secara keseluruhan ditingkatkan untuk sebagian besar beban kerja. Untuk informasi selengkapnya, lihat Akselerasi kueri singkat.

Analisis

Penasihat memeriksa pola beban kerja dan melaporkan jumlah kueri terbaru di mana SQA akan mengurangi latensi dan waktu antrian harian untuk SQA-eligible kueri.

Rekomendasi

Ubah konfigurasi WLM untuk mengaktifkan SQA. Amazon Redshift menggunakan algoritma pembelajaran mesin untuk menganalisis setiap kueri yang memenuhi syarat. Prediksi meningkat saat SQA belajar dari pola kueri Anda. Untuk informasi selengkapnya, lihat Mengon figurasi Manajemen Beban Kerja.

Saat Anda mengaktifkan SQA, WLM menetapkan runtime maksimum untuk kueri pendek menjadi dinamis secara default. Sebaiknya simpan pengaturan dinamis untuk runtime maksimum SQA.

Kiat implementasi

Untuk memeriksa apakah SQA dihidupkan, jalankan kueri berikut. Jika kueri mengembalikan baris, maka SQA dihidupkan.

```
select * from stv_wlm_service_class_config
where service_class = 14;
```

Untuk informasi selengkapnya, lihat Pemantauan SQA.

Mengubah kunci distribusi pada tabel

Amazon Redshift mendistribusikan baris tabel di seluruh cluster sesuai dengan gaya distribusi tabel. Tabel dengan distribusi KEY memerlukan kolom sebagai kunci distribusi (DISTKEY). Baris tabel ditetapkan ke irisan simpul cluster berdasarkan nilai kolom DISTKEY.

DISTKEY yang sesuai menempatkan jumlah baris yang sama pada setiap irisan simpul dan sering direferensikan dalam kondisi gabungan. Gabungan yang dioptimalkan terjadi ketika tabel digabungkan pada kolom DISTKEY mereka, mempercepat kinerja kueri.

Analisis

Advisor menganalisis beban kerja cluster Anda untuk mengidentifikasi kunci distribusi yang paling tepat untuk tabel yang dapat memperoleh manfaat signifikan dari gaya distribusi KEY.

Rekomendasi

Advisor memberikan ALTER TABLE pernyataan yang mengubah DISTSTYLE dan DISTKEY dari tabel berdasarkan analisisnya. Untuk mewujudkan manfaat kinerja yang signifikan, pastikan untuk menerapkan semua pernyataan SQL dalam grup rekomendasi.

Mendistribusikan ulang tabel besar dengan ALTER TABLE menghabiskan sumber daya cluster dan memerlukan kunci tabel sementara di berbagai waktu. Terapkan setiap grup rekomendasi saat beban kerja cluster lainnya ringan. Untuk detail selengkapnya tentang mengoptimalkan properti distribusi tabel, lihat Pedoman Desain Tabel Tingkat Lanjut Amazon Redshift Engineering: Gaya Distribusi dan Kunci Distribusi.

Untuk informasi lebih lanjut tentang ALTER DISTSYLE dan DISTKEY, lihat. ALTER TABLE

> **Note**
>
> Jika Anda tidak melihat rekomendasi, itu tidak berarti bahwa gaya distribusi saat ini adalah yang paling tepat. Penasihat tidak memberikan rekomendasi ketika tidak ada cukup data atau manfaat yang diharapkan dari redistribusi kecil.
> Rekomendasi penasihat berlaku untuk tabel tertentu dan tidak selalu berlaku untuk tabel yang berisi kolom dengan nama yang sama. Tabel yang berbagi nama kolom dapat memiliki karakteristik yang berbeda untuk kolom tersebut kecuali data di dalam tabel sama.
> Jika Anda melihat rekomendasi untuk tabel pementasan yang dibuat atau dihapus oleh pekerjaan ETL, ubah proses ETL Anda untuk menggunakan kunci distribusi yang direkomendasikan Penasihat.

Ubah kunci pengurutan pada tabel

Amazon Redshift mengurutkan baris tabel sesuai dengan kunci pengurutan tabel. Penyortiran baris tabel didasarkan pada nilai kolom kunci pengurutan.

Menyortir tabel pada kunci pengurutan yang sesuai dapat mempercepat kinerja kueri, terutama yang memiliki predikat terbatas rentang, dengan membutuhkan lebih sedikit blok tabel untuk dibaca dari disk.

Analisis

Advisor menganalisis beban kerja cluster Anda selama beberapa hari untuk mengidentifikasi kunci pengurutan yang bermanfaat untuk tabel Anda.

Rekomendasi

Advisor menyediakan dua kelompok pernyataan ALTER TABLE yang mengubah kunci pengurutan tabel berdasarkan analisisnya:

- Pernyataan yang mengubah tabel yang saat ini tidak memiliki kunci pengurutan untuk menambahkan kunci pengurutan COMPOUND.
- Pernyataan yang mengubah kunci pengurutan dari INTERLEAVED menjadi COMPOUND atau tanpa kunci pengurutan.

  Menggunakan kunci pengurutan majemuk secara signifikan mengurangi biaya perawatan. Tabel dengan kunci pengurutan majemuk tidak memerlukan operasi VACUUM REINDEX mahal yang diperlukan untuk pengurutan yang disisipkan. Dalam praktiknya, kunci pengurutan majemuk lebih

efektif daripada kunci pengurutan yang disisipkan untuk sebagian besar beban kerja Amazon Redshift. Namun, jika tabel kecil, lebih efisien untuk tidak memiliki kunci pengurutan untuk menghindari overhead penyimpanan kunci pengurutan.

Saat menyortir tabel besar dengan ALTER TABLE, sumber daya cluster dikonsumsi dan kunci tabel diperlukan pada berbagai waktu. Menerapkan setiap rekomendasi ketika beban kerja cluster sedang. Rincian lebih lanjut tentang mengoptimalkan konfigurasi kunci pengurutan tabel dapat ditemukan di Buku Pedoman Desain Tabel Tingkat Lanjut Amazon Redshift Engineering: Compound and Interleaved Sort Keys.

Untuk informasi lebih lanjut tentang ALTER SORTKEY, lihat. ALTER TABLE

> **Note**
>
> Jika Anda tidak melihat rekomendasi untuk tabel, itu tidak berarti bahwa konfigurasi saat ini adalah yang terbaik. Penasihat tidak memberikan rekomendasi ketika tidak ada cukup data atau manfaat yang diharapkan dari penyortiran kecil.
> Rekomendasi penasihat berlaku untuk tabel tertentu dan tidak selalu berlaku untuk tabel yang berisi kolom dengan nama dan tipe data yang sama. Tabel yang berbagi nama kolom dapat memiliki rekomendasi yang berbeda berdasarkan data dalam tabel dan beban kerja.

Ubah pengkodean kompresi pada kolom

Kompresi adalah operasi tingkat kolom yang mengurangi ukuran data saat disimpan. Kompresi digunakan di Amazon Redshift untuk menghemat ruang penyimpanan dan meningkatkan kinerja kueri dengan mengurangi jumlah disk I/O. Kami merekomendasikan pengkodean kompresi optimal untuk setiap kolom berdasarkan tipe data dan pola kueri. Dengan kompresi optimal, query dapat berjalan lebih efisien dan database dapat mengambil ruang penyimpanan minimal.

Analisis

Advisor melakukan analisis beban kerja cluster dan skema database secara terus-menerus untuk mengidentifikasi pengkodean kompresi optimal untuk setiap kolom tabel.

Rekomendasi

Advisor menyediakan pernyataan ALTER TABLE yang mengubah pengkodean kompresi kolom tertentu, berdasarkan analisisnya.

Mengubah pengkodean kompresi kolom ALTER TABLE dengan menghabiskan sumber daya cluster dan membutuhkan kunci tabel di berbagai waktu. Yang terbaik adalah menerapkan rekomendasi saat beban kerja cluster ringan.

Untuk referensi, Contoh ALTER TABLE menunjukkan beberapa pernyataan yang mengubah pengkodean untuk kolom.

> **Note**
>
> Penasihat tidak memberikan rekomendasi ketika tidak ada cukup data atau manfaat yang diharapkan dari mengubah pengkodean kecil.

Rekomendasi tipe data

Amazon Redshift memiliki perpustakaan tipe data SQL untuk berbagai kasus penggunaan. Ini termasuk tipe integer seperti `INT` dan tipe untuk menyimpan karakter, seperti`VARCHAR`. Redshift menyimpan jenis dengan cara yang dioptimalkan untuk memberikan akses cepat dan kinerja kueri yang baik. Selain itu, Redshift menyediakan fungsi untuk tipe tertentu, yang dapat Anda gunakan untuk memformat atau melakukan perhitungan pada hasil kueri.

Analisis

Advisor melakukan analisis beban kerja cluster dan skema database secara terus-menerus untuk mengidentifikasi kolom yang dapat memperoleh manfaat signifikan dari perubahan tipe data.

Rekomendasi

Advisor memberikan `ALTER TABLE` pernyataan yang menambahkan kolom baru dengan tipe data yang disarankan. Per `UPDATE` nyataan yang menyertainya menyalin data dari kolom yang ada ke kolom baru. Setelah Anda membuat kolom baru dan memuat data, ubah kueri dan skrip penyerapan Anda untuk mengakses kolom baru. Kemudian manfaatkan fitur dan fungsi khusus untuk tipe data baru, yang ditemukan diReferensi fungsi SQL.

Menyalin data yang ada ke kolom baru bisa memakan waktu. Kami menyarankan Anda menerapkan setiap rekomendasi penasihat ketika beban kerja cluster ringan. Referensikan daftar tipe data yang tersedia diJenis Data.

Perhatikan bahwa Advisor tidak memberikan rekomendasi ketika tidak ada cukup data atau manfaat yang diharapkan dari mengubah tipe data kecil.

Tutorial untuk Amazon Redshift

Ikuti langkah-langkah dalam tutorial ini untuk mempelajari fitur Amazon Redshift:

Tutorial: Memuat data dari Amazon S3	Dalam tutorial ini, Anda berjalan melalui proses memuat data ke tabel database Amazon Redshift Anda dari file data dalam bucket S3 dari awal hingga akhir.
Tutorial: Menanyakan data bersarang dengan Amazon Redshift Spectrum	Dalam tutorial ini, Anda menggunakan Redshift Spectrum untuk menanyakan data bersarang . Dengan Redshift Spectrum, Anda dapat menanyakan data dalam format file Parquet, ORC, JSON, atau Ion.
Tutorial: Mengkonfigurasi antrian manajemen beban kerja manual (WLM)	Dalam tutorial ini, Anda mengonfigurasi Amazon Redshift untuk menggunakan antrian manajemen beban kerja manual (WLM). Amazon Redshift menggunakan antrian WLM untuk mengelola bagaimana sumber daya dibagi untuk menjalankan kueri bersamaan. Anda harus mengonfigurasi Amazon Redshift untuk menggunakan WLM manual jika Anda perlu menggunakan beberapa antrian WLM WLM.
Tutorial: Menggunakan fungsi SQL spasial dengan Amazon Redshift	Dalam tutorial ini, Anda menanyakan data menggunakan fungsi spasial. Anda menggunak an fungsi spasial untuk menanyakan data geometri dan geografi.
Tutorial untuk Amazon Redshift ML	Dalam tutorial ini, Anda membuat dan menggunakan model pembelajaran mesin.
Tutorial: Membuat peran dan menanyakan dengan RBAC	Dalam tutorial ini, Anda menggunakan kontrol akses berbasis peran (RBAC) untuk membuat dan menggunakan izin dalam database

	yang Anda buat. Dengan RBAC, Anda dapat membuat peran dengan izin tertentu, seperti izin baca-saja atau baca-tulis. Anda kemudian dapat menetapkan peran ini kepada pengguna untuk memberi mereka izin yang ditentukan.

Optimasi database otomatis

Amazon Redshift menampung serangkaian fitur otomatis, yang secara kolektif disebut sebagai otonom, yang meningkatkan kinerja, mengurangi pemeliharaan manual, dan mengoptimalkan penggunaan sumber daya. Otonom memanfaatkan pembelajaran mesin dan proses latar belakang untuk mengelola operasi database secara efisien, mengotomatiskan banyak tugas pemeliharaan rutin untuk mengurangi beban kerja administrator basis data.

Tabel berikut merinci fitur otonom Amazon Redshift:

Fitur otonom	Deskripsi
Penyortiran vakum otomatis	Amazon Redshift secara otomatis mengatur ulang data tabel berdasarkan pola kueri yang diamati untuk urutan pengurutan yang optimal. Fitur ini diaktifkan secara default untuk tabel dengan kunci pengurutan tertentu. Untuk informasi selengkapnya, lihat Pengurutan tabel otomatis.
Hapus vakum otomatis	Amazon Redshift secara otomatis menjalank an operasi vakum untuk mendapatkan kembali ruang dari baris yang dihapus dan mengurutk an data. Untuk informasi selengkapnya tentang operasi penghapusan vakum otomatis, lihatHapus vakum otomatis.
Pengoptimalan tabel otomatis	Amazon Redshift memantau kinerja kueri dan metadata tabel untuk secara otomatis menentukan kunci pengurutan dan distribusi terbaik untuk tabel, dan memilih jenis kompresi yang diterapkan ke kolom nilai data saat baris ditambahkan ke tabel. Untuk informasi selengkapnya, lihat Pengoptimalan tabel otomatis dan Pengkodean kompresi.

Fitur otonom	Deskripsi
Analisis otomatis	Amazon Redshift secara otomatis menganalisis tabel saat data di dalamnya berubah, memastikan bahwa perencana kueri memiliki informasi terkini untuk rencana eksekusi yang optimal. Untuk informasi selengkapnya tentang operasi analisis otomatis, lihatAnalisis otomatis.
Tampilan terwujud otomatis	Amazon Redshift secara otomatis membuat dan menyegarkan tampilan yang terwujud berdasarkan pola kueri yang diamati. Ini mengurangi kebutuhan pengguna untuk membuat atau menyegarkan tampilan secara manual untuk mendapatkan manfaat dari respons kueri yang lebih cepat. Untuk informasi lebih lanjut tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

Fitur otonom ini diaktifkan secara default dan berjalan secara otomatis di latar belakang selama periode lalu lintas rendah untuk mengoptimalkan kinerja cluster Anda. Anda dapat merujuk ke nilai parameter Def ault di Panduan Manajemen Amazon Redshift untuk mengonfigurasi fitur otomatis.

Untuk cluster atau kelompok kerja dengan lalu lintas tinggi yang berkelanjutan, sebaiknya aktifkan sumber daya komputasi tambahan untuk pengoptimalan berkelanjutan. Untuk informasi selengkapnya, lihat Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis.

Topik

- Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis
- Penagihan untuk operasi otonom
- Metrik penggunaan untuk operasi otonom

Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis

Otonom meningkatkan kinerja kueri pada cluster Anda dan membantu mengelola biaya penyimpanan dan pengoptimalan. Anda dapat memilih untuk mengalokasikan sumber daya komputasi tambahan untuk memastikan bahwa fitur otonom berjalan secara konsisten, bahkan selama masa lalu lintas tinggi, untuk mendapatkan keuntungan dari pengoptimalan. Perhatikan bahwa otonom yang dijalankan menggunakan sumber daya tambahan ini dapat ditagih. Untuk informasi selengkapnya tentang penagihan untuk otonom, lihat Penagihan untuk operasi otonom dan harga Amazon Redshift.

Jika sumber daya komputasi tambahan untuk otonom dinonaktifkan, Amazon Redshift sementara menangguhkan operasi otonom selama periode beban sistem yang tinggi untuk meminimalkan dampak pada beban kerja bersamaan, sampai ada sumber daya yang cukup untuk menjalankannya tanpa berdampak negatif pada kueri pengguna, yang berpotensi berdampak pada kinerja.

Mengelola sumber daya komputasi tambahan untuk otonom

Secara default, sumber daya komputasi tambahan untuk otonom dinonaktifkan untuk cluster yang disediakan dan kelompok kerja tanpa server, dengan pengecualian kelompok kerja tanpa server dengan penskalaan dan pengoptimalan. AI-driven Untuk kelompok kerja tanpa server dengan AI-driven penskalaan dan pengoptimalan, sumber daya komputasi tambahan untuk otonom selalu diaktifkan dan dikelola secara otomatis tergantung pada profil harga-kinerja Anda.

Anda dapat memilih untuk mengaktifkan sumber daya komputasi tambahan baik selama atau setelah membuat cluster atau workgroup.

Mengelola sumber daya komputasi tambahan untuk otonom pada cluster yang disediakan menggunakan konsol Amazon Redshift

Saat membuat cluster baru, Anda dapat mengaktifkan sumber daya komputasi tambahan untuk otonom dengan memilih Aktifkan komputasi tambahan di panel konfigurasi Autonomics. Untuk melakukannya:

- Pilih Buat klaster.
- Di panel konfigurasi Autonomics, pilih Aktifkan kom putasi tambahan.
- Anda akan membuat cluster dengan komputasi tambahan diaktifkan.

Untuk informasi selengkapnya, lihat Membuat cluster di Panduan Manajemen Amazon Redshift.

Untuk mengaktifkan sumber daya komputasi tambahan setelah membuat cluster:

- Arahkan ke cluster di konsol Amazon Redshift.
- Di halaman detail cluster, pilih Edit konfigurasi otonom dari menu drop-down T indakan.
- Pilih Aktifkan komputasi tambahan.

Mengelola otonom komputasi ekstra untuk cluster yang disediakan menggunakan AWS CLI

Untuk membuat cluster yang disediakan dengan sumber daya komputasi tambahan untuk otonom, gunakan `create-cluster` perintah dengan opsi. `extra-compute-for-automatic-optimization`

```
aws redshift create-cluster --cluster-identifier <value> \
--node-type <value> \
--cluster-type <value> \
--master-username <value> \
--master-user-password <value> \
--extra-compute-for-automatic-optimization \
--region <value>
```

Untuk mengaktifkan sumber daya komputasi tambahan untuk otonom pada cluster yang disediakan, gunakan `modify-cluster` perintah dengan opsi. `extra-compute-for-automatic-optimization`

```
aws redshift modify-cluster --cluster-identifier <value> \
--extra-compute-for-automatic-optimization \
--region <value>
```

Untuk menonaktifkan sumber daya komputasi tambahan untuk otonom pada cluster yang disediakan, gunakan `modify-cluster` perintah dengan opsi. `no-extra-compute-for-automatic-optimization`

```
aws redshift modify-cluster --cluster-identifier <value> \
--no-extra-compute-for-automatic-optimization \
--region <value>
```

Mengelola otonom komputasi ekstra untuk kelompok kerja tanpa server menggunakan konsol Amazon Redshift

Saat membuat grup kerja baru, Anda dapat mengaktifkan sumber daya komputasi tambahan untuk otonom dengan memilih Aktifkan komputasi tambahan di panel konfigurasi Autonomics.

- Pilih Buat grup kerja.
- Pada Langkah 1, di panel konfigurasi Aut onomics, pilih Aktifkan kom putasi tambahan.
- Anda akan membuat workgroup tanpa server dengan komputasi tambahan diaktifkan.

> **Note**
>
> Opsi Aktifkan kom putasi tambahan mungkin tidak tersedia jika Anda telah memilih Price-performance target. Ini karena untuk kelompok kerja tanpa server dengan AI-driven penskalaan dan pengoptimalan, sumber daya komputasi tambahan untuk otonom secara otomatis diaktifkan dan dikelola berdasarkan profil harga-kinerja Anda.

Untuk mengaktifkan sumber daya komputasi tambahan setelah membuat workgroup:

- Arahkan ke grup kerja di konsol Amazon Redshift Serverless.
- Dari halaman detail kelompok kerja, pilih Edit konfigurasi otonom dari menu drop-down Tind akan.
- Pilih Aktifkan komputasi tambahan.

> **Note**
>
> Opsi Edit konfigurasi otonom mungkin berwarna abu-abu jika kelompok kerja memiliki Price-performance set target. Ini karena untuk kelompok kerja tanpa server dengan AI-driven penskalaan dan pengoptimalan, sumber daya komputasi tambahan untuk otonom secara otomatis diaktifkan dan dikelola berdasarkan profil harga-kinerja Anda.

Mengelola otonom komputasi ekstra untuk kelompok kerja tanpa server menggunakan AWS CLI

Untuk membuat workgroup tanpa server dengan sumber daya komputasi tambahan yang dialokasikan untuk otonom, gunakan `create-workgroup` perintah dengan opsi. `extra-compute-for-automatic-optimization`

```
aws redshift-serverless create-workgroup --base-capacity <value> \
--namespace-name <value> \
--workgroup-name <value> \
--extra-compute-for-automatic-optimization \
--region <value>
```

Untuk mengaktifkan sumber daya komputasi tambahan untuk otonom pada kelompok kerja tanpa server yang ada, gunakan `update-workgroup` perintah dengan opsi. `--extra-compute-for-automatic-optimization`

```
aws redshift-serverless update-workgroup \
--workgroup-name <value> \
--extra-compute-for-automatic-optimization \
--region <value>
```

Untuk menonaktifkan sumber daya komputasi tambahan untuk otonom pada workgroup tanpa server yang ada, gunakan `update-workgroup` perintah dengan opsi. `--no-extra-compute-for-automatic-optimization`

```
aws redshift-serverless update-workgroup \
--workgroup-name <value> \
--no-extra-compute-for-automatic-optimization \
--region <value>
```

Kontrol biaya untuk otonom menggunakan sumber daya komputasi tambahan

Kontrol biaya untuk otonom komputasi ekstra pada cluster yang disediakan

Anda dapat menetapkan batas penggunaan pada sumber daya komputasi tambahan yang digunakan untuk otonom pada cluster yang disediakan menggunakan konsol Amazon Redshift. Sumber daya komputasi tambahan yang digunakan untuk otonom tidak diperhitungkan terhadap batasan apa pun yang mungkin telah Anda tetapkan untuk penskalaan konkurensi dan sebaliknya.

Untuk membuat batas penggunaan untuk komputasi tambahan menggunakan konsol:

- Arahkan ke halaman detail cluster dan pilih tab Maintenance dari menu navigasi cluster.
- Gulir ke bawah dan pilih Buat batas penggunaan, itu akan menavigasi ke halaman Kel ola batas penggunaan.
- Pada halaman Kel ola batas penggunaan, cari bagian berjudul Batas penggunaan untuk komputasi tambahan untuk pengoptimalan otomatis dan pilih Tam bah batas.

Untuk membuat batas penggunaan untuk komputasi tambahan menggunakan AWS CLI, gunakan `create-usage-limit` perintah dengan jenis `extra-compute-for-automatic-optimization` fitur. `--limit-type`Parameter harus diatur `time` untuk jenis fitur ini.

```
aws redshift create-usage-limit \
--cluster-identifier <value> \
--feature-type extra-compute-for-automatic-optimization \
--limit-type time \
--period <value> \
--amount <value> \
--breach-action <value>
```

`--breach-action`Param `--period` eter dan menerima nilai yang sama dengan batas penggunaan lainnya. Untuk informasi selengkapnya, lihat Menyetel batas penggunaan di Panduan Manajemen Amazon Redshift.

Kontrol biaya untuk otonom komputasi ekstra pada kelompok kerja tanpa server

Amazon Redshift Serverless menyediakan kemampuan untuk menetapkan batas penggunaan untuk kelompok kerja tanpa server, lihat Menyetel batas penggunaan, termasuk menetapkan batas RPU

di Panduan Manajemen Amazon Redshift. Penggunaan untuk otonom komputasi ekstra di Amazon Redshift Serverless termasuk dalam penggunaan tanpa server secara keseluruhan, termasuk kueri pengguna.

Anda dapat melihat berapa banyak RPU-seconds yang digunakan grup kerja tanpa server Anda untuk otonom komputasi tambahan menggunakan konsol Amazon Redshift Serverless. Untuk melakukannya:

- Arahkan ke konsol Amazon Redshift Serverless.
- Pilih Pemantauan sumber daya dari sidebar.
- Pilih kelompok kerja.
- Gulir ke bawah ke grafik Kom putasi ekstra untuk pengoptimalan otomatis yang dibebankan detik.

Untuk informasi selengkapnya, lihat Amazon Redshift Serverless console di Panduan Manajemen Amazon Redshift.

Anda juga dapat memeriksa tampilan PENGGUNAAN_SYS_TANPA SERVER sistem untuk menemukan`charged_extra_compute_for_automatic_optimization_seconds`, yang mencatat akumulasi unit komputasi (RPU) detik yang dibebankan untuk pengoptimalan otomatis.

Penagihan untuk operasi otonom

Secara default, Amazon Redshift tidak akan menagih Anda untuk sumber daya yang digunakan untuk otonom. Jika Anda memilih untuk mengalokasikan sumber daya komputasi tambahan untuk otonom yang lebih konsisten, Amazon Redshift hanya akan menagih Anda untuk operasi otonom yang benar-benar menggunakan sumber daya tambahan, seperti cluster penskalaan bersamaan (atau RPU tambahan), yang dialokasikan hanya ketika cluster utama atau CPU dasar sepenuhnya dimanfaatkan saat menjalankan beban kerja pengguna. Untuk informasi selengkapnya, lihat Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis dan Penskalaan konkurensi.

Penagihan otonom untuk cluster yang disediakan

Pada cluster yang disediakan, penagihan untuk operasi otonom mengikuti logika berikut:

Jenis operasi otonom	Perilaku penagihan
Amazon Redshift menjalankan operasi otonom di cluster utama.	Amazon Redshift tidak menagih Anda untuk operasi otonom.
Amazon Redshift menjalankan operasi otonom pada cluster penskalaan konkurensi.	Amazon Redshift menagih Anda untuk operasi otonom.

Untuk informasi selengkapnya, lihat harga https://aws.amazon.com/redshift/pricing/ Amazon Redshift.

Penagihan otonom untuk kelompok kerja tanpa server

Pada kelompok kerja tanpa server, penagihan untuk operasi otonom mengikuti logika berikut:

Jenis operasi otonom	Perilaku penagihan
Amazon Redshift menjalankan kueri pengguna dan sementara itu menyelesaikan operasi otonom.	Amazon Redshift tidak menagih Anda untuk operasi otonom.
Amazon Redshift harus mengalokasikan kapasitas komputasi tambahan untuk eksekusi otonom selama periode beban sistem yang tinggi.	Amazon Redshift menagih Anda untuk kapasitas komputasi tambahan yang digunakan pada eksekusi otonom.
Amazon Redshift mulai menjalankan operasi otonom sementara tidak ada kueri pengguna yang berjalan. Amazon Redshift hanya akan melakukan ini jika Anda telah mengaktifkan sumber daya komputasi tambahan untuk otonom.	Amazon Redshift menagih Anda untuk operasi otonom.

Anda hanya ditagih sekali untuk aktivitas yang tumpang tindih dari beban kerja pengguna atau otonom. Misalnya, Amazon Redshift memulai operasi otonom saat tidak ada kueri pengguna yang berjalan, dan permintaan pengguna dimulai selama operasi itu. Dalam hal ini, penagihan diterapkan sekali untuk interval di mana kelompok kerja aktif.

Untuk informasi selengkapnya, lihat Penagihan untuk Amazon Redshift Serverless di Panduan Manajemen Amazon Redshift.

Metrik penggunaan untuk operasi otonom

Jika Anda telah mengaktifkan sumber daya komputasi tambahan untuk pengoptimalan otomatis, Anda dapat melihat informasi tentang penggunaan otonom yang ditagih menggunakan konsol Amazon Redshift. Untuk informasi selengkapnya tentang metrik otonom di konsol untuk kelompok kerja tanpa server, lihat bagian manajemen sumber daya di konsol Amazon Redshift Serverless di Panduan Manajemen Amazon Redshift.

Tabel sistem berikut menyimpan metrik penggunaan tentang otonom:

- SYS_EXTRA_COMPUTE_UNTUK_OPTIMASI_OTOMATIS- Gunakan SYS_EXTRA_COMPUTE_FOR_AUTOMATIC_OPTIMIZATION untuk melihat jumlah waktu Amazon Redshift menjalankan operasi otonom cluster yang disediakan dalam periode penggunaan tertentu.
- SYS_OTOMATIS_OPTIMASI- Gunakan SYS_AUTOMATIC_OPTIMIZATION untuk melihat informasi terperinci tentang operasi otonom untuk cluster yang disediakan dan kelompok kerja tanpa server.
- PENGGUNAAN_SYS_TANPA SERVER- Gunakan SYS_SERVERLESS_USAGE untuk melihat jumlah waktu Amazon Redshift menjalankan operasi otonom kelompok kerja tanpa server dalam periode penggunaan tertentu.

Pengoptimalan tabel otomatis

Pengoptimalan tabel otomatis adalah kemampuan penyetelan mandiri yang secara otomatis mengoptimalkan desain tabel dengan menerapkan kunci pengurutan dan distribusi tanpa perlu intervensi administrator. Dengan menggunakan otomatisasi untuk menyetel desain tabel, Anda dapat memulai dan mendapatkan kinerja tercepat tanpa menginvestasikan waktu untuk menyetel dan menerapkan pengoptimalan tabel secara manual.

Pengoptimalan tabel otomatis terus mengamati bagaimana kueri berinteraksi dengan tabel. Ini menggunakan metode kecerdasan buatan tingkat lanjut untuk memilih kunci pengurutan dan distribusi untuk mengoptimalkan kinerja untuk beban kerja cluster. Jika Amazon Redshift menentukan bahwa menerapkan kunci meningkatkan kinerja cluster, tabel secara otomatis diubah dalam beberapa jam sejak cluster dibuat, dengan dampak minimal terhadap kueri.

Untuk memanfaatkan otomatisasi ini, administrator Amazon Redshift membuat tabel baru, atau mengubah tabel yang ada untuk mengaktifkannya menggunakan pengoptimalan otomatis. Tabel yang ada dengan gaya distribusi atau kunci pengurutan sudah `AUTO` diaktifkan untuk otomatisasi. Saat Anda menjalankan kueri terhadap tabel tersebut, Amazon Redshift menentukan apakah kunci pengurutan atau kunci distribusi akan meningkatkan kinerja. Jika demikian, maka Amazon Redshift secara otomatis memodifikasi tabel tanpa memerlukan intervensi administrator. Jika jumlah kueri minimum dijalankan, pengoptimalan diterapkan dalam beberapa jam setelah cluster diluncurkan.

Jika Amazon Redshift menentukan bahwa kunci distribusi meningkatkan kinerja kueri, tabel yang memiliki gaya distribusi `AUTO` dapat diubah menjadi gaya distribusinya. `KEY`

Topik

- Mengaktifkan, menonaktifkan, dan memantau pengoptimalan tabel otomatis
- Mengelola pengecualian beban kerja dari Autonomics
- Kompresi kolom untuk mengurangi ukuran data yang disimpan
- Distribusi data untuk optimasi kueri
- Urutkan kunci
- Kendala tabel

Mengaktifkan, menonaktifkan, dan memantau pengoptimalan tabel otomatis

Secara default, tabel yang dibuat tanpa secara eksplisit mendefinisikan kunci pengurutan atau kunci distribusi diatur keAUTO. Pada saat pembuatan tabel, Anda juga dapat secara eksplisit mengatur pengurutan atau kunci distribusi secara manual. Jika Anda mengatur kunci pengurutan atau distribusi, maka tabel tidak dikelola secara otomatis.

Mengaktifkan pengoptimalan tabel otomatis

Untuk mengaktifkan tabel yang ada agar dioptimalkan secara otomatis, gunakan opsi pernyataan ALTER untuk mengubah tabel menjadiAUTO. Anda mungkin memilih untuk mendefinisikan otomatisasi untuk kunci pengurutan, tetapi tidak untuk kunci distribusi (dan sebaliknya). Jika Anda menjalankan pernyataan ALTER untuk mengonversi tabel menjadi tabel otomatis, kunci pengurutan dan gaya distribusi yang ada akan dipertahankan.

```
ALTER TABLE table_name ALTER SORTKEY AUTO;
```

```
ALTER TABLE table_name ALTER DISTSTYLE AUTO;
```

Untuk informasi selengkapnya, lihat ALTER TABLE.

Awalnya, tabel tidak memiliki kunci distribusi atau kunci pengurutan. Gaya distribusi diatur ke salah satu EVEN atau ALL tergantung pada ukuran tabel. Seiring bertambahnya ukuran tabel, Amazon Redshift menerapkan kunci distribusi optimal dan kunci pengurutan. Optimasi diterapkan dalam beberapa jam setelah jumlah kueri minimum dijalankan. Saat menentukan pengoptimalan kunci pengurutan, Amazon Redshift mencoba mengoptimalkan blok data yang dibaca dari disk selama pemindaian tabel. Saat menentukan pengoptimalan gaya distribusi, Amazon Redshift mencoba mengoptimalkan jumlah byte yang ditransfer antar node cluster.

Menghapus pengoptimalan tabel otomatis dari tabel

Anda dapat menghapus tabel dari pengoptimalan otomatis. Menghapus tabel dari otomatisasi melibatkan pemilihan kunci pengurutan atau gaya distribusi. Untuk mengubah gaya distribusi, tentukan gaya distribusi tertentu.

```
ALTER TABLE table_name ALTER DISTSTYLE EVEN;
```

```
ALTER TABLE table_name ALTER DISTSTYLE ALL;
```

```
ALTER TABLE table_name ALTER DISTSTYLE KEY DISTKEY c1;
```

Untuk mengubah kunci pengurutan, Anda dapat menentukan kunci pengurutan atau memilih tidak ada.

```
ALTER TABLE table_name ALTER SORTKEY(c1, c2);
```

```
ALTER TABLE table_name ALTER SORTKEY NONE;
```

Pemantauan pengoptimalan tabel otomatis

Tampilan sistem SVV_ALTER_TABLE_RECOMMENDATIONS mencatat rekomendasi Amazon Redshift Advisor saat ini untuk tabel. Tampilan ini menunjukkan rekomendasi untuk semua tabel, yang ditentukan untuk pengoptimalan otomatis dan yang tidak.

Untuk melihat apakah tabel didefinisikan untuk pengoptimalan otomatis, kueri tampilan sistemSVV_TABLE_INFO. Entri hanya muncul untuk tabel yang terlihat di database sesi saat ini. Rekomendasi dimasukkan ke dalam tampilan dua kali per hari mulai dalam beberapa jam sejak cluster dibuat. Setelah rekomendasi tersedia, itu dimulai dalam satu jam. Setelah rekomendasi diterapkan (baik oleh Amazon Redshift atau oleh Anda), itu tidak lagi muncul di tampilan.

Tampilan sistem SVL_AUTO_WORKER_ACTION menunjukkan log audit dari semua tindakan yang dilakukan oleh Amazon Redshift, dan status tabel sebelumnya.

Tampilan sistem SVV_TABLE_INFO mencantumkan semua tabel dalam sistem, bersama dengan kolom untuk menunjukkan apakah kunci pengurutan dan gaya distribusi tabel diatur keAUTO.

Untuk informasi selengkapnya tentang tampilan sistem ini, lihatPemantauan sistem (hanya disediakan).

Mengelola pengecualian beban kerja dari Autonomics

Anda dapat mengecualikan titik akhir tertentu yang disediakan atau kelompok kerja tanpa server agar tidak mempengaruhi keputusan otonom seperti kunci distribusi dan kunci pengurutan melalui fitur

denylist. Ini memungkinkan Anda mengontrol beban kerja yang dipertimbangkan Amazon Redshift saat membuat keputusan pengoptimalan untuk data Redshift Managed Storage (RMS) Anda.

Menggunakan Denylist

Anda dapat mengelola daftar penolakan melalui bagian Autonomics di konsol Amazon Redshift:

1. Menambahkan atau menghapus item

   Tambahkan titik akhir tertentu yang disediakan atau grup kerja tanpa server ke daftar penolakan, dan hapus jika diperlukan.
2. Lihat dan cari

   Lihat semua item yang dicantumkan dan cari titik akhir atau kelompok kerja tertentu di daftar penolakan.

Denylist sangat berguna ketika Anda menjalankan pasar data, berbagi data dengan pengguna eksternal, atau memiliki beberapa unit bisnis, di mana Anda ingin mencegah pola penggunaan tertentu mempengaruhi keputusan pengoptimalan yang sesuai dengan beban kerja Anda. Misalnya, jika kelompok kerja A menjalankan beban kerja A, dan kelompok kerja B menjalankan beban kerja B pada tabel bersama T, kunci pengurutan T akan ditentukan oleh beban kerja A dan B. Jika Anda ingin hanya beban kerja A yang mempengaruhi keputusan kunci pengurutan, maka tambahkan kelompok kerja B ke daftar tolak titik akhir atau kelompok kerja yang memiliki tabel T. Secara default, Amazon Redshift Autonomics mempertimbangkan pola kueri dari produsen dan semua konsumen clusters/ workgroups kecuali secara khusus dikecualikan melalui daftar penolakan.

> **Note**
>
> Anda dapat menolak daftar sumber daya di berbagai Wil AWS ayah dalam akun yang sama. Cross-account denylisting belum didukung.

Mengelola sumber daya denylist di konsol Amazon Redshift

Di konsol Amazon Redshift Serverless, selesaikan langkah-langkah berikut:

1. Pilih Cluster atau Serverless workgroup.
2. Arahkan ke halaman detail cluster atau kelompok kerja tertentu.
3. Pilih Otonom di bagian tab.

4. Di bawah tab Autonomics, Anda dapat melihat dan mengelola daftar penolakan Anda.
5. Untuk mengelola daftar penolakan lintas wilayah, pilih Wilayah yang sesuai. AWS

Menambahkan sumber daya denylist

1. Arahkan ke tab Autonomics dari cluster atau kelompok kerja yang dipilih, pilih Wil AWS ayah, dan pilih Tambahkan sumber daya.
2. Pilih satu atau beberapa cluster yang disediakan atau grup kerja tanpa server untuk ditambahkan ke daftar penolakan dan pilih Tambah.
3. Tabel menunjukkan daftar sumber daya denylist.

Hapus sumber daya denylist

1. Arahkan ke tab Autonomics dari cluster atau kelompok kerja yang dipilih, pilih Wilayah. AWS
2. Pilih cluster atau kelompok kerja yang ingin Anda hapus dari daftar dan pilih Hapus.
3. Kotak dialog konfirmasi terbuka. Pilih Hapus untuk mengonfirmasi.

Kompresi kolom untuk mengurangi ukuran data yang disimpan

Kompresi adalah operasi tingkat kolom yang mengurangi ukuran data saat disimpan. Kompresi menghemat ruang penyimpanan dan mengurangi ukuran data yang dibaca dari penyimpanan, yang mengurangi jumlah disk I/O dan karenanya meningkatkan kinerja kueri.

ENCODE AUTO adalah default untuk tabel. Saat tabel disetel ke ENCODE AUTO, Amazon Redshift secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel. Untuk informasi selengkapnya, lihat CREATE TABLE dan ALTER TABLE.

Namun, jika Anda menentukan pengkodean kompresi untuk kolom apa pun dalam tabel, tabel tidak lagi disetel ke ENCODE AUTO. Amazon Redshift tidak lagi secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel.

Anda dapat menerapkan jenis kompresi, atau pengkodean, ke kolom dalam tabel secara manual saat Anda membuat tabel. Atau Anda dapat menggunakan perintah COPY untuk menganalisis dan menerapkan kompresi secara otomatis. Untuk informasi selengkapnya, lihat Biarkan COPY memilih pengkodean kompresi. Untuk detail tentang menerapkan kompresi otomatis, lihatMemuat tabel dengan kompresi otomatis.

> **Note**
>
> Kami sangat menyarankan menggunakan perintah COPY untuk menerapkan kompresi otomatis.

Anda dapat memilih untuk menerapkan pengkodean kompresi secara manual jika tabel baru berbagi karakteristik data yang sama dengan tabel lain. Atau Anda dapat melakukannya jika Anda menemukan dalam pengujian bahwa pengkodean kompresi yang diterapkan selama kompresi otomatis bukan yang paling cocok untuk data Anda. Jika Anda memilih untuk menerapkan pengkodean kompresi secara manual, Anda dapat menjalankan MENGANALISIS KOMPRESI perintah terhadap tabel yang sudah diisi dan menggunakan hasilnya untuk memilih pengkodean kompresi.

Untuk menerapkan kompresi secara manual, Anda menentukan pengkodean kompresi untuk kolom individual sebagai bagian dari pernyataan CREATE TABLE. Sintaksnya adalah sebagai berikut.

```
CREATE TABLE table_name (column_name
data_type ENCODE encoding-type)[, ...]
```

Di sini, tipe pengkodean diambil dari tabel kata kunci di bagian berikut.

Misalnya, pernyataan berikut membuat tabel dua kolom, PRODUCT. Ketika data dimuat ke dalam tabel, kolom PRODUCT_ID tidak dikompresi, tetapi kolom PRODUCT_NAME dikompresi, menggunakan pengkodean kamus byte (BYTEDICT).

```
create table product(
product_id int encode raw,
product_name char(20) encode bytedict);
```

Anda dapat menentukan pengkodean untuk kolom ketika ditambahkan ke tabel menggunakan perintah ALTER TABLE.

```
ALTER TABLE table-name ADD [ COLUMN ] column_name column_type ENCODE encoding-type
```

Topik

- Pengkodean kompresi
- Menguji pengkodean kompresi

Pengkodean kompresi

Pengkodean kompresi menentukan jenis kompresi yang diterapkan ke kolom nilai data saat baris ditambahkan ke tabel.

ENCODE AUTO adalah default untuk tabel. Saat tabel disetel ke ENCODE AUTO, Amazon Redshift secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel. Untuk informasi selengkapnya, lihat CREATE TABLE dan ALTER TABLE.

Namun, jika Anda menentukan pengkodean kompresi untuk kolom apa pun dalam tabel, tabel tidak lagi disetel ke ENCODE AUTO. Amazon Redshift tidak lagi secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel.

Saat Anda menggunakan CREATE TABLE, ENCODE AUTO dinonaktifkan saat Anda menentukan pengkodean kompresi untuk kolom apa pun dalam tabel. Jika ENCODE AUTO dinonaktifkan, Amazon Redshift secara otomatis menetapkan pengkodean kompresi ke kolom yang tidak Anda tentukan jenis ENCODE sebagai berikut:

- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, atau DOUBLE PRECISION diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai tipe data CHAR atau VARCHAR diberi kompresi LZO.

Anda dapat mengubah pengkodean tabel setelah membuatnya dengan menggunakan ALTER TABLE. Jika Anda menonaktifkan ENCODE AUTO menggunakan ALTER TABLE, Amazon Redshift tidak lagi secara otomatis mengelola pengkodean kompresi untuk kolom Anda. Semua kolom akan menyimpan jenis pengkodean kompresi yang mereka miliki ketika Anda menonaktifkan ENCODE AUTO sampai Anda mengubahnya atau Anda mengaktifkan ENCODE AUTO lagi.

Amazon Redshift mendukung pengkodean kompresi berikut:

Raw

> Penyandian mentah adalah pengkodean default untuk kolom yang ditetapkan sebagai kunci pengurutan dan kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, atau DOUBLE PRECISION. Dengan penyandian mentah, data disimpan dalam bentuk mentah dan tidak terkompresi.

AZ64

AZ64 adalah algoritma pengkodean kompresi eksklusif yang dirancang oleh Amazon untuk mencapai rasio kompresi tinggi dan pemrosesan kueri yang lebih baik. Pada intinya, algoritma AZ64 memampatkan kelompok nilai data yang lebih kecil dan menggunakan instruksi tunggal, instruksi multiple data (SIMD) untuk pemrosesan paralel. Gunakan AZ64 untuk mencapai penghematan penyimpanan yang signifikan dan kinerja tinggi untuk tipe data numerik, tanggal, dan waktu.

Anda dapat menggunakan AZ64 sebagai pengkodean kompresi saat mendefinisikan kolom menggunakan pernyataan CREATE TABLE dan ALTER TABLE dengan tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- DATE
- TIMESTAMP
- TIMESTAMPTZ

Byte-dictionary

Dalam pengkodean kamus byte, kamus terpisah dari nilai unik dibuat untuk setiap blok nilai kolom pada disk. (Blok disk Amazon Redshift menempati 1 MB.) Kamus berisi hingga 256 nilai satu-byte yang disimpan sebagai indeks ke nilai data asli. Jika lebih dari 256 nilai disimpan dalam satu blok, nilai tambahan ditulis ke dalam blok dalam bentuk mentah dan tidak terkompresi. Proses berulang untuk setiap blok disk.

Pengkodean ini sangat efektif pada kolom string kardinalitas rendah. Pengkodean ini optimal ketika domain data kolom kurang dari 256 nilai unik.

Untuk kolom dengan tipe data string (CHAR dan VARCHAR) yang dikodekan dengan BYTEDICT, Amazon Redshift melakukan pemindaian vektor dan evaluasi predikat yang beroperasi pada data terkompresi secara langsung. Pemindaian ini menggunakan instruksi tunggal khusus perangkat keras dan instruksi multiple data (SIMD) untuk pemrosesan paralel. Ini secara signifikan mempercepat pemindaian kolom string. Byte-dictionary pengkodean sangat hemat ruang jika CHAR/VARCHAR kolom memegang string karakter yang panjang.

Misalkan tabel memiliki kolom COUNTRY dengan tipe data CHAR (30). Saat data dimuat, Amazon Redshift membuat kamus dan mengisi kolom COUNTRY dengan nilai indeks. Kamus berisi nilai unik yang diindeks, dan tabel itu sendiri hanya berisi subskrip satu byte dari nilai yang sesuai.

> **Note**
>
> Bagian kosong yang tersisa disimpan untuk kolom karakter dengan panjang tetap. Oleh karena itu, dalam kolom CHAR (30), setiap nilai terkompresi menyimpan 29 byte penyimpanan saat Anda menggunakan pengkodean byte-kamus.

Tabel berikut mewakili kamus untuk kolom COUNTRY.

Nilai data unik	Indeks kamus	Ukuran (panjang tetap, 30 byte per nilai)
England	0	30
United States of America	1	30
Venezuela	2	30
Sri Lanka	3	30
Argentina	4	30
Japan	5	30
Total		180

Tabel berikut mewakili nilai-nilai di kolom COUNTRY.

Nilai data asli	Ukuran asli (panjang tetap, 30 byte per nilai)	Nilai terkompresi (indeks)	Ukuran baru (byte)
England	30	0	1

Nilai data asli	Ukuran asli (panjang tetap, 30 byte per nilai)	Nilai terkompresi (indeks)	Ukuran baru (byte)
England	30	0	1
United States of America	30	1	1
United States of America	30	1	1
Venezuela	30	2	1
Sri Lanka	30	3	1
Argentina	30	4	1
Japan	30	5	1
Sri Lanka	30	3	1
Argentina	30	4	1
Total	300		10

Total ukuran terkompresi dalam contoh ini dihitung sebagai berikut: 6 entri yang berbeda disimpan dalam kamus (6 * 30 = 180), dan tabel berisi 10 nilai terkompresi 1-byte, dengan total 190 byte.

Delta

Pengkodean delta sangat berguna untuk kolom waktu tanggal.

Pengkodean delta memampatkan data dengan merekam perbedaan antara nilai-nilai yang mengikuti satu sama lain di kolom. Perbedaan ini dicatat dalam kamus terpisah untuk setiap blok nilai kolom pada disk. (Blok disk Amazon Redshift menempati 1 MB.) Misalnya, anggaplah kolom berisi 10 bilangan bulat secara berurutan dari 1 hingga 10. Yang pertama disimpan sebagai bilangan bulat 4-byte (ditambah bendera 1-byte). Sembilan berikutnya masing-masing disimpan sebagai byte dengan nilai 1, menunjukkan bahwa itu satu lebih besar dari nilai sebelumnya.

Pengkodean delta hadir dalam dua variasi:

- DELTA mencatat perbedaan sebagai nilai 1-byte (bilangan bulat 8-bit)
- DELTA32K mencatat perbedaan sebagai nilai 2-byte (bilangan bulat 16-bit)

Jika sebagian besar nilai dalam kolom dapat dikompresi dengan menggunakan satu byte, variasi 1-byte sangat efektif. Namun, jika delta lebih besar, pengkodean ini, dalam kasus terburuk, agak kurang efektif daripada menyimpan data yang tidak terkompresi. Logika serupa berlaku untuk versi 16-bit.

Jika perbedaan antara dua nilai melebihi rentang 1-byte (DELTA) atau rentang 2-byte (DELTA32K), nilai asli penuh disimpan, dengan flag 1-byte terdepan. Rentang 1-byte adalah dari -127 hingga 127, dan rentang 2-byte adalah dari -32K hingga 32K.

Tabel berikut menunjukkan bagaimana pengkodean delta bekerja untuk kolom numerik.

Nilai data asli	Ukuran asli (byte)	Perbedaan (delta)	Nilai terkompresi	Ukuran terkompresi (byte)
1	4		1	1+4 (bendera + nilai aktual)
5	4	4	4	1
50	4	45	45	1
200	4	150	150	1+4 (bendera + nilai aktual)
185	4	-15	-15	1
220	4	35	35	1
221	4	1	1	1
Total	28			15

LZO

Pengkodean LZO memberikan rasio kompresi yang sangat tinggi dengan kinerja yang baik. Pengkodean LZO bekerja sangat baik untuk kolom CHAR dan VARCHAR yang menyimpan string karakter yang sangat panjang. Mereka sangat baik untuk teks bentuk bebas, seperti deskripsi produk, komentar pengguna, atau string JSON.

Mostly

Sebagian besar pengkodean berguna ketika tipe data untuk kolom lebih besar dari yang dibutuhkan sebagian besar nilai yang disimpan. Dengan menentukan sebagian besar pengkodean untuk jenis kolom ini, Anda dapat mengompres sebagian besar nilai dalam kolom ke ukuran penyimpanan standar yang lebih kecil. Nilai yang tersisa yang tidak dapat dikompresi disimpan dalam bentuk mentah. Misalnya, Anda dapat mengompres kolom 16-bit, seperti kolom INT2, ke penyimpanan 8-bit.

Secara umum, sebagian besar pengkodean bekerja dengan tipe data berikut:

- SMALLINT/INT2 (16-bit)
- INTEGER/INT (32-bit)
- BIGINT/INT8 (64-bit)
- DECIMAL/NUMERIC (64-bit)

Pilih variasi yang sesuai dari sebagian besar pengkodean agar sesuai dengan ukuran tipe data untuk kolom. Misalnya, terapkan MOSTLY8 ke kolom yang didefinisikan sebagai kolom integer 16-bit. Menerapkan MOSTLY16 ke kolom dengan tipe data 16-bit atau MOSTLY32 ke kolom dengan tipe data 32-bit tidak diperbolehkan.

Sebagian besar pengkodean mungkin kurang efektif daripada tanpa kompresi ketika jumlah nilai yang relatif tinggi di kolom tidak dapat dikompresi. Sebelum menerapkan salah satu pengkodean ini ke kolom, lakukan pemeriksaan. Sebagian besar nilai yang akan Anda muat sekarang (dan kemungkinan akan dimuat di masa mendatang) harus sesuai dengan rentang yang ditunjukkan pada tabel berikut.

Pengodean	Ukuran penyimpanan terkompresi	Rentang nilai yang dapat dikompresi (nilai di luar rentang disimpan mentah)
MAYORIT8	1 byte (8 bit)	-128 hingga 127

Pengodean	Ukuran penyimpanan terkompresi	Rentang nilai yang dapat dikompresi (nilai di luar rentang disimpan mentah)
MAYORIT16	2 byte (16 bit)	-32768 sampai 32767
MAYORIT32	4 byte (32 bit)	-2147483648 sampai +2147483647

> **Note**
>
> Untuk nilai desimal, abaikan titik desimal untuk menentukan apakah nilainya cocok dengan rentang. Misalnya, 1.234.56 diperlakukan sebagai 123.456 dan dapat dikompresi dalam kolom MOSTLY32.

Misalnya, kolom VENUEID dalam tabel VENUE didefinisikan sebagai kolom integer mentah, yang berarti bahwa nilainya menghabiskan 4 byte penyimpanan. Namun, kisaran nilai saat ini di kolom adalah **0** untuk**309**. Oleh karena itu, membuat ulang dan memuat ulang tabel ini dengan pengkodean MOSTLY16 untuk VENUEID akan mengurangi penyimpanan setiap nilai di kolom itu menjadi 2 byte.

Jika nilai VENUEID yang direferensikan dalam tabel lain sebagian besar berada dalam kisaran 0 hingga 127, mungkin masuk akal untuk mengkodekan kolom kunci asing itu sebagai MOSTLY8. Sebelum membuat pilihan, jalankan beberapa kueri terhadap data tabel referensi untuk mengetahui apakah nilai-nilai sebagian besar jatuh ke dalam rentang 8-bit, 16-bit, atau 32-bit.

Tabel berikut menunjukkan ukuran terkompresi untuk nilai numerik tertentu ketika pengkodean MOSTLY8, MOSTLY16, dan MOSTLY32 digunakan:

Nilai asli	Ukuran INT atau BIGINT asli (byte)	MOSTLY8 ukuran terkompresi (byte)	MOSTLY16 ukuran terkompresi (byte)	MOSTLY32 ukuran terkompresi (byte)
1	4	1	2	4
10	4	1	2	4
100	4	1	2	4

Nilai asli	Ukuran INT atau BIGINT asli (byte)	MOSTLY8 ukuran terkompresi (byte)	MOSTLY16 ukuran terkompresi (byte)	MOSTLY32 ukuran terkompresi (byte)
1000	4	Sama seperti ukuran data mentah	2	4
10000	4		2	4
2000	4		2	4
40000	8		Sama seperti ukuran data mentah	4
100000	8			4
2000000000	8			4

Run length

Run length encoding menggantikan nilai yang diulang secara berurutan dengan token yang terdiri dari nilai dan hitungan jumlah kejadian berturut-turut (panjang run). Kamus terpisah dari nilai unik dibuat untuk setiap blok nilai kolom pada disk. (Blok disk Amazon Redshift menempati 1 MB.) Pengkodean ini paling cocok untuk tabel di mana nilai data sering diulang secara berurutan, misalnya, ketika tabel diurutkan berdasarkan nilai-nilai tersebut.

Sebagai contoh, misalkan kolom dalam tabel dimensi besar memiliki domain kecil yang dapat diprediksi, seperti kolom COLOR dengan kurang dari 10 kemungkinan nilai. Nilai-nilai ini cenderung jatuh dalam urutan panjang di seluruh tabel, bahkan jika data tidak diurutkan.

Kami tidak menyarankan menerapkan pengkodean run length pada kolom apa pun yang ditetapkan sebagai kunci pengurutan. Range-restricted pemindaian berkinerja lebih baik ketika blok berisi jumlah baris yang sama. Jika kolom kunci pengurutan dikompresi jauh lebih tinggi daripada kolom lain dalam kueri yang sama, pemindaian terbatas rentang mungkin berkinerja buruk.

Tabel berikut menggunakan contoh kolom COLOR untuk menunjukkan cara kerja pengkodean run length.

Nilai data asli	Ukuran asli (byte)	Nilai terkompresi (token)	Ukuran terkompresi (byte)
Blue	4	{2, Biru}	5
Blue	4		0
Green	5	{3, Hijau}	6
Green	5		0
Green	5		0
Blue	4	{1,Blue}	5
Yellow	6	{4,Yellow}	7
Yellow	6		0
Yellow	6		0
Yellow	6		0
Total	51		23

Text255 and Text32k

Pengkodean Text255 dan text32k berguna untuk mengompresi kolom VARCHAR di mana kata yang sama sering berulang. Kamus kata-kata unik yang terpisah dibuat untuk setiap blok nilai kolom pada disk. (Blok disk Amazon Redshift menempati 1 MB.) Kamus berisi 245 kata unik pertama di kolom. Kata-kata tersebut diganti pada disk dengan nilai indeks satu-byte yang mewakili salah satu dari 245 nilai, dan kata-kata apa pun yang tidak diwakili dalam kamus disimpan tanpa kompresi. Proses berulang untuk setiap blok disk 1-MB. Jika kata-kata yang diindeks sering muncul di kolom, kolom menghasilkan rasio kompresi yang tinggi.

Untuk pengkodean text32k, prinsipnya sama, tetapi kamus untuk setiap blok tidak menangkap sejumlah kata tertentu. Sebaliknya, kamus mengindeks setiap kata unik yang ditemukannya sampai entri gabungan mencapai panjang 32K, dikurangi beberapa overhead. Nilai indeks disimpan dalam dua byte.

Misalnya, pertimbangkan kolom VENUENAME di tabel VENUE. Kata-kata seperti**Arena**,**Center**, dan **Theatre** berulang di kolom ini dan kemungkinan akan berada di antara 245 kata pertama yang ditemui di setiap blok jika kompresi text255 diterapkan. Jika demikian, kolom ini mendapat manfaat dari kompresi. Ini karena setiap kali kata-kata itu muncul, mereka hanya menempati 1 byte penyimpanan (bukan 5, 6, atau 7 byte, masing-masing).

ZSTD

Pengkodean Zstandard (ZSTD) memberikan rasio kompresi tinggi dengan kinerja yang sangat baik di berbagai kumpulan data. ZSTD bekerja sangat baik dengan kolom CHAR dan VARCHAR yang menyimpan berbagai string panjang dan pendek, seperti deskripsi produk, komentar pengguna, log, dan string JSON. Di mana beberapa algoritma, seperti pengkodean Delta atau pengkodean Mostly, berpotensi menggunakan lebih banyak ruang penyimpanan daripada tanpa kompresi, ZSTD tidak mungkin meningkatkan penggunaan disk.

ZSTD mendukung tipe data SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, TIMESTAMP, dan TIMESTAMPTZ.

Tabel berikut mengidentifikasi pengkodean kompresi yang didukung dan tipe data yang mendukung pengkodean.

Jenis pengkodean	Kata kunci dalam CREATE TABLE dan ALTER TABLE	Jenis Data
Mentah (tanpa kompresi)	RAW	Semua
AZ64	AZ64	SMALLINT, INTEGER, BIGINT, DESIMAL, TANGGAL, STEMPEL WAKTU, TIMESTAMPTZ
Kamus byte	BYTEDICT	SMALLINT, INTEGER, BIGINT, DESIMAL, REAL, PRESISI GANDA, CHAR, VARCHAR, TANGGAL, STEMPEL WAKTU, TIMESTAMPTZ
kuala	DELTA DELTA32K	SMALLINT, INT, BIGINT, TANGGAL, STEMPEL WAKTU, DESIMAL

Jenis pengkodean	Kata kunci dalam CREATE TABLE dan ALTER TABLE	Jenis Data
		INT, BIGINT, TANGGAL, CAP WAKTU, DESIMAL
LZO	LZO	SMALLINT, INTEGER, BIGINT, DESIMAL, CHAR, VARCHAR, TANGGAL, STEMPEL WAKTU, TIMESTAMPTZ, SUPER
Sebagian besar n	MAYORIT8	SMALLINT, INT, BIGINT, DESIMAL
	MAYORIT16	INT, BIGINT, DESIMAL
	MAYORIT32	BIGINT, DESIMAL
Run-length	PANJANG	SMALLINT, INTEGER, BIGINT, DESIMAL, REAL, PRESISI GANDA, BOOLEAN, CHAR, VARCHAR, TANGGAL, STEMPEL WAKTU, TIMESTAMPTZ
Teks	TEKS255	Hanya VARCHAR
	TEKS32K	Hanya VARCHAR
Zstandard	ZSTD	SMALLINT, INTEGER, BIGINT, DESIMAL, REAL, PRESISI GANDA, BOOLEAN, CHAR, VARCHAR, TANGGAL, STEMPEL WAKTU, TIMESTAMPTZ, SUPER

Menguji pengkodean kompresi

Jika Anda memutuskan untuk menentukan pengkodean kolom secara manual, Anda mungkin ingin menguji pengkodean yang berbeda dengan data Anda.

> **Note**
>
> Kami menyarankan Anda menggunakan perintah COPY untuk memuat data bila memungkinkan, dan mengizinkan perintah COPY untuk memilih pengkodean optimal berdasarkan data Anda. Atau Anda dapat menggunakan MENGANALISIS KOMPRESI perintah untuk melihat pengkodean yang disarankan untuk data yang ada. Untuk detail tentang menerapkan kompresi otomatis, lihatMemuat tabel dengan kompresi otomatis.

Untuk melakukan tes kompresi data yang berarti, Anda harus memiliki sejumlah besar baris. Untuk contoh ini, buat tabel dan sisipkan baris dengan menggunakan pernyataan yang memilih dari dua tabel: VENUE dan LISTING. Tinggalkan klausa WHERE yang biasanya bergabung dengan dua tabel. Hasilnya adalah bahwa setiap baris dalam tabel VENUE digabungkan ke semua baris dalam tabel LISTING, dengan total lebih dari 32 juta baris. Ini dikenal sebagai gabungan Cartesian dan biasanya tidak disarankan. Namun, untuk tujuan ini, ini adalah metode yang nyaman untuk membuat banyak baris. Jika Anda memiliki tabel yang ada dengan data yang ingin Anda uji, Anda dapat melewati langkah ini.

Setelah Anda memiliki tabel dengan data sampel, buat tabel dengan tujuh kolom. Masing-masing memiliki pengkodean kompresi yang berbeda: raw, bytedict, lzo, run length, text255, text32k, dan zstd. Isi setiap kolom dengan data yang persis sama dengan menjalankan perintah INSERT yang memilih data dari tabel pertama.

Untuk menguji pengkodean kompresi, lakukan hal berikut:

1. (Opsional) Pertama, gunakan gabungan Cartesian untuk membuat tabel dengan sejumlah besar baris. Lewati langkah ini jika Anda ingin menguji tabel yang ada.

```
create table cartesian_venue(
venueid smallint not null distkey sortkey,
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer);

insert into cartesian_venue
select venueid, venuename, venuecity, venuestate, venueseats
from venue, listing;
```

2. Selanjutnya, buat tabel dengan pengkodean yang ingin Anda bandingkan.

```
create table encodingvenue (
venueraw varchar(100) encode raw,
venuebytedict varchar(100) encode bytedict,
venuelzo varchar(100) encode lzo,
venuerunlength varchar(100) encode runlength,
venuetext255 varchar(100) encode text255,
venuetext32k varchar(100) encode text32k,
venuezstd varchar(100) encode zstd);
```

3. Masukkan data yang sama ke semua kolom menggunakan pernyataan INSERT dengan klausa SELECT.

```
insert into encodingvenue
select venuename as venueraw, venuename as venuebytedict, venuename as venuelzo,
 venuename as venuerunlength, venuename as  venuetext32k, venuename as  venuetext255,
 venuename as venuezstd
from cartesian_venue;
```

4. Verifikasi jumlah baris dalam tabel baru.

```
select count(*) from encodingvenue

  count
----------
 38884394
(1 row)
```

5. Kueri tabel STV_BLOCKLIST sistem untuk membandingkan jumlah blok disk 1 MB yang digunakan oleh setiap kolom.

   Fungsi agregat MAX mengembalikan nomor blok tertinggi untuk setiap kolom. Tabel STV_BLOCKLIST menyertakan detail untuk tiga kolom yang dihasilkan sistem. Contoh ini menggunakan kla `col < 6` usa WHERE untuk mengecualikan kolom yang dihasilkan sistem.

```
select col, max(blocknum)
from stv_blocklist b, stv_tbl_perm p
where (b.tbl=p.id) and name ='encodingvenue'
and col < 7
group by name, col
order by col;
```

Query mengembalikan hasil berikut. Kolom diberi nomor dimulai dengan nol. Bergantung pada bagaimana cluster Anda dikonfigurasi, hasil Anda mungkin memiliki angka yang berbeda, tetapi ukuran relatifnya harus serupa. Anda dapat melihat bahwa pengkodean BYTEDICT pada kolom kedua menghasilkan hasil terbaik untuk dataset ini. Pendekatan ini memiliki rasio kompresi yang lebih baik dari 20:1. Pengkodean LZO dan ZSTD juga menghasilkan hasil yang sangat baik. Kumpulan data yang berbeda menghasilkan hasil yang berbeda, tentu saja. Ketika kolom berisi string teks yang lebih panjang, LZO sering menghasilkan hasil kompresi terbaik.

```
 col | max
-----+-----
   0 | 203
   1 |  10
   2 |  22
   3 | 204
   4 |  56
   5 |  72
   6 |  20
(7 rows)
```

Jika Anda memiliki data dalam tabel yang ada, Anda dapat menggunakan MENGANALISIS KOMPRESI perintah untuk melihat pengkodean yang disarankan untuk tabel. Misalnya, contoh berikut menunjukkan pengkodean yang direkomendasikan untuk salinan tabel VENUE, CARTESIAN_VENUE, yang berisi 38 juta baris. Perhatikan bahwa ANALYZE COMPRESSION merekomendasikan pengkodean LZO untuk kolom VENUENAME. ANALISIS KOMPRESI memilih kompresi optimal berdasarkan beberapa faktor, yang mencakup persen pengurangan. Dalam kasus khusus ini, BYTEDICT memberikan kompresi yang lebih baik, tetapi LZO juga menghasilkan kompresi yang lebih besar dari 90 persen.

```
analyze compression cartesian_venue;

Table          | Column     | Encoding | Est_reduction_pct
---------------+------------+----------+------------------
reallybigvenue | venueid    | lzo      | 97.54
reallybigvenue | venuename  | lzo      | 91.71
reallybigvenue | venuecity  | lzo      | 96.01
reallybigvenue | venuestate | lzo      | 97.68
reallybigvenue | venueseats | lzo      | 98.21
```

Contoh

Contoh berikut membuat tabel CUSTOMER yang memiliki kolom dengan berbagai tipe data. Pernyataan CREATE TABLE ini menunjukkan salah satu dari banyak kemungkinan kombinasi pengkodean kompresi untuk kolom ini.

```
create table customer(
custkey int encode delta,
custname varchar(30) encode raw,
gender varchar(7) encode text255,
address varchar(200) encode text255,
city varchar(30) encode text255,
state char(2) encode raw,
zipcode char(5) encode bytedict,
start_date date encode delta32k);
```

Tabel berikut menunjukkan pengkodean kolom yang dipilih untuk tabel CUSTOMER dan memberikan penjelasan untuk pilihan:

Kolom	Jenis data	Pengodean	Penjelasan
CUSTKEY	int	delta	CUSTKEY terdiri dari nilai integer yang unik dan berurutan . Karena perbedaan nya satu byte, DELTA adalah pilihan yang baik.
NAMA PELANGGAN	bangkai (30)	mentah	CUSTNAME memiliki domain besar dengan beberapa nilai berulang. Pengkodea n kompresi apa pun mungkin tidak efektif.
JENIS KELAMIN	Washer (7)	teks255	GENDER adalah domain yang sangat kecil dengan banyak

Kolom	Jenis data	Pengodean	Penjelasan
			nilai berulang. Text255 bekerja dengan baik dengan kolom VARCHAR di mana kata yang sama berulang.
ALAMAT	bangkai (200)	teks255	ADDRESS adalah domain besar, tetapi berisi banyak kata berulang, seperti Street, Avenue, North, South, dan sebagainya. Teks 255 dan teks 32k berguna untuk mengompresi kolom VARCHAR di mana kata yang sama berulang. Panjang kolom pendek, jadi text255 adalah pilihan yang baik.
KOTA	bangkai (30)	teks255	CITY adalah domain besar, dengan beberapa nilai berulang. Nama-nama kota tertentu digunakan jauh lebih umum daripada yang lain. Text255 adalah pilihan yang baik untuk alasan yang sama seperti ADDRESS.

Kolom	Jenis data	Pengodean	Penjelasan
STATE	haru (2)	mentah	Di Amerika Serikat, STATE adalah domain yang tepat dari 50 nilai dua karakter. Pengkodea n bytedict akan menghasilkan beberapa kompresi, tetapi karena ukuran kolom hanya dua karakter, kompresi mungkin tidak sebanding dengan biaya tambahan untuk membuka kompresi data.
KODE POS	haru (5)	bytedict	ZIPCODE adalah domain yang diketahui kurang dari 50.000 nilai unik. Kode pos tertentu terjadi jauh lebih umum daripada yang lain. Pengkodean Bytedict sangat efektif ketika kolom berisi sejumlah nilai unik yang terbatas.

Kolom	Jenis data	Pengodean	Penjelasan
TANGGAL MULAI	date	delta32k	Pengkodean delta sangat berguna untuk kolom waktu tanggal, terutama jika baris dimuat dalam urutan tanggal.

Distribusi data untuk optimasi kueri

Saat Anda memuat data ke dalam tabel, Amazon Redshift mendistribusikan baris tabel ke masing-masing node komputasi sesuai dengan gaya distribusi tabel. Saat Anda menjalankan kueri, pengoptimal kueri mendistribusikan ulang baris ke node komputasi sesuai kebutuhan untuk melakukan gabungan dan agregasi apa pun. Tujuan dalam memilih gaya distribusi tabel adalah untuk meminimalkan dampak dari langkah redistribusi dengan menemukan data di tempat yang harus sebelum query dijalankan.

> **Note**
>
> Bagian ini akan memperkenalkan Anda pada prinsip-prinsip distribusi data dalam database Amazon Redshift. Kami menyarankan Anda membuat tabel Anda dengan`DISTSTYLE AUTO`. Jika Anda melakukannya, maka Amazon Redshift menggunakan pengoptimalan tabel otomatis untuk memilih gaya distribusi data. Untuk informasi selengkapnya, lihat Pengoptimalan tabel otomatis. Sisa bagian ini memberikan rincian tentang gaya distribusi.

Topik

- Konsep distribusi data
- Gaya distribusi
- Melihat gaya distribusi
- Mengevaluasi pola kueri
- Menunjuk gaya distribusi
- Mengevaluasi rencana kueri

- Contoh rencana kueri
- Contoh distribusi

Konsep distribusi data

Beberapa konsep distribusi data untuk Amazon Redshift berikut.

Node dan irisan

Cluster Amazon Redshift adalah sekumpulan node. Setiap node dalam cluster memiliki sistem operasi sendiri, memori khusus, dan penyimpanan disk khusus. Satu node adalah node pemimpin, yang mengelola distribusi data dan tugas pemrosesan kueri ke node komputasi. Node komputasi menyediakan sumber daya untuk melakukan tugas-tugas tersebut.

Penyimpanan disk untuk node komputasi dibagi menjadi sejumlah irisan. Jumlah irisan per node tergantung pada ukuran node cluster. Semua node berpartisipasi dalam menjalankan kueri paralel, mengerjakan data yang didistribusikan serata mungkin di seluruh irisan. Untuk informasi selengkapnya tentang jumlah irisan yang dimiliki setiap ukuran node, lihat Tentang cluster dan node di Panduan Manajemen Amazon Redshift.

Redistribusi data

Saat Anda memuat data ke dalam tabel, Amazon Redshift mendistribusikan baris tabel ke setiap irisan node sesuai dengan gaya distribusi tabel. Sebagai bagian dari rencana kueri, pengoptimal menentukan di mana blok data harus ditempatkan untuk menjalankan kueri dengan sebaik-baiknya. Data kemudian dipindahkan secara fisik, atau didistribusikan kembali, sementara kueri berjalan. Redistribusi mungkin melibatkan pengiriman baris tertentu ke node untuk bergabung atau menyiarkan seluruh tabel ke semua node.

Redistribusi data dapat menyumbang sebagian besar biaya rencana kueri, dan lalu lintas jaringan yang dihasilkannya dapat mempengaruhi operasi database lain dan memperlambat kinerja sistem secara keseluruhan. Sejauh Anda mengantisipasi tempat terbaik untuk menemukan data pada awalnya, Anda dapat meminimalkan dampak redistribusi data.

Tujuan distribusi data

Saat memuat data ke dalam tabel, Amazon Redshift mendistribusikan baris tabel ke node komputasi dan irisan sesuai dengan gaya distribusi yang Anda pilih saat membuat tabel. Distribusi data memiliki dua tujuan utama:

- Untuk mendistribusikan beban kerja secara seragam di antara node dalam cluster. Distribusi yang tidak merata, atau distribusi data miring, memaksa beberapa node melakukan lebih banyak pekerjaan daripada yang lain, yang mengganggu kinerja kueri.
- Untuk meminimalkan pergerakan data saat kueri berjalan. Jika baris yang berpartisipasi dalam gabungan atau agregat sudah dikolokasikan pada node dengan baris penggabungannya di tabel lain, pengoptimal tidak perlu mendistribusikan kembali data sebanyak itu saat kueri dijalankan.

Strategi distribusi yang Anda pilih untuk database Anda memiliki konsekuensi penting untuk kinerja kueri, persyaratan penyimpanan, pemuatan data, dan pemeliharaan. Dengan memilih gaya distribusi terbaik untuk setiap tabel, Anda dapat menyeimbangkan distribusi data Anda dan secara signifikan meningkatkan kinerja sistem secara keseluruhan.

Gaya distribusi

Saat Anda membuat tabel, Anda dapat menentukan salah satu gaya distribusi berikut: AUTO, EVEN, KEY, atau ALL.

Jika Anda tidak menentukan gaya distribusi, Amazon Redshift menggunakan distribusi AUTO.

Distribusi AUTO

Dengan distribusi AUTO, Amazon Redshift menetapkan gaya distribusi optimal berdasarkan ukuran data tabel. Misalnya, jika gaya distribusi AUTO ditentukan, Amazon Redshift awalnya menetapkan gaya distribusi ALL ke tabel kecil. Saat tabel bertambah besar, Amazon Redshift mungkin mengubah gaya distribusi menjadi KEY, memilih kunci utama (atau kolom kunci primer komposit) sebagai kunci distribusi. Jika tabel bertambah besar dan tidak ada kolom yang cocok untuk menjadi kunci distribusi, Amazon Redshift mengubah gaya distribusi menjadi EVEN. Perubahan gaya distribusi terjadi di latar belakang dengan dampak minimal terhadap kueri pengguna.

Untuk melihat tindakan yang dilakukan Amazon Redshift secara otomatis untuk mengubah kunci distribusi tabel, lihatSVL_AUTO_WORKER_ACTION. Untuk melihat rekomendasi saat ini mengenai mengubah kunci distribusi tabel, lihatSVV_ALTER_TABLE_REKOMENDASI.

Untuk melihat gaya distribusi yang diterapkan pada tabel, kueri tampilan katalog sistem PG_CLASS_INFO. Untuk informasi selengkapnya, lihat Melihat gaya distribusi. Jika Anda tidak menentukan gaya distribusi dengan pernyataan CREATE TABLE, Amazon Redshift menerapkan distribusi AUTO.

Distribusi EVEN

Node pemimpin mendistribusikan baris di seluruh irisan dengan cara round-robin, terlepas dari nilai dalam kolom tertentu. Distribusi EVEN sesuai ketika tabel tidak berpartisipasi dalam gabungan. Ini juga tepat ketika tidak ada pilihan yang jelas antara distribusi KEY dan distribusi SEMUA.

Distribusi KUNCI

Baris didistribusikan sesuai dengan nilai dalam satu kolom. Node pemimpin menempatkan nilai yang cocok pada irisan simpul yang sama. Jika Anda mendistribusikan sepasang tabel pada kunci penggabungan, node pemimpin mengkolokasikan baris pada irisan sesuai dengan nilai-nilai di kolom penggabungan. Dengan cara ini, nilai yang cocok dari kolom umum secara fisik disimpan bersama.

SEMUA distribusi

Salinan seluruh tabel didistribusikan ke setiap node. Di mana distribusi EVEN atau distribusi KEY hanya menempatkan sebagian dari baris tabel pada setiap node, distribusi ALL memastikan bahwa setiap baris dikolokasikan untuk setiap gabungan yang diikuti tabel.

Distribusi ALL mengalikan penyimpanan yang diperlukan dengan jumlah node dalam cluster, sehingga dibutuhkan waktu lebih lama untuk memuat, memperbarui, atau menyisipkan data ke beberapa tabel. Distribusi ALL hanya sesuai untuk tabel yang bergerak relatif lambat; yaitu tabel yang tidak sering diperbarui atau ekstensif. Karena biaya mendistribusikan ulang tabel kecil selama kueri rendah, tidak ada manfaat signifikan untuk mendefinisikan tabel dimensi kecil sebagai DISTSTYLE ALL.

> **Note**
>
> Setelah Anda menentukan gaya distribusi untuk kolom, Amazon Redshift menangani distribusi data di tingkat cluster. Amazon Redshift tidak memerlukan atau mendukung konsep partisi data dalam objek database. Anda tidak perlu membuat ruang tabel atau menentukan skema partisi untuk tabel.

Dalam skenario tertentu, Anda dapat mengubah gaya distribusi tabel setelah dibuat. Untuk informasi selengkapnya, lihat ALTER TABLE. Untuk skenario ketika Anda tidak dapat mengubah gaya distribusi tabel setelah dibuat, Anda dapat membuat ulang tabel dan mengisi tabel baru dengan salinan mendalam. Untuk informasi selengkapnya, lihat Melakukan salinan mendalam

Melihat gaya distribusi

Untuk melihat gaya distribusi tabel, kueri tampilan PG_CLASS_INFO atau tampilan SVV_TABLE_INFO.

Kolom RELEFFECTIVEDISTSTYLE di PG_CLASS_INFO menunjukkan gaya distribusi saat ini untuk tabel. Jika tabel menggunakan distribusi otomatis, RELEFFECTIVEDISTSTYLE adalah 10, 11, atau 12, yang menunjukkan apakah gaya distribusi efektif adalah AUTO (ALL), AUTO (EVEN), atau AUTO (KEY). Jika tabel menggunakan distribusi otomatis, gaya distribusi mungkin awalnya menampilkan AUTO (ALL), lalu ubah ke AUTO (EVEN) atau AUTO (KEY) saat tabel tumbuh.

Tabel berikut memberikan gaya distribusi untuk setiap nilai di kolom RELEFFECTIVEDISTSTYLE:

RELEFFECTIVEDISTSTYLE	Gaya distribusi saat ini
0	BAHKAN
1	KUNCI
8	SEMUA
10	OTOMATIS (SEMUA)
11	OTOMATIS (GENAP)
12	OTOMATIS (KUNCI)

Kolom DISTSTYLE di SVV_TABLE_INFO menunjukkan gaya distribusi saat ini untuk tabel. Jika tabel menggunakan distribusi otomatis, DISTSTYLE adalah AUTO (ALL), AUTO (EVEN), atau AUTO (KEY).

Contoh berikut membuat empat tabel menggunakan tiga gaya distribusi dan distribusi otomatis, kemudian menanyakan SVV_TABLE_INFO untuk melihat gaya distribusi.

```
create table public.dist_key (col1 int)
diststyle key distkey (col1);

insert into public.dist_key values (1);
```

```
create table public.dist_even (col1 int)
diststyle even;

insert into public.dist_even values (1);

create table public.dist_all (col1 int)
diststyle all;

insert into public.dist_all values (1);

create table public.dist_auto (col1 int);

insert into public.dist_auto values (1);

select "schema", "table", diststyle from SVV_TABLE_INFO
where "table" like 'dist%';

        schema   |    table        | diststyle
     ------------+-----------------+------------
      public     | dist_key        | KEY(col1)
      public     | dist_even       | EVEN
      public     | dist_all        | ALL
      public     | dist_auto       | AUTO(ALL)
```

Mengevaluasi pola kueri

Memilih gaya distribusi hanyalah salah satu aspek dari desain database. Pertimbangkan gaya distribusi dalam konteks keseluruhan sistem, menyeimbangkan distribusi dengan faktor-faktor penting lainnya seperti ukuran cluster, metode pengkodean kompresi, kunci pengurutan, dan batasan tabel.

Uji sistem Anda dengan data yang sedekat mungkin dengan data nyata.

Untuk membuat pilihan yang baik untuk gaya distribusi, Anda harus memahami pola kueri untuk aplikasi Amazon Redshift Anda. Identifikasi kueri yang paling mahal di sistem Anda dan mendasarkan desain database awal Anda pada tuntutan kueri tersebut. Faktor-faktor yang menentukan total biaya kueri termasuk berapa lama kueri berjalan dan berapa banyak sumber daya komputasi yang dikonsumsi. Faktor lain yang menentukan biaya kueri adalah seberapa sering dijalankan, dan seberapa mengganggu kueri dan operasi database lainnya.

Identifikasi tabel yang digunakan oleh kueri paling mahal, dan evaluasi perannya dalam runtime kueri. Pertimbangkan bagaimana tabel digabungkan dan digabungkan.

Gunakan pedoman di bagian ini untuk memilih gaya distribusi untuk setiap tabel. Setelah Anda melakukannya, buat tabel dan muat dengan data yang sedekat mungkin dengan data nyata. Kemudian uji tabel untuk jenis kueri yang Anda harapkan untuk digunakan. Anda dapat mengevaluasi kueri menjelaskan rencana untuk mengidentifikasi peluang penyetelan. Bandingkan waktu muat, ruang penyimpanan, dan runtime kueri untuk menyeimbangkan persyaratan keseluruhan sistem Anda.

Menunjuk gaya distribusi

Pertimbangan dan rekomendasi untuk menunjuk gaya distribusi di bagian ini menggunakan skema bintang sebagai contoh. Desain database Anda mungkin didasarkan pada skema bintang, beberapa varian skema bintang, atau skema yang sama sekali berbeda. Amazon Redshift dirancang untuk bekerja secara efektif dengan desain skema apa pun yang Anda pilih. Prinsip-prinsip dalam bagian ini dapat diterapkan pada skema desain apa pun.

1. Tentukan kunci utama dan kunci asing untuk semua tabel Anda.

   Amazon Redshift tidak menerapkan batasan kunci utama dan kunci asing, tetapi pengoptimal kueri menggunakannya saat menghasilkan rencana kueri. Jika Anda menetapkan kunci utama dan kunci asing, aplikasi Anda harus mempertahankan validitas kunci.

2. Bagikan tabel fakta dan tabel dimensi terbesarnya pada kolom umum mereka.

   Pilih dimensi terbesar berdasarkan ukuran dataset yang berpartisipasi dalam gabungan yang paling umum, bukan hanya ukuran tabel. Jika tabel biasanya difilter, menggunakan klausa WHERE, hanya sebagian dari barisnya yang berpartisipasi dalam gabungan. Tabel seperti itu memiliki dampak yang lebih kecil pada redistribusi daripada tabel yang lebih kecil yang menyumbang lebih banyak data. Tentukan kunci utama tabel dimensi dan kunci asing yang sesuai tabel fakta sebagai DISTKEY. Jika beberapa tabel menggunakan kunci distribusi yang sama, mereka juga dikolokasikan dengan tabel fakta. Tabel fakta Anda hanya dapat memiliki satu kunci distribusi. Setiap tabel yang bergabung pada kunci lain tidak dikolokasikan dengan tabel fakta.

3. Tentukan kunci distribusi untuk tabel dimensi lainnya.

   Bagikan tabel pada kunci utama atau kunci asing mereka, tergantung pada bagaimana mereka paling sering bergabung dengan tabel lain.

4. Mengevaluasi apakah akan mengubah beberapa tabel dimensi untuk menggunakan distribusi SEMUA.

Jika tabel dimensi tidak dapat dikolokasikan dengan tabel fakta atau tabel penggabungan penting lainnya, Anda dapat meningkatkan kinerja kueri secara signifikan dengan mendistribusikan seluruh tabel ke semua node. Menggunakan distribusi ALL melipatgandakan kebutuhan ruang penyimpanan dan meningkatkan waktu pemuatan dan operasi pemeliharaan, jadi Anda harus mempertimbangkan semua faktor sebelum memilih distribusi SEMUA. Bagian berikut menjelaskan cara mengidentifikasi kandidat untuk distribusi ALL dengan mengevaluasi rencana EXPLAIN.

5. Gunakan distribusi AUTO untuk tabel yang tersisa.

   Jika tabel sebagian besar didenormalisasi dan tidak berpartisipasi dalam gabungan, atau jika Anda tidak memiliki pilihan yang jelas untuk gaya distribusi lain, gunakan distribusi AUTO.

Untuk membiarkan Amazon Redshift memilih gaya distribusi yang sesuai, jangan secara eksplisit menentukan gaya distribusi.

Mengevaluasi rencana kueri

Anda dapat menggunakan rencana kueri untuk mengidentifikasi kandidat untuk mengoptimalkan gaya distribusi.

Setelah membuat keputusan desain awal Anda, buat tabel Anda, muat dengan data, dan uji. Gunakan kumpulan data uji yang sedekat mungkin dengan data nyata. Ukur waktu muat untuk digunakan sebagai garis dasar untuk perbandingan.

Evaluasi kueri yang mewakili kueri paling mahal yang Anda harapkan untuk dijalankan, khususnya kueri yang menggunakan gabungan dan agregasi. Bandingkan runtime untuk berbagai opsi desain. Saat Anda membandingkan waktu proses, jangan hitung kali kueri dijalankan pertama kali, karena runtime pertama menyertakan waktu kompilasi.

DS_DIST_NONE

Tidak diperlukan redistribusi, karena irisan yang sesuai dikolokasikan pada node komputasi. Anda biasanya hanya memiliki satu langkah DS_DIST_NONE, gabungan antara tabel fakta dan satu tabel dimensi.

DS_DIST_ALL_NONE

Tidak diperlukan redistribusi, karena tabel gabungan bagian dalam menggunakan DISTSTYLE ALL. Seluruh tabel terletak di setiap node.

DS_DIST_INNER

Tabel bagian dalam didistribusikan kembali.

DS_DIST_LUAR

Tabel luar didistribusikan kembali.

DS_BCAST_INNER

Salinan dari seluruh tabel bagian dalam disiarkan ke semua node komputasi.

DS_DIST_ALL_INNER

Seluruh tabel bagian dalam didistribusikan kembali ke satu irisan karena tabel luar menggunakan DISTSTYLE ALL.

DS_DIST_KEDUANYA

Kedua tabel didistribusikan kembali.

DS_DIST_ERR

Ketika tabel tidak memiliki gaya distribusi yang dipilih.

DS_DIST_NONE dan DS_DIST_ALL_NONE bagus. Mereka menunjukkan bahwa tidak ada distribusi yang diperlukan untuk langkah itu karena semua gabungan dikolokasikan.

DS_DIST_INNER berarti bahwa langkah tersebut mungkin memiliki biaya yang relatif tinggi karena tabel bagian dalam sedang didistribusikan kembali ke node. DS_DIST_INNER menunjukkan bahwa tabel luar sudah didistribusikan dengan benar pada kunci gabungan. Atur kunci distribusi tabel bagian dalam ke kunci gabungan untuk mengonversinya menjadi DS_DIST_NONE. Dalam beberapa kasus, mendistribusikan tabel dalam pada kunci gabungan tidak mungkin karena tabel luar tidak didistribusikan pada kunci gabungan. Jika ini masalahnya, evaluasi apakah akan menggunakan distribusi ALL untuk tabel bagian dalam. Jika tabel tidak sering diperbarui atau ekstensif, dan cukup besar untuk membawa biaya redistribusi yang tinggi, ubah gaya distribusi ke SEMUA dan uji lagi. Distribusi SEMUA menyebabkan peningkatan waktu muat, jadi ketika Anda menguji ulang, sertakan waktu muat dalam faktor evaluasi Anda.

DS_DIST_ALL_INNER tidak bagus. Ini berarti bahwa seluruh tabel dalam didistribusikan kembali ke satu irisan karena tabel luar menggunakan DISTSTYLE ALL, sehingga salinan dari seluruh tabel luar terletak di setiap node. Hal ini menghasilkan runtime serial yang tidak efisien dari gabungan pada satu node, alih-alih memanfaatkan runtime paralel menggunakan semua node. DISTSTYLE ALL

dimaksudkan untuk digunakan hanya untuk tabel gabungan dalam. Sebagai gantinya, tentukan kunci distribusi atau gunakan distribusi genap untuk tabel luar.

DS_BCAST_INNER dan DS_DIST_BOTH tidak bagus. Biasanya redistribusi ini terjadi karena tabel tidak bergabung pada kunci distribusinya. Jika tabel fakta belum memiliki kunci distribusi, tentukan kolom penggabungan sebagai kunci distribusi untuk kedua tabel. Jika tabel fakta sudah memiliki kunci distribusi di kolom lain, evaluasi apakah mengubah kunci distribusi untuk mencolokasikan gabungan ini meningkatkan kinerja secara keseluruhan. Jika mengubah kunci distribusi tabel luar bukanlah pilihan yang optimal, Anda dapat mencapai kolokasi dengan menentukan DISTSTYLE ALL untuk tabel bagian dalam.

Contoh berikut menunjukkan sebagian dari rencana kueri dengan label DS_BCAST_INNER dan DS_DIST_NONE.

```
->  XN Hash Join DS_BCAST_INNER  (cost=112.50..3272334142.59 rows=170771 width=84)
        Hash Cond: ("outer".venueid = "inner".venueid)
        ->  XN Hash Join DS_BCAST_INNER  (cost=109.98..3167290276.71 rows=172456
 width=47)
              Hash Cond: ("outer".eventid = "inner".eventid)
              ->  XN Merge Join DS_DIST_NONE  (cost=0.00..6286.47 rows=172456 width=30)
                    Merge Cond: ("outer".listid = "inner".listid)
                    ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497
 width=14)
                    ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=24)
```

Setelah mengubah tabel dimensi untuk menggunakan DISTSTYLE ALL, rencana kueri untuk kueri yang sama menunjukkan DS_DIST_ALL_NONE sebagai pengganti DS_BCAST_INNER. Juga, ada perubahan dramatis dalam biaya relatif untuk langkah-langkah gabungan. Total biaya `14142.59` dibandingkan `3272334142.59` dengan kueri sebelumnya.

```
->  XN Hash Join DS_DIST_ALL_NONE  (cost=112.50..14142.59 rows=170771 width=84)
        Hash Cond: ("outer".venueid = "inner".venueid)
        ->  XN Hash Join DS_DIST_ALL_NONE  (cost=109.98..10276.71 rows=172456 width=47)
              Hash Cond: ("outer".eventid = "inner".eventid)
              ->  XN Merge Join DS_DIST_NONE  (cost=0.00..6286.47 rows=172456 width=30)
                    Merge Cond: ("outer".listid = "inner".listid)
                    ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497
 width=14)
                    ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=24)
```

Contoh rencana kueri

Contoh ini menunjukkan bagaimana mengevaluasi rencana kueri untuk menemukan peluang untuk mengoptimalkan distribusi.

Jalankan query berikut dengan perintah EXPLAIN untuk menghasilkan rencana kueri.

```
explain
select lastname, catname, venuename, venuecity, venuestate, eventname,
month, sum(pricepaid) as buyercost, max(totalprice) as maxtotalprice
from category join event on category.catid = event.catid
join venue on venue.venueid = event.venueid
join sales on sales.eventid = event.eventid
join listing on sales.listid = listing.listid
join date on sales.dateid = date.dateid
join users on users.userid = sales.buyerid
group by lastname, catname, venuename, venuecity, venuestate, eventname, month
having sum(pricepaid)>9999
order by catname, buyercost desc;
```

Dalam database TICKIT, SALES adalah tabel fakta dan LISTING adalah dimensi terbesarnya. Untuk mengkolokasikan tabel, SALES didistribusikan pada LISTID, yang merupakan kunci asing untuk LISTING, dan LISTING didistribusikan pada kunci utamanya, LISTID. Contoh berikut menunjukkan perintah CREATE TABLE untuk SALES dan LISTING.

```
create table sales(
 salesid integer not null,
 listid integer not null distkey,
 sellerid integer not null,
 buyerid integer not null,
 eventid integer not null encode mostly16,
 dateid smallint not null,
 qtysold smallint not null encode mostly8,
 pricepaid decimal(8,2) encode delta32k,
 commission decimal(8,2) encode delta32k,
 saletime timestamp,
 primary key(salesid),
 foreign key(listid) references listing(listid),
 foreign key(sellerid) references users(userid),
 foreign key(buyerid) references users(userid),
 foreign key(dateid) references date(dateid))
        sortkey(listid,sellerid);
```

```
create table listing(
 listid integer not null distkey sortkey,
 sellerid integer not null,
 eventid integer not null encode mostly16,
 dateid smallint not null,
 numtickets smallint not null encode mostly8,
 priceperticket decimal(8,2) encode bytedict,
 totalprice decimal(8,2) encode mostly32,
 listtime timestamp,
 primary key(listid),
 foreign key(sellerid) references users(userid),
 foreign key(eventid) references event(eventid),
 foreign key(dateid) references date(dateid));
```

Dalam rencana kueri berikut, langkah Gabungkan Gabung untuk bergabung di SALES dan LISTING menampilkan DS_DIST_NONE, yang menunjukkan bahwa tidak diperlukan redistribusi untuk langkah tersebut. Namun, menaikkan rencana kueri, gabungan dalam lainnya menunjukkan DS_BCAST_INNER, yang menunjukkan bahwa tabel bagian dalam disiarkan sebagai bagian dari eksekusi kueri. Karena hanya satu pasang tabel yang dapat dikolokasikan menggunakan distribusi kunci, lima tabel harus disiarkan ulang.

```
QUERY PLAN
XN Merge  (cost=1015345167117.54..1015345167544.46 rows=1000 width=103)
  Merge Key: category.catname, sum(sales.pricepaid)
  ->  XN Network  (cost=1015345167117.54..1015345167544.46 rows=170771 width=103)
        Send to leader
        ->  XN Sort  (cost=1015345167117.54..1015345167544.46 rows=170771 width=103)
              Sort Key: category.catname, sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=15345150568.37..15345152276.08 rows=170771
 width=103)
                    Filter: (sum(pricepaid) > 9999.00)
                     ->  XN Hash Join DS_BCAST_INNER  (cost=742.08..15345146299.10
 rows=170771 width=103)
                           Hash Cond: ("outer".catid = "inner".catid)
                           ->  XN Hash Join DS_BCAST_INNER
 (cost=741.94..15342942456.61 rows=170771 width=97)
                                 Hash Cond: ("outer".dateid = "inner".dateid)
                                 ->  XN Hash Join DS_BCAST_INNER
 (cost=737.38..15269938609.81 rows=170766 width=90)
                                       Hash Cond: ("outer".buyerid = "inner".userid)
```

```
                                        ->  XN Hash Join DS_BCAST_INNER
(cost=112.50..3272334142.59 rows=170771 width=84)
                                              Hash Cond: ("outer".venueid =
"inner".venueid)
                                              ->  XN Hash Join DS_BCAST_INNER
(cost=109.98..3167290276.71 rows=172456 width=47)
                                                    Hash Cond: ("outer".eventid =
"inner".eventid)
                                                    ->  XN Merge Join DS_DIST_NONE
(cost=0.00..6286.47 rows=172456 width=30)
                                                          Merge Cond: ("outer".listid =
"inner".listid)
                                                          ->  XN Seq Scan on listing
(cost=0.00..1924.97 rows=192497 width=14)
                                                          ->  XN Seq Scan on sales
(cost=0.00..1724.56 rows=172456 width=24)
                                                    ->  XN Hash  (cost=87.98..87.98
rows=8798 width=25)
                                                          ->  XN Seq Scan on event
(cost=0.00..87.98 rows=8798 width=25)
                                              ->  XN Hash  (cost=2.02..2.02 rows=202
width=41)
                                                    ->  XN Seq Scan on venue
(cost=0.00..2.02 rows=202 width=41)
                                        ->  XN Hash  (cost=499.90..499.90 rows=49990
width=14)
                                              ->  XN Seq Scan on users
(cost=0.00..499.90 rows=49990 width=14)
                                  ->  XN Hash  (cost=3.65..3.65 rows=365 width=11)
                                        ->  XN Seq Scan on date  (cost=0.00..3.65
rows=365 width=11)
                              ->  XN Hash  (cost=0.11..0.11 rows=11 width=10)
                                    ->  XN Seq Scan on category  (cost=0.00..0.11 rows=11
width=10)
```

Salah satu solusinya adalah mengubah tabel agar memiliki DISTSTYLE ALL.

```
ALTER TABLE users ALTER DISTSTYLE ALL;
ALTER TABLE venue ALTER DISTSTYLE ALL;
ALTER TABLE category ALTER DISTSTYLE ALL;
ALTER TABLE date ALTER DISTSTYLE ALL;
ALTER TABLE event ALTER DISTSTYLE ALL;
```

Jalankan kueri yang sama dengan EXPLAIN lagi, dan periksa rencana kueri baru. Gabungan sekarang menunjukkan DS_DIST_ALL_NONE, menunjukkan bahwa tidak ada redistribusi yang diperlukan karena data didistribusikan ke setiap node menggunakan DISTSTYLE ALL.

```
QUERY PLAN
XN Merge  (cost=1000000047117.54..1000000047544.46 rows=1000 width=103)
  Merge Key: category.catname, sum(sales.pricepaid)
  ->  XN Network  (cost=1000000047117.54..1000000047544.46 rows=170771 width=103)
        Send to leader
        ->  XN Sort  (cost=1000000047117.54..1000000047544.46 rows=170771 width=103)
              Sort Key: category.catname, sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=30568.37..32276.08 rows=170771 width=103)
                    Filter: (sum(pricepaid) > 9999.00)
                    ->  XN Hash Join DS_DIST_ALL_NONE  (cost=742.08..26299.10
 rows=170771 width=103)
                          Hash Cond: ("outer".buyerid = "inner".userid)
                          ->  XN Hash Join DS_DIST_ALL_NONE  (cost=117.20..21831.99
 rows=170766 width=97)
                                Hash Cond: ("outer".dateid = "inner".dateid)
                                ->  XN Hash Join DS_DIST_ALL_NONE
 (cost=112.64..17985.08 rows=170771 width=90)
                                      Hash Cond: ("outer".catid = "inner".catid)
                                      ->  XN Hash Join DS_DIST_ALL_NONE
 (cost=112.50..14142.59 rows=170771 width=84)
                                            Hash Cond: ("outer".venueid =
 "inner".venueid)
                                            ->  XN Hash Join DS_DIST_ALL_NONE
 (cost=109.98..10276.71 rows=172456 width=47)
                                                  Hash Cond: ("outer".eventid =
 "inner".eventid)
                                                  ->  XN Merge Join DS_DIST_NONE
 (cost=0.00..6286.47 rows=172456 width=30)
                                                        Merge Cond: ("outer".listid =
 "inner".listid)
                                                        ->  XN Seq Scan on listing
 (cost=0.00..1924.97 rows=192497 width=14)
                                                        ->  XN Seq Scan on sales
 (cost=0.00..1724.56 rows=172456 width=24)
                                                  ->  XN Hash  (cost=87.98..87.98
 rows=8798 width=25)
                                                        ->  XN Seq Scan on event
 (cost=0.00..87.98 rows=8798 width=25)
```

```
                                          ->  XN Hash  (cost=2.02..2.02 rows=202
 width=41)
                                                ->  XN Seq Scan on venue
 (cost=0.00..2.02 rows=202 width=41)
                                    ->  XN Hash  (cost=0.11..0.11 rows=11 width=10)
                                          ->  XN Seq Scan on category
 (cost=0.00..0.11 rows=11 width=10)
                               ->  XN Hash  (cost=3.65..3.65 rows=365 width=11)
                                     ->  XN Seq Scan on date  (cost=0.00..3.65
 rows=365 width=11)
                          ->  XN Hash  (cost=499.90..499.90 rows=49990 width=14)
                                ->  XN Seq Scan on users  (cost=0.00..499.90 rows=49990
 width=14)
```

Contoh distribusi

Contoh berikut menunjukkan bagaimana data didistribusikan sesuai dengan opsi yang Anda tentukan dalam pernyataan CREATE TABLE.

Contoh DISTKEY

Lihatlah skema tabel USERS di database TICKIT. USERID didefinisikan sebagai kolom SORTKEY dan kolom DISTKEY:

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'users';

    column     |          type          | encoding | distkey | sortkey
---------------+------------------------+----------+---------+---------
 userid        | integer                | none     | t       |       1
 username      | character(8)           | none     | f       |       0
 firstname     | character varying(30)  | text32k  | f       |       0

...
```

USERID adalah pilihan yang baik untuk kolom distribusi pada tabel ini. Jika Anda menanyakan tampilan sistem SVV_DISKUSAGE, Anda dapat melihat bahwa tabel didistribusikan dengan sangat merata. Nomor kolom berbasis nol, jadi USERID adalah kolom 0.

```
select slice, col, num_values as rows, minvalue, maxvalue
from svv_diskusage
```

```
where name='users' and col=0 and rows>0
order by slice, col;

slice| col | rows  | minvalue | maxvalue
-----+-----+-------+----------+----------
0    | 0   | 12496 | 4        | 49987
1    | 0   | 12498 | 1        | 49988
2    | 0   | 12497 | 2        | 49989
3    | 0   | 12499 | 3        | 49990
(4 rows)
```

Tabel berisi 49.990 baris. Kolom baris (num_values) menunjukkan bahwa setiap irisan berisi jumlah baris yang hampir sama. Kolom minvalue dan maxvalue menunjukkan rentang nilai pada setiap irisan. Setiap irisan mencakup hampir seluruh rentang nilai, jadi ada kemungkinan besar bahwa setiap irisan berpartisipasi dalam menjalankan kueri yang memfilter berbagai ID pengguna.

Contoh ini menunjukkan distribusi pada sistem uji kecil. Jumlah total irisan biasanya jauh lebih tinggi.

Jika Anda biasanya bergabung atau mengelompokkan menggunakan kolom STATE, Anda dapat memilih untuk mendistribusikan di kolom STATE. Contoh berikut menunjukkan kasus di mana Anda membuat tabel baru dengan data yang sama dengan tabel USERS tetapi mengatur DISTKEY ke kolom STATE. Dalam hal ini, distribusinya tidak sama rata. Slice 0 (13.587 baris) menampung sekitar 30 persen lebih banyak baris daripada irisan 3 (10.150 baris). Dalam tabel yang jauh lebih besar, jumlah kemiringan distribusi ini dapat berdampak buruk pada pemrosesan kueri.

```
create table userskey distkey(state) as select * from users;

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'userskey' and col=0 and rows>0
order by slice, col;

slice | col | rows  | minvalue | maxvalue
------+-----+-------+----------+----------
    0 |   0 | 13587 |        5 |    49989
    1 |   0 | 11245 |        2 |    49990
    2 |   0 | 15008 |        1 |    49976
    3 |   0 | 10150 |        4 |    49986
(4 rows)
```

Contoh DISTSTYLE EVEN

Jika Anda membuat tabel baru dengan data yang sama dengan tabel USERS tetapi mengatur DISTSTYLE ke EVEN, baris selalu didistribusikan secara merata di seluruh irisan.

```
create table userseven diststyle even as
select * from users;

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'userseven' and col=0 and rows>0
order by slice, col;

slice | col | rows  | minvalue | maxvalue
------+-----+-------+----------+----------
    0 |   0 | 12497 |        4 |    49990
    1 |   0 | 12498 |        8 |    49984
    2 |   0 | 12498 |        2 |    49988
    3 |   0 | 12497 |        1 |    49989
(4 rows)
```

Namun, karena distribusi tidak didasarkan pada kolom tertentu, pemrosesan kueri dapat terdegradasi, terutama jika tabel digabungkan ke tabel lain. Kurangnya distribusi pada kolom penggabungan sering mempengaruhi jenis operasi gabungan yang dapat dilakukan secara efisien. Operasi gabungan, agregasi, dan pengelompokan dioptimalkan ketika kedua tabel didistribusikan dan diurutkan pada kolom penggabungannya masing-masing.

Contoh DISTSTYLE ALL

Jika Anda membuat tabel baru dengan data yang sama dengan tabel USERS tetapi mengatur DISTSTYLE ke ALL, semua baris didistribusikan ke irisan pertama dari setiap node.

```
select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'usersall' and col=0 and rows > 0
order by slice, col;

slice | col | rows  | minvalue | maxvalue
------+-----+-------+----------+----------
    0 |   0 | 49990 |        4 |    49990
    2 |   0 | 49990 |        2 |    49990

(4 rows)
```

Urutkan kunci

> **Note**
>
> Kami menyarankan Anda membuat tabel Anda dengan`SORTKEY AUTO`. Jika Anda melakukannya, maka Amazon Redshift menggunakan pengoptimalan tabel otomatis untuk memilih kunci pengurutan. Untuk informasi selengkapnya, lihat Pengoptimalan tabel otomatis. Sisa bagian ini memberikan rincian tentang urutan pengurutan.

Saat Anda membuat tabel, Anda dapat menentukan satu atau lebih kolom sebagai kunci pengurutan. Ketika data awalnya dimuat ke tabel kosong, baris disimpan pada disk dalam urutan yang diurutkan. Informasi tentang kolom kunci pengurutan diteruskan ke perencana kueri, dan perencana menggunakan informasi ini untuk membangun rencana yang memanfaatkan cara data diurutkan. Untuk informasi selengkapnya, lihat CREATE TABLE. Untuk informasi tentang praktik terbaik saat membuat kunci pengurutan, lihatPilih kunci pengurutan terbaik.

Penyortiran memungkinkan penanganan predikat terbatas rentang yang efisien. Amazon Redshift menyimpan data kolumnar dalam blok disk 1 MB. Nilai min dan max untuk setiap blok disimpan sebagai bagian dari metadata. Jika kueri menggunakan predikat terbatas rentang, prosesor kueri dapat menggunakan nilai min dan max untuk dengan cepat melewati sejumlah besar blok selama pemindaian tabel. Misalnya, sebuah tabel menyimpan lima tahun data yang diurutkan berdasarkan tanggal dan kueri menentukan rentang tanggal satu bulan. Dalam hal ini, Anda dapat menghapus hingga 98 persen blok disk dari pemindaian. Jika data tidak diurutkan, lebih banyak blok disk (mungkin semuanya) harus dipindai.

Anda dapat menentukan kunci pengurutan majemuk atau disisipkan. Kunci pengurutan majemuk lebih efisien ketika predikat kueri menggunakan awalan, yang merupakan subset dari kolom kunci pengurutan secara berurutan. Kunci pengurutan yang disisipkan memberikan bobot yang sama untuk setiap kolom dalam kunci pengurutan, sehingga predikat kueri dapat menggunakan subset kolom apa pun yang membentuk kunci pengurutan, dalam urutan apa pun.

Untuk memahami dampak kunci pengurutan yang dipilih pada kinerja kueri, gunakan EXPLAIN perintah. Untuk informasi selengkapnya, lihat Perencanaan kueri dan alur kerja pelaksanaan.

Untuk menentukan jenis pengurutan, gunakan kata kunci INTERLEAVED atau COMPOUND dengan pernyataan CREATE TABLE atau CREATE TABLE AS. Defaultnya adalah COMPOUND. COMPOUND direkomendasikan saat Anda memperbarui tabel secara teratur dengan operasi

INSERT, UPDATE, atau DELETE. Kunci pengurutan INTERLEAVED dapat menggunakan maksimal delapan kolom. Bergantung pada data dan ukuran cluster Anda, VACUUM REINDEX membutuhkan waktu lebih lama secara signifikan daripada VACUUM FULL karena membuat izin tambahan untuk menganalisis kunci pengurutan yang disisipkan. Operasi pengurutan dan penggabungan dapat memakan waktu lebih lama untuk tabel yang disisipkan karena pengurutan yang disisipkan mungkin harus mengatur ulang lebih banyak baris daripada pengurutan majemuk.

Untuk melihat kunci pengurutan untuk tabel, kueri SVV_TABLE_INFO tampilan sistem.

Topik

- Penyortiran tata letak data multidimensi
- Kunci pengurutan majemuk
- Kunci pengurutan yang disisipkan

Penyortiran tata letak data multidimensi

Kunci pengurutan tata letak data multidimensi adalah jenis kunci pengurutan OTOMATIS yang didasarkan pada predikat berulang yang ditemukan dalam beban kerja. Jika beban kerja Anda memiliki predikat berulang, maka Amazon Redshift dapat meningkatkan kinerja pemindaian tabel dengan menempatkan baris data yang memenuhi predikat berulang. Alih-alih menyimpan data tabel dalam urutan kolom yang ketat, tata letak data multidimensi mengurutkan kunci menyimpan data dengan menganalisis predikat berulang yang muncul dalam beban kerja. Lebih dari satu predikat berulang dapat ditemukan dalam beban kerja. Tergantung pada beban kerja Anda, kunci pengurutan semacam ini dapat meningkatkan kinerja banyak predikat. Amazon Redshift secara otomatis menentukan apakah metode kunci pengurutan ini harus digunakan untuk tabel yang ditentukan dengan kunci peng AUT0 urutan.

Misalnya, Anda memiliki tabel yang memiliki data yang diurutkan dalam urutan kolom. Banyak blok data mungkin perlu diperiksa untuk menentukan apakah mereka memenuhi predikat dalam beban kerja Anda. Tetapi, jika data disimpan pada disk dalam urutan predikat, maka lebih sedikit blok yang perlu dipindai untuk memenuhi permintaan. Menggunakan kunci pengurutan tata letak data multidimensi bermanfaat dalam kasus ini.

Untuk melihat apakah kueri menggunakan tombol tata letak data multidimensi, lihat `step_attribute` kolom SYS_QUERY_DETAIL tampilan. Ketika nilainya `multi-dimensional` kemudian tata letak data multidimensi digunakan untuk query.

Untuk mencegah Amazon Redshift menggunakan tombol pengurutan tata letak data multidimensi, pilih opsi kunci pengurutan tabel yang berbeda selain. `SORTKEY AUTO` Untuk informasi selengkapnya tentang opsi SORTKEY, lihat. CREATE TABLE

Kunci pengurutan majemuk

Kunci majemuk terdiri dari semua kolom yang tercantum dalam definisi kunci pengurutan, dalam urutan tercantum. Kunci pengurutan majemuk paling berguna ketika filter kueri menerapkan kondisi, seperti filter dan gabungan, yang menggunakan awalan kunci pengurutan. Manfaat kinerja penyortiran majemuk menurun ketika kueri hanya bergantung pada kolom pengurutan sekunder, tanpa mereferensikan kolom utama. COMPOUND adalah tipe pengurutan default.

Tombol pengurutan majemuk dapat mempercepat operasi join, GROUP BY dan ORDER BY, dan fungsi jendela yang menggunakan PARTITION BY dan ORDER BY. Misalnya, gabungan gabungan, yang seringkali lebih cepat daripada gabungan hash, layak ketika data didistribusikan dan diurutkan sebelumnya pada kolom penggabungan. Tombol pengurutan majemuk juga membantu meningkatkan kompresi.

Saat Anda menambahkan baris ke tabel yang diurutkan yang sudah berisi data, wilayah yang tidak disortir tumbuh, yang memiliki efek signifikan pada kinerja. Efeknya lebih besar ketika tabel menggunakan pengurutan yang disisipkan, terutama ketika kolom pengurutan menyertakan data yang meningkat secara monoton, seperti kolom tanggal atau stempel waktu. Jalankan operasi VACUUM secara teratur, terutama setelah pemuatan data yang besar, untuk mengurutkan ulang dan menganalisis ulang data. Untuk informasi selengkapnya, lihat Kurangi ukuran wilayah yang tidak disortir. Setelah menyedot debu untuk menggunakan data, sebaiknya jalankan perintah ANALYZE untuk memperbarui metadata statistik untuk perencana kueri. Untuk informasi selengkapnya, lihat Menganalisis tabel.

Kunci pengurutan yang disisipkan

Urutan yang disisipkan memberikan bobot yang sama untuk setiap kolom, atau subset kolom, dalam kunci pengurutan. Jika beberapa kueri menggunakan kolom yang berbeda untuk filter, maka Anda sering dapat meningkatkan kinerja untuk kueri tersebut dengan menggunakan gaya pengurutan yang disisipkan. Ketika kueri menggunakan predikat terbatas pada kolom pengurutan sekunder, penyortiran interleaved secara signifikan meningkatkan kinerja kueri dibandingkan dengan penyortiran majemuk.

> **Important**
>
> Jangan gunakan kunci pengurutan yang disisipkan pada kolom dengan atribut yang meningkat secara monoton, seperti kolom identitas, tanggal, atau stempel waktu.

Peningkatan kinerja yang Anda peroleh dengan menerapkan kunci pengurutan yang disisipkan harus ditimbang terhadap peningkatan waktu beban dan vakum.

Sortir yang disisipkan paling efektif dengan kueri yang sangat selektif yang memfilter satu atau lebih kolom kunci pengurutan dalam klausa WHERE, misalnya. `select c_name from customer where c_region = 'ASIA'` Manfaat penyortiran interleaved meningkat dengan jumlah kolom yang diurutkan yang dibatasi.

Urutan yang disisipkan lebih efektif dengan tabel besar. Penyortiran diterapkan pada setiap irisan. Dengan demikian, pengurutan yang disisipkan paling efektif ketika tabel cukup besar untuk membutuhkan beberapa blok 1 MB per irisan. Di sini, prosesor kueri dapat melewati sebagian besar blok menggunakan predikat terbatas. Untuk melihat jumlah blok yang digunakan tabel, kueri STV_BLOCKLIST tampilan sistem.

Saat menyortir pada satu kolom, pengurutan yang disisipkan mungkin memberikan kinerja yang lebih baik daripada pengurutan majemuk jika nilai kolom memiliki awalan umum yang panjang. Misalnya, URL biasanya dimulai dengan "http://www”. Kunci pengurutan majemuk menggunakan sejumlah karakter terbatas dari awalan, yang menghasilkan banyak duplikasi kunci. Sortir yang disisipkan menggunakan skema kompresi internal untuk nilai peta zona yang memungkinkannya membedakan dengan lebih baik di antara nilai-nilai kolom yang memiliki awalan umum yang panjang.

Saat memigrasikan cluster yang disediakan Amazon Redshift ke Amazon Redshift Serverless, Redshift mengonversi tabel dengan kunci pengurutan yang disisipkan dan KUNCI DISTSTYLE menjadi kunci pengurutan majemuk. Namun, tabel dengan hanya kunci pengurutan yang disisipkan tetap tidak berubah. Untuk informasi selengkapnya tentang gaya distribusi, lihat Bekerja dengan gaya distribusi data.

INDEKS ULANG VAKUM

Saat Anda menambahkan baris ke tabel yang diurutkan yang sudah berisi data, kinerja mungkin memburuk seiring waktu. Kerusakan ini terjadi untuk jenis majemuk dan yang disisipkan, tetapi memiliki efek yang lebih besar pada tabel yang disisipkan. VACUUM mengembalikan urutan

pengurutan, tetapi operasi dapat memakan waktu lebih lama untuk tabel yang disisipkan karena menggabungkan data interleaved baru mungkin melibatkan modifikasi setiap blok data.

Saat tabel dimuat pada awalnya, Amazon Redshift menganalisis distribusi nilai dalam kolom kunci pengurutan dan menggunakan informasi tersebut untuk penyisipan optimal kolom kunci pengurutan. Seiring bertambahnya tabel, distribusi nilai dalam kolom kunci pengurutan dapat berubah, atau miring, terutama dengan kolom tanggal atau stempel waktu. Jika kemiringan menjadi terlalu besar, kinerja mungkin terpengaruh. Untuk menganalisis ulang kunci pengurutan dan mengembalikan kinerja, jalankan perintah VACUUM dengan kata kunci REINDEX. Karena harus membutuhkan analisis ekstra untuk melewati data, VACUUM REINDEX dapat memakan waktu lebih lama daripada VACUUM standar untuk tabel yang disisipkan. Untuk melihat informasi tentang kemiringan distribusi kunci dan waktu indeks ulang terakhir, kueri tampilan SVV_KOLUMN_INTERLEAVED_ sistem.

Untuk informasi selengkapnya tentang cara menentukan seberapa sering menjalankan VACUUM dan kapan menjalankan VACUUM REINDEX, lihatPutuskan apakah akan mengindeks ulang.

Kendala tabel

Kendala keunikan, kunci utama, dan kunci asing hanya bersifat informasi; batasan tersebut tidak diberlakukan oleh Amazon Redshift saat Anda mengisi tabel. Misalnya, jika Anda memasukkan data ke dalam tabel dengan dependensi, penyisipan dapat berhasil meskipun melanggar batasan. Meskipun demikian, kunci utama dan kunci asing digunakan sebagai petunjuk perencanaan dan mereka harus dideklarasikan jika proses ETL Anda atau proses lain dalam aplikasi Anda menegakkan integritasnya.

Misalnya, perencana kueri menggunakan kunci primer dan asing dalam perhitungan statistik tertentu. Ini dilakukan untuk menyimpulkan keunikan dan hubungan referensial yang memengaruhi teknik dekorrelasi subkueri. Dengan melakukan ini, ia dapat memesan sejumlah besar gabungan dan menghapus gabungan yang berlebihan.

Perencana memanfaatkan hubungan kunci ini, tetapi mengasumsikan bahwa semua kunci di tabel Amazon Redshift valid saat dimuat. Jika aplikasi Anda mengizinkan kunci asing atau kunci utama yang tidak valid, beberapa kueri dapat mengembalikan hasil yang salah. Misalnya, kueri SELECT DISTINCT mungkin mengembalikan baris duplikat jika kunci primer tidak unik. Jangan mendefinisikan batasan utama untuk tabel Anda jika Anda meragukan validitasnya. Namun, selalu deklarasikan kunci primer dan asing serta kendala keunikan ketika Anda tahu bahwa kunci tersebut valid.

Amazon Redshift member lakukan batasan kolom NOT NULL.

Untuk informasi selengkapnya tentang batasan tabel, lihatCREATE TABLE. Untuk informasi tentang cara menjatuhkan tabel dengan dependensi, lihatMEJA DROP.

Memuat data di Amazon Redshift

Ada beberapa cara untuk memuat data ke database Amazon Redshift. Salah satu sumber data yang populer untuk dimuat adalah file Amazon S3. Tabel berikut merangkum beberapa metode yang digunakan dengan mulai dari sumber Amazon S3.

Metode untuk digunakan	Deskripsi	Ketika metode dibutuhkan
Perintah SALIN	Menjalankan penyerapan file batch untuk memuat data dari file Amazon S3 Anda. Metode ini memanfaatkan kemampuan pemrosesan paralel Amazon Redshift. Untuk informasi selengkapnya, lihat Memuat tabel dengan perintah COPY.	Harus digunakan ketika persyaratan pemuatan data dasar untuk memulai penyerapan file batch secara manual diperlukan. Metode ini sebagian besar digunakan dengan saluran penyerapan file kustom dan pihak ketiga atau beban kerja penyerapan file satu kali, atau ad hoc.
MENYALIN... Perintah CREATE JOB (salin otomatis)	Menjalankan perintah COPY Anda secara otomatis saat file baru dibuat di jalur Amazon S3 yang dilacak. Untuk informasi selengkapnya, lihat Buat integrasi acara S3 untuk menyalin file secara otomatis dari bucket Amazon S3.	Harus digunakan saat pipeline penyerapan file perlu menelan data secara otomatis saat file baru dibuat di Amazon S3. Amazon Redshift melacak file yang dicerna untuk mencegah duplikasi data. Metode ini memerlukan konfigurasi oleh pemilik bucket Amazon S3.
Muat dari kueri danau data	Buat tabel eksternal untuk menjalankan kueri data lake pada file Amazon S3 Anda, lalu jalankan perintah INSERT INTO untuk memuat hasil dari kueri data lake Anda ke tabel lokal. Untuk informasi	Harus digunakan dalam salah satu skenario berikut: • Memuat dari AWS Glue dan membuka format tabel (seperti Apache Iceberg, Apache Hudi, atau Delta Lake).

Metode untuk digunakan	Deskripsi	Ketika metode dibutuhkan
	selengkapnya, lihat Tabel eksternal untuk Redshift Spectrum.	• File sumber perlu dicerna sebagian (misalnya, diperlukan untuk menjalankan klausa WHERE untuk menelan baris tertentu). • Lebih banyak fleksibilitas diperlukan untuk menelan kolom tertentu (seperti menjalankan perintah SELECT) atau melakukan transformasi data dasar saat bepergian (seperti menerapkan operasi dasar atau memanggil UDF pada nilai dari file sumber).
Metode lain yang dapat Anda pertimbangkan		
Konsumsi streaming	Penyerapan streaming menyediakan penyerapan data aliran berkecepatan tinggi dengan latensi rendah dari Amazon Kinesis Data Streams dan Amazon Managed Streaming untuk Apache Kafka ke dalam tampilan terwujud Amazon Redshift atau Redshift Serverless. Untuk informasi selengkapnya, lihat Memulai penyerapan streaming dari Amazon Kinesis Data Streams dan Memulai dengan konsumsi streaming dari sumber Apache Kafka.	Harus dipertimbangkan untuk kasus penggunaan ketika data pertama kali dialirkan ke file di Amazon S3 dan kemudian dimuat dari Amazon S3. Jika menyimpan data di Amazon S3 tidak diperlukan, Anda sering dapat mempertimbangkan streaming data Anda langsung ke Amazon Redshift.

Metode untuk digunakan	Deskripsi	Ketika metode dibutuhkan
Menjalankan kueri danau data	Menjalankan kueri langsung dari tabel danau data alih-alih menelan konten tabel ke tabel lokal. Untuk informasi selengkapnya, lihat Spektrum Pergeseran Merah Amazon.	Harus digunakan ketika kasus penggunaan tidak memerlukan kinerja kueri tabel lokal di Amazon Redshift.
Pemuatan batch menggunakan editor kueri Amazon Redshift v2	Anda dapat menyiapkan dan menjalankan beban kerja penyerapan file batch secara visual di editor kueri Amazon Redshift v2. Untuk informasi selengkapnya, lihat Memuat data dari S3 di Panduan Manajemen Amazon Redshift.	Harus digunakan ketika Anda ingin editor kueri v2 menyiapkan pernyataan COPY dan Anda menginginkan alat visual untuk menyederhanakan proses persiapan pernyataan COPY.
Muat data dari file lokal menggunakan editor kueri Amazon Redshift v2	Anda dapat langsung mengunggah file dari desktop Anda ke tabel Amazon Redshift tanpa perlu mengunggah file Anda secara manual ke Amazon S3. Untuk informasi selengkapnya, lihat Memuat data dari pengaturan file lokal dan alur kerja di Panduan Manajemen Amazon Redshift.	Harus digunakan ketika Anda perlu memuat file dengan cepat dari komputer lokal Anda untuk tujuan kueri satu kali. Dengan metode ini, editor kueri Amazon Redshift v2 menyimpan file sementara di bucket Amazon S3 milik pelanggan dan menjalankan perintah salin menggunakan jalur Amazon S3 ini.

Perintah COPY adalah cara paling efisien untuk memuat tabel. Anda juga dapat menambahkan data ke tabel Anda menggunakan perintah INSERT, meskipun jauh kurang efisien daripada menggunakan COPY. Perintah COPY dapat membaca dari beberapa file data atau beberapa aliran data secara bersamaan. Amazon Redshift mengalokasikan beban kerja ke node Amazon Redshift dan melakukan operasi pemuatan secara paralel, termasuk menyortir baris dan mendistribusikan data di seluruh irisan node.

> **Note**
>
> Tabel eksternal Amazon Redshift Spectrum bersifat read-only. Anda tidak dapat menyalin atau menyisipkan ke tabel eksternal.

Untuk mengakses data pada AWS sumber daya lain, Amazon Redshift harus memiliki izin untuk mengakses sumber daya tersebut dan untuk melakukan tindakan yang diperlukan untuk mengakses data. Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk membatasi akses yang dimiliki pengguna ke sumber daya dan data Amazon Redshift.

Setelah data awal dimuat, jika Anda menambahkan, memodifikasi, atau menghapus sejumlah besar data, Anda harus menindaklanjuti dengan menjalankan perintah VACUUM untuk mengatur ulang data Anda dan merebut kembali ruang setelah dihapus. Anda juga harus menjalankan perintah ANALYZE untuk memperbarui statistik tabel.

Topik

- Memuat tabel dengan perintah COPY
- Buat integrasi acara S3 untuk menyalin file secara otomatis dari bucket Amazon S3
- Memuat tabel dengan perintah DML
- Melakukan salinan mendalam
- Menganalisis tabel
- Meja penyedot debu
- Mengelola operasi penulisan bersamaan
- Tutorial: Memuat data dari Amazon S3

Memuat tabel dengan perintah COPY

Perintah COPY memanfaatkan arsitektur pemrosesan paralel besar-besaran (MPP) Amazon Redshift untuk membaca dan memuat data secara paralel dari file di Amazon S3, dari tabel DynamoDB, atau dari output teks dari satu atau beberapa host jarak jauh.

Sebelum mempelajari semua opsi perintah COPY, kami sarankan mempelajari opsi dasar untuk memuat data Amazon S3. Panduan Memulai Amazon Redshift menunjukkan penggunaan sederhana perintah COPY untuk memuat data Amazon S3 menggunakan peran IAM default. Lihat Langkah 4: Muat data dari Amazon S3 ke Amazon Redshift untuk detailnya.

> **Note**
>
> Kami sangat menyarankan menggunakan perintah COPY untuk memuat data dalam jumlah besar. Menggunakan pernyataan INSERT individual untuk mengisi tabel mungkin sangat lambat. Atau, jika data Anda sudah ada di tabel database Amazon Redshift lainnya, gunakan INSERT INTO... PILIH atau BUAT TABEL AS untuk meningkatkan kinerja. Untuk informasi, lihat INSERT atauCREATE TABLE AS.

Untuk memuat data dari AWS sumber daya lain, Amazon Redshift harus memiliki izin untuk mengakses sumber daya dan melakukan tindakan yang diperlukan.

Untuk memberikan atau mencabut hak istimewa untuk memuat data ke dalam tabel menggunakan perintah COPY, berikan atau cabut hak istimewa INSERT.

Data Anda harus dalam format yang tepat untuk dimuat ke tabel Amazon Redshift Anda. Bagian ini menyajikan pedoman untuk menyiapkan dan memverifikasi data Anda sebelum memuat dan untuk memvalidasi pernyataan COPY sebelum Anda menjalankannya.

Untuk melindungi informasi dalam file Anda, Anda dapat mengenkripsi file data sebelum mengunggahnya ke bucket Amazon S3 Anda; COPY akan mendekripsi data saat melakukan pemuatan. Anda juga dapat membatasi akses ke data muat Anda dengan memberikan kredentif keamanan sementara kepada pengguna. KredenSIAL keamanan sementara memberikan keamanan yang ditingkatkan karena memiliki rentang hidup yang pendek dan tidak dapat digunakan kembali setelah kedaluwarsa.

Amazon Redshift memiliki fitur bawaan untuk COPY untuk memuat data yang tidak terkompresi dan dibatasi dengan cepat. Tetapi Anda dapat mengompres file Anda menggunakan gzip, lzop, atau bzip2 untuk menghemat waktu mengunggah file.

Jika kata kunci berikut ada dalam kueri COPY, pemisahan otomatis data yang tidak terkompresi tidak didukung: ESCAPE, REMOVEQUOTES, dan FIXEDWIDTH. Tetapi kata kunci CSV didukung.

Untuk membantu menjaga keamanan data Anda saat transit di dalam AWS Cloud, Amazon Redshift menggunakan SSL yang dipercepat perangkat keras untuk berkomunikasi dengan Amazon S3 atau Amazon DynamoDB untuk operasi COPY, UNLOAD, backup, dan restore.

Saat memuat tabel langsung dari tabel Amazon DynamoDB, Anda memiliki opsi untuk mengontrol jumlah throughput yang disediakan Amazon DynamoDB yang Anda konsumsi.

Anda dapat secara opsional membiarkan COPY menganalisis data input Anda dan secara otomatis menerapkan pengkodean kompresi optimal ke tabel Anda sebagai bagian dari proses pemuatan.

Topik

- Kredensi dan izin akses
- Mempersiapkan data input Anda
- Memuat data dari Amazon S3
- Memuat data dari Amazon EMR
- Memuat data dari host jarak jauh
- Memuat data dari tabel Amazon DynamoDB
- Memverifikasi bahwa data dimuat dengan benar
- Memvalidasi data masukan
- Memuat tabel dengan kompresi otomatis
- Mengoptimalkan penyimpanan untuk tabel sempit
- Memuat nilai kolom default
- Memecahkan masalah beban data

Kredensi dan izin akses

Untuk memuat atau membongkar data menggunakan AWS sumber daya lain, seperti Amazon S3, Amazon DynamoDB, Amazon EMR, atau Amazon EC2, Amazon Redshift harus memiliki izin untuk mengakses sumber daya dan melakukan tindakan yang diperlukan untuk mengakses data. Misalnya, untuk memuat data dari Amazon S3, COPY harus memiliki akses LIST ke bucket dan akses GET untuk objek bucket.

Untuk mendapatkan otorisasi untuk mengakses sumber daya, Amazon Redshift harus diautentikasi. Anda dapat memilih kontrol akses berbasis peran atau kontrol akses berbasis kunci. Bagian ini menyajikan ikhtisar dari dua metode. Untuk detail dan contoh lengkap, lihatIzin untuk mengakses lainnya AWS Sumber daya.

Role-based kontrol akses

Dengan kontrol akses berbasis peran, Amazon Redshift sementara mengambil peran AWS Identity and Access Management (IAM) atas nama Anda. Kemudian, berdasarkan otorisasi yang diberikan untuk peran tersebut, Amazon Redshift dapat mengakses sumber daya yang diperlukan AWS .

Sebaiknya gunakan kontrol akses berbasis peran karena memberikan kontrol akses yang lebih aman dan berbutir halus atas akses ke AWS sumber daya dan data pengguna yang sensitif, selain melindungi kredensibilitas Anda. AWS

Untuk menggunakan kontrol akses berbasis peran, Anda harus terlebih dahulu membuat peran IAM menggunakan jenis peran layanan Amazon Redshift, lalu melampirkan peran tersebut ke gudang data Anda. Peran harus memiliki, minimal, izin yang tercantum dilzin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY. Untuk langkah-langkah membuat peran IAM dan melampirkannya ke cluster Anda, lihat Membuat Peran IAM untuk Memungkinkan Cluster Amazon Redshift Anda Mengakses AWS Layanan di Panduan Manajemen Amazon Redshift.

Anda dapat menambahkan peran ke cluster atau melihat peran yang terkait dengan cluster menggunakan Amazon Redshift Management Console, CLI, atau API. Untuk informasi selengkapnya, lihat Men gotorisasi Operasi COPY dan UNLOAD Menggunakan Peran IAM di Panduan Manajemen Amazon Redshift.

Saat Anda membuat peran IAM, IAM mengembalikan Nama Sumber Daya Amazon (ARN) untuk peran tersebut. Untuk menjalankan perintah COPY menggunakan peran IAM, berikan peran ARN menggunakan parameter IAM_ROLE atau parameter CREDENTIALS.

Contoh perintah COPY berikut menggunakan parameter IAM_ROLE dengan peran untuk otentikasi. `MyRedshiftRole`

```
COPY customer FROM 's3://amzn-s3-demo-bucket/mydata'
IAM_ROLE 'arn:aws:iam::12345678901:role/MyRedshiftRole';
```

AWS Pengguna harus memiliki, minimal, izin yang tercantum dilzin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY.

Key-based kontrol akses

Dengan kontrol akses berbasis kunci, Anda memberikan ID kunci akses dan kunci akses rahasia untuk pengguna yang berwenang mengakses AWS sumber daya yang berisi data.

> **Note**
>
> Kami sangat menyarankan menggunakan peran IAM untuk otentikasi alih-alih memberikan ID kunci akses teks biasa dan kunci akses rahasia. Jika Anda memilih kontrol akses berbasis kunci, jangan pernah menggunakan kredentif AWS akun (root) Anda. Selalu buat pengguna

IAM dan berikan ID kunci akses dan kunci akses rahasia pengguna tersebut. Untuk langkah-langkah membuat pengguna IAM, lihat Membuat Pengguna IAM di Akun Anda AWS .

Mempersiapkan data input Anda

Jika data input Anda tidak kompatibel dengan kolom tabel yang akan menerimanya, perintah COPY akan gagal.

Gunakan panduan berikut untuk membantu memastikan bahwa data input Anda valid:

- Data Anda hanya dapat berisi UTF-8 karakter hingga empat byte panjang.
- Pastikan string CHAR dan VARCHAR tidak lebih panjang dari panjang kolom yang sesuai. String VARCHAR diukur dalam byte, bukan karakter, jadi, misalnya, string empat karakter karakter Mandarin yang masing-masing menempati empat byte memerlukan kolom VARCHAR (16).
- Karakter multibyte hanya dapat digunakan dengan kolom VARCHAR. Pastikan bahwa karakter multibyte tidak lebih dari empat byte panjangnya.
- Verifikasi bahwa data untuk kolom CHAR hanya berisi karakter byte tunggal.
- Jangan sertakan karakter khusus atau sintaks untuk menunjukkan bidang terakhir dalam catatan. Bidang ini bisa menjadi pembatas.
- Jika data Anda menyertakan terminator null, juga disebut sebagai NUL (UTF-8 0000) atau biner nol (0x000), Anda dapat memuat karakter ini sebagai NULLS ke kolom CHAR atau VARCHAR dengan menggunakan opsi NULL AS dalam perintah COPY: or. `null as '\0'` `null as '\000'` Jika Anda tidak menggunakan NULL AS, terminator null akan menyebabkan COPY Anda gagal.
- Jika string Anda berisi karakter khusus, seperti pembatas dan baris baru yang disematkan, gunakan opsi ESCAPE dengan perintah. MENYALIN
- Pastikan semua tanda kutip tunggal dan ganda cocok dengan tepat.
- Verifikasi bahwa string floating-point berada dalam format floating-point standar, seperti 12.123, atau format eksponensial, seperti 1.0E4.
- Verifikasi bahwa semua string stempel waktu dan tanggal mengikuti spesifikasi untuk. String DATEFORMAT dan TIMEFORMAT Format stempel waktu default adalah YYYY-MM-DD hh: mm: ss, dan format tanggal default adalah. YYYY-MM-DD
- Untuk informasi selengkapnya tentang batas dan batasan pada tipe data individual, lihatJenis Data. Untuk informasi tentang kesalahan karakter multibyte, lihat Kesalahan pemuatan karakter multibyte

Memuat data dari Amazon S3

Perintah COPY memanfaatkan arsitektur pemrosesan paralel besar-besaran (MPP) Amazon Redshift untuk membaca dan memuat data secara paralel dari file atau beberapa file dalam bucket Amazon S3. Anda dapat mengambil keuntungan maksimal dari pemrosesan paralel dengan membagi data Anda menjadi beberapa file, dalam kasus di mana file dikompresi. (Ada pengecualian untuk aturan ini. Ini dirinci dalam Mem uat file data.) Anda juga dapat memanfaatkan pemrosesan paralel secara maksimal dengan mengatur kunci distribusi pada tabel Anda. Untuk informasi selengkapnya tentang kunci distribusi, lihatDistribusi data untuk optimasi kueri.

Data dimuat ke tabel target, satu baris per baris. Bidang dalam file data dicocokkan dengan kolom tabel secara berurutan, kiri ke kanan. Bidang dalam file data dapat berupa lebar tetap atau dibatasi karakter; pembatas default adalah pipa (|). Secara default, semua kolom tabel dimuat, tetapi Anda dapat secara opsional menentukan daftar kolom yang dipisahkan koma. Jika kolom tabel tidak termasuk dalam daftar kolom yang ditentukan dalam perintah COPY, kolom tersebut dimuat dengan nilai default. Untuk informasi selengkapnya, lihat Memuat nilai kolom default.

Topik

- Memuat data dari file terkompresi dan tidak terkompresi
- Mengunggah file ke Amazon S3 untuk digunakan dengan COPY
- Menggunakan perintah COPY untuk memuat dari Amazon S3

Memuat data dari file terkompresi dan tidak terkompresi

Saat memuat data terkompresi, sebaiknya Anda membagi data untuk setiap tabel menjadi beberapa file. Saat Anda memuat data yang tidak terkompresi dan dibatasi, perintah COPY menggunakan pemrosesan paralel besar-besaran (MPP) dan rentang pemindaian untuk memuat data dari file besar dalam bucket Amazon S3.

Memuat data dari beberapa file terkompresi

Jika Anda memiliki data terkompresi, kami sarankan Anda membagi data untuk setiap tabel menjadi beberapa file. Perintah COPY dapat memuat data dari beberapa file secara paralel. Anda dapat memuat beberapa file dengan menentukan awalan umum, atau kunci awalan, untuk set, atau dengan mencantumkan file secara eksplisit dalam file manifes.

Pisahkan data Anda menjadi file sehingga jumlah file adalah kelipatan dari jumlah irisan di cluster Anda. Dengan begitu, Amazon Redshift dapat membagi data secara merata di antara irisan. Jumlah

irisan per node tergantung pada ukuran node cluster. Misalnya, setiap node komputasi dc2.large memiliki dua irisan, dan setiap node komputasi dc2.8xlarge memiliki 16 irisan. Untuk informasi selengkapnya tentang jumlah irisan yang dimiliki setiap ukuran node, lihat Tentang cluster dan node di Panduan Manajemen Amazon Redshift.

Semua node berpartisipasi dalam menjalankan kueri paralel, mengerjakan data yang didistribusikan serata mungkin di seluruh irisan. Jika Anda memiliki cluster dengan dua node dc2.large, Anda dapat membagi data Anda menjadi empat file atau beberapa kelipatan dari empat. Amazon Redshift tidak memperhitungkan ukuran file saat membagi beban kerja. Dengan demikian, Anda perlu memastikan bahwa file kira-kira berukuran sama, dari 1 MB hingga 1 GB setelah kompresi.

Untuk menggunakan awalan objek untuk mengidentifikasi file beban, beri nama setiap file dengan awalan umum. Misalnya, Anda dapat membagi `venue.txt` file mungkin dibagi menjadi empat file, sebagai berikut.

```
venue.txt.1
venue.txt.2
venue.txt.3
venue.txt.4
```

Jika Anda meletakkan beberapa file dalam folder di bucket Anda dan menentukan nama folder sebagai awalan, COPY memuat semua file dalam folder. Jika Anda secara eksplisit mencantumkan file yang akan dimuat dengan menggunakan file manifes, file tersebut dapat berada di bucket atau folder yang berbeda.

Untuk informasi selengkapnya tentang file manifes, lihatExample: COPY from Amazon S3 using a manifest.

Memuat data dari file yang tidak terkompresi dan dibatasi

Saat Anda memuat data yang tidak terkompresi dan dibatasi, perintah COPY menggunakan arsitektur pemrosesan paralel massif (MPP) di Amazon Redshift. Amazon Redshift secara otomatis menggunakan irisan yang bekerja secara paralel untuk memuat rentang data dari file besar dalam bucket Amazon S3. File harus dibatasi agar pemuatan paralel terjadi. Misalnya, pipa dibatasi. Pemuatan data paralel otomatis dengan perintah COPY juga tersedia untuk file CSV. Anda juga dapat memanfaatkan pemrosesan paralel dengan mengatur kunci distribusi pada tabel Anda. Untuk informasi selengkapnya tentang kunci distribusi, lihatDistribusi data untuk optimasi kueri.

Pemuatan data paralel otomatis tidak didukung ketika kueri COPY menyertakan salah satu kata kunci berikut: ESCAPE, REMOVEQUOTES, dan FIXEDWIDTH.

Data dari file atau file dimuat ke tabel target, satu baris per baris. Bidang dalam file data dicocokkan dengan kolom tabel secara berurutan, kiri ke kanan. Bidang dalam file data dapat berupa lebar tetap atau dibatasi karakter; pembatas default adalah pipa (|). Secara default, semua kolom tabel dimuat, tetapi Anda dapat secara opsional menentukan daftar kolom yang dipisahkan koma. Jika kolom tabel tidak disertakan dalam daftar kolom yang ditentukan dalam perintah COPY, kolom tersebut dimuat dengan nilai default. Untuk informasi selengkapnya, lihat Memuat nilai kolom default.

Ikuti proses umum ini untuk memuat data dari Amazon S3, saat data Anda tidak terkompresi dan dibatasi:

1. Unggah file Anda ke Amazon S3.
2. Jalankan perintah COPY untuk memuat tabel.
3. Pastikan data dimuat dengan benar.

Untuk contoh perintah SALIN, lihatContoh COPY. Untuk informasi tentang data yang dimuat ke Amazon Redshift, periksa tabel STL_LOAD_COMMIT dan STL_LOAD_ERROR sistem.

Untuk informasi selengkapnya tentang node dan irisan yang terkandung di masing-masing node, lihat Tentang cluster dan node di Panduan Manajemen Amazon Redshift.

Mengunggah file ke Amazon S3 untuk digunakan dengan COPY

Ada beberapa pendekatan yang harus diambil saat mengunggah file teks ke Amazon S3:

- Jika Anda memiliki file terkompresi, kami sarankan Anda membagi file besar untuk memanfaatkan pemrosesan paralel di Amazon Redshift.
- Di sisi lain, COPY secara otomatis membagi data file besar, tidak terkompresi, dan dibatasi teks untuk memfasilitasi paralelisme dan secara efektif mendistribusikan data dari file besar.

Buat bucket Amazon S3 untuk menyimpan file data Anda, lalu unggah file data ke bucket. Untuk informasi tentang membuat bucket dan mengunggah file, lihat Bek erja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Important**
>
> Bucket Amazon S3 yang menyimpan file data harus dibuat di Wil AWS ayah yang sama dengan cluster Anda kecuali Anda menggunakan REGION opsi untuk menentukan Wilayah tempat bucket Amazon S3 berada.

Pastikan bahwa rentang IP S3 ditambahkan ke daftar izin Anda. Untuk mempelajari selengkapnya tentang rentang IP S3 yang diperlukan, lihat Isolasi jaringan.

Anda dapat membuat bucket Amazon S3 di Wilayah tertentu dengan memilih Wilayah saat Anda membuat bucket menggunakan konsol Amazon S3, atau dengan menentukan titik akhir saat Anda membuat bucket menggunakan API atau CLI Amazon S3.

Setelah pemuatan data, verifikasi bahwa file yang benar ada di Amazon S3.

Topik

- Mengelola konsistensi data
- Mengunggah data terenkripsi ke Amazon S3
- Memverifikasi bahwa file yang benar ada di bucket Anda

Mengelola konsistensi data

Amazon S3 memberikan konsistensi baca-sesudah tulis yang kuat untuk operasi COPY, UNLOAD, INSERT (tabel eksternal), CREATE EXTERNAL TABLE AS, dan Amazon Redshift Spectrum pada bucket Amazon S3 di semua Wilayah. AWS Selain itu, operasi baca pada Amazon S3 Select, Daftar Kontrol Akses Amazon S3, Tag Objek Amazon S3, dan metadata objek (misalnya, objek HEAD) sangat konsisten. Untuk informasi selengkapnya tentang konsistensi data, lihat Model Konsistensi Data Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengunggah data terenkripsi ke Amazon S3

Amazon S3 mendukung enkripsi sisi server dan enkripsi sisi klien. Topik ini membahas perbedaan antara enkripsi sisi server dan sisi klien dan menjelaskan langkah-langkah untuk menggunakan enkripsi sisi klien dengan Amazon Redshift. Server-sideenkripsi transparan untuk Amazon Redshift.

Server-side enkripsi

Server-side enkripsi adalah enkripsi data yang diam—yaitu, Amazon S3 mengenkripsi data Anda saat mengunggahnya dan mendekripsi untuk Anda saat Anda mengaksesnya. Saat memuat tabel menggunakan perintah COPY, tidak ada perbedaan dalam cara Anda memuat dari objek terenkripsi atau tidak terenkripsi sisi server di Amazon S3. Untuk informasi selengkapnya tentang enkripsi sisi server, lihat Menggunakan Server-Side Enkripsi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Client-side enkripsi

Dalam enkripsi sisi klien, aplikasi klien Anda mengelola enkripsi data Anda, kunci enkripsi, dan alat terkait. Anda dapat mengunggah data ke bucket Amazon S3 menggunakan enkripsi sisi klien, lalu memuat data menggunakan perintah SALIN dengan opsi ENCRYPTED dan kunci enkripsi pribadi untuk memberikan keamanan yang lebih baik.

Anda mengenkripsi data Anda menggunakan enkripsi amplop. Dengan enkripsi amplop, aplikasi Anda menangani semua enkripsi secara eksklusif. Kunci enkripsi pribadi Anda dan data Anda yang tidak terenkripsi tidak pernah dikirim AWS, jadi sangat penting bagi Anda untuk mengelola kunci enkripsi Anda dengan aman. Jika Anda kehilangan kunci enkripsi, Anda tidak akan dapat membuka enkripsi data Anda, dan Anda tidak dapat memulihkan kunci enkripsi Anda. AWS Enkripsi amplop menggabungkan kinerja enkripsi simetris cepat sambil mempertahankan keamanan yang lebih besar yang disediakan oleh manajemen kunci dengan kunci asimetris. Kunci simetris sekali pakai (kunci simetris amplop) dihasilkan oleh klien enkripsi Amazon S3 Anda untuk mengenkripsi data Anda, kemudian kunci tersebut dienkripsi oleh kunci root Anda dan disimpan bersama data Anda di Amazon S3. Saat Amazon Redshift mengakses data Anda selama pemuatan, kunci simetris terenkripsi diambil dan didekripsi dengan kunci asli Anda, kemudian data didekripsi.

Untuk bekerja dengan data terenkripsi sisi klien Amazon S3 di Amazon Redshift, ikuti langkah-langkah yang diuraikan dalam Mel indungi Data Menggunakan Client-Side Enkripsi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon, dengan persyaratan tambahan yang Anda gunakan:

- Enkripsi simetris — AWS SDK untuk `AmazonS3EncryptionClient` kelas Java menggunakan enkripsi amplop, dijelaskan sebelumnya, yang didasarkan pada enkripsi kunci simetris. Gunakan kelas ini untuk membuat klien Amazon S3 untuk mengunggah data terenkripsi sisi klien.
- Kunci simetris akar AES 256-bit — Kunci root mengenkripsi kunci amplop. Anda meneruskan kunci root ke instance `AmazonS3EncryptionClient` kelas Anda. Simpan kunci ini, karena Anda akan membutuhkannya untuk menyalin data ke Amazon Redshift.

- Metadata objek untuk menyimpan kunci amplop terenkripsi — Secara default, Amazon S3 menyimpan kunci amplop sebagai metadata objek untuk `AmazonS3EncryptionClient` kelas. Kunci amplop terenkripsi yang disimpan sebagai metadata objek digunakan selama proses dekripsi.

> **Note**
>
> Jika Anda mendapatkan pesan kesalahan enkripsi cipher saat menggunakan API enkripsi untuk pertama kalinya, versi JDK Anda mungkin memiliki file kebijakan yurisdiksi Java Cryptography Extension (JCE) yang membatasi panjang kunci maksimum untuk transformasi enkripsi dan dekripsi hingga 128 bit. Untuk informasi tentang mengatasi masalah ini, buka Menentukan En Client-Side kripsi Menggunakan AWS SDK untuk Java di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk informasi tentang memuat file terenkripsi sisi klien ke tabel Amazon Redshift menggunakan perintah COPY, lihat. Memuat file data terenkripsi dari Amazon S3

Contoh: Mengunggah data terenkripsi sisi klien

Untuk contoh cara menggunakan AWS SDK untuk Java untuk mengunggah data terenkripsi sisi klien, buka Mel indungi data menggunakan enkripsi sisi klien di Panduan Pengguna Amazon Simple Storage Service.

Opsi kedua menunjukkan pilihan yang harus Anda buat selama enkripsi sisi klien sehingga data dapat dimuat di Amazon Redshift. Secara khusus, contoh menunjukkan penggunaan metadata objek untuk menyimpan kunci amplop terenkripsi dan penggunaan kunci simetris akar AES 256-bit.

Contoh ini memberikan contoh kode menggunakan AWS SDK untuk Java untuk membuat kunci akar simetris AES 256-bit dan menyimpannya ke file. Kemudian contoh mengunggah objek ke Amazon S3 menggunakan klien enkripsi S3 yang terlebih dahulu mengenkripsi data sampel di sisi klien. Contoh ini juga mengunduh objek dan memverifikasi bahwa datanya sama.

Memverifikasi bahwa file yang benar ada di bucket Anda

Setelah mengunggah file ke bucket Amazon S3, sebaiknya daftar isi bucket untuk memverifikasi bahwa semua file yang benar ada dan tidak ada file yang tidak diinginkan. Misalnya, jika bucket `amzn-s3-demo-bucket` menyimpan file bernama`venue.txt.back`, file itu akan dimuat, mungkin secara tidak sengaja, dengan perintah berikut:

```
COPY venue FROM 's3://amzn-s3-demo-bucket/venue' … ;
```

Jika Anda ingin mengontrol secara spesifik file mana yang dimuat, Anda dapat menggunakan file manifes untuk mencantumkan file data secara eksplisit. Untuk informasi selengkapnya tentang menggunakan file manifes, lihat copy_from_s3_manifest_file opsi untuk perintah SALIN dan Example: COPY from Amazon S3 using a manifest dalam contoh SALIN.

Untuk informasi selengkapnya tentang mencantumkan konten bucket, lihat Daftar Kunci Ob jek di Panduan Pengembang Amazon S3.

Menggunakan perintah COPY untuk memuat dari Amazon S3

Gunakan MENYALIN perintah untuk memuat tabel secara paralel dari file data di Amazon S3. Anda dapat menentukan file yang akan dimuat dengan menggunakan awalan objek Amazon S3 atau dengan menggunakan file manifes.

Sintaks untuk menentukan file yang akan dimuat dengan menggunakan awalan adalah sebagai berikut:

```
COPY <table_name> FROM 's3://<bucket_name>/<object_prefix>'
authorization;
```

File manifes adalah JSON-formatted file yang mencantumkan file data yang akan dimuat. Sintaks untuk menentukan file yang akan dimuat dengan menggunakan file manifes adalah sebagai berikut:

```
COPY <table_name> FROM 's3://<bucket_name>/<manifest_file>'
authorization
MANIFEST;
```

Tabel yang akan dimuat harus sudah ada di database. Untuk informasi tentang membuat tabel, lihat CREATE TABLE di Referensi SQL.

Nilai untuk otorisasi memberikan AWS otorisasi yang dibutuhkan Amazon Redshift untuk mengakses objek Amazon S3. Untuk informasi tentang izin yang diperlukan, lihatIzin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY. Metode otentikasi yang lebih disukai adalah menentukan parameter IAM_ROLE dan memberikan Amazon Resource Name (ARN) untuk peran IAM dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Role-based kontrol akses.

Untuk mengotentikasi menggunakan parameter IAM_ROLE, ganti *<aws-account-id>* dan *<role-name>* seperti yang ditunjukkan dalam sintaks berikut.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

Contoh berikut menunjukkan otentikasi menggunakan peran IAM.

```
COPY customer
FROM 's3://amzn-s3-demo-bucket/mydata'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Untuk informasi selengkapnya tentang opsi otorisasi lainnya, lihat Parameter otorisasi

Jika Anda ingin memvalidasi data Anda tanpa benar-benar memuat tabel, gunakan opsi NOLOAD dengan perintahMENYALIN.

Contoh berikut menunjukkan beberapa baris pertama dari data yang dibatasi pipa dalam file bernama. `venue.txt`

```
1|Toyota Park|Bridgeview|IL|0
2|Columbus Crew Stadium|Columbus|OH|0
3|RFK Stadium|Washington|DC|0
```

Sebelum mengunggah file ke Amazon S3, pisahkan file menjadi beberapa file sehingga perintah COPY dapat memuatnya menggunakan pemrosesan paralel. Jumlah file harus kelipatan dari jumlah irisan di cluster Anda. Pisahkan file data muat Anda sehingga file berukuran hampir sama, antara 1 MB dan 1 GB setelah kompresi. Untuk informasi selengkapnya, lihat Memuat data dari file terkompresi dan tidak terkompresi.

Misalnya, `venue.txt` file dapat dibagi menjadi empat file, sebagai berikut:

```
venue.txt.1
venue.txt.2
venue.txt.3
venue.txt.4
```

Perintah COPY berikut memuat tabel VENUE menggunakan data yang dibatasi pipa dalam file data dengan awalan 'venue' di bucket Amazon S3. `amzn-s3-demo-bucket`

> **Note**
>
> Bucket Amazon S3 `amzn-s3-demo-bucket` dalam contoh berikut tidak ada. Untuk contoh perintah COPY yang menggunakan data nyata di bucket Amazon S3 yang ada, lihat Mem uat data sampel.

```
COPY venue FROM 's3://amzn-s3-demo-bucket/venue'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
DELIMITER '|';
```

Jika tidak ada objek Amazon S3 dengan awalan kunci 'venue', pemuatan gagal.

Topik

- Menggunakan manifes untuk menentukan file data
- Memuat file data terkompresi dari Amazon S3
- Memuat data lebar tetap dari Amazon S3
- Memuat data multibyte dari Amazon S3
- Memuat file data terenkripsi dari Amazon S3

Menggunakan manifes untuk menentukan file data

Anda dapat menggunakan manifes untuk memastikan bahwa perintah COPY memuat semua file yang diperlukan, dan hanya file yang diperlukan, untuk pemuatan data. Anda dapat menggunakan manifes untuk memuat file dari bucket atau file yang berbeda yang tidak memiliki awalan yang sama. Alih-alih menyediakan jalur objek untuk perintah COPY, Anda memberikan nama file JSON-formatted teks yang secara eksplisit mencantumkan file yang akan dimuat. URL dalam manifes harus menentukan nama bucket dan jalur objek lengkap untuk file, bukan hanya awalan.

Untuk informasi selengkapnya tentang file manifes, lihat contoh SAL IN Menggunakan manifes untuk menentukan file data.

Contoh berikut menunjukkan JSON untuk memuat file dari bucket yang berbeda dan dengan nama file yang dimulai dengan stempel tanggal.

```
{
```

```
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket1/2013-10-05-custdata", "mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket2/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket2/2013-10-05-custdata", "mandatory":true}
  ]
}
```

`mandatory`Bendera opsional menentukan apakah COPY harus mengembalikan kesalahan jika file tidak ditemukan. Defaultnya `mandatory` adalah`false`. Terlepas dari pengaturan wajib, COPY akan berakhir jika tidak ada file yang ditemukan.

Contoh berikut menjalankan perintah COPY dengan manifes dalam contoh sebelumnya, yang diberi nama`cust.manifest`.

```
COPY customer
FROM 's3://amzn-s3-demo-bucket/cust.manifest'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
MANIFEST;
```

Menggunakan manifes yang dibuat oleh UNLOAD

Manifes yang dibuat oleh MEMBONGKAR operasi menggunakan parameter MANIFEST mungkin memiliki kunci yang tidak diperlukan untuk operasi COPY. Misalnya, `UNLOAD` manifes berikut menyertakan `meta` kunci yang diperlukan untuk tabel eksternal Amazon Redshift Spectrum dan untuk memuat file data dalam format `Parquet` file `ORC` atau. K `meta` unci berisi `content_length` kunci dengan nilai yang merupakan ukuran sebenarnya dari file dalam byte. Operasi COPY hanya membutuhkan `url` kunci dan `mandatory` kunci opsional.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/unload/manifest_0000_part_00", "meta":
 { "content_length": 5956875 }},
    {"url":"s3://amzn-s3-demo-bucket/unload/unload/manifest_0001_part_00", "meta":
 { "content_length": 5997091 }}
 ]
}
```

Untuk informasi selengkapnya tentang file manifes, lihatExample: COPY from Amazon S3 using a manifest.

Memuat file data terkompresi dari Amazon S3

Untuk memuat file data yang dikompresi menggunakan gzip, lzop, atau bzip2, sertakan opsi yang sesuai: GZIP, LZOP, atau BZIP2.

Misalnya, perintah berikut dimuat dari file yang dikompresi menggunakan lzop.

```
COPY customer FROM 's3://amzn-s3-demo-bucket/customer.lzo'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
DELIMITER '|' LZOP;
```

> **Note**
>
> Jika Anda mengompres file data dengan kompresi lzop dan menggunakan opsi --filter, perintah COPY tidak mendukungnya.

Memuat data lebar tetap dari Amazon S3

Fixed-width file data memiliki panjang yang seragam untuk setiap kolom data. Setiap bidang dalam file data lebar tetap memiliki panjang dan posisi yang persis sama. Untuk data karakter (CHAR dan VARCHAR) dalam file data lebar tetap, Anda harus menyertakan spasi di depan atau di belakang sebagai placeholder agar lebar tetap seragam. Untuk bilangan bulat, Anda harus menggunakan nol di depan sebagai placeholder. File data lebar tetap tidak memiliki pembatas untuk memisahkan kolom.

Untuk memuat file data dengan lebar tetap ke tabel yang ada, GUNAKAN parameter FIXEDWIDTH dalam perintah COPY. Spesifikasi tabel Anda harus sesuai dengan nilai fixedwidth_spec agar data dimuat dengan benar.

Untuk memuat data dengan lebar tetap dari file ke tabel, berikan perintah berikut:

```
COPY table_name FROM 's3://amzn-s3-demo-bucket/prefix'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
FIXEDWIDTH 'fixedwidth_spec';
```

Parameter fixedwidth_spec adalah string yang berisi pengidentifikasi untuk setiap kolom dan lebar setiap kolom, dipisahkan oleh titik dua. Pas **`column:width`** angan dibatasi oleh koma. Pengidentifikasi dapat berupa apa pun yang Anda pilih: angka, huruf, atau kombinasi keduanya.

Pengidentifikasi tidak memiliki hubungan dengan tabel itu sendiri, sehingga spesifikasi harus berisi kolom dalam urutan yang sama dengan tabel.

Dua contoh berikut menunjukkan spesifikasi yang sama, dengan yang pertama menggunakan pengidentifikasi numerik dan yang kedua menggunakan pengidentifikasi string:

```
'0:3,1:25,2:12,3:2,4:6'
```

```
'venueid:3,venuename:25,venuecity:12,venuestate:2,venueseats:6'
```

Contoh berikut menunjukkan data sampel dengan lebar tetap yang dapat dimuat ke dalam tabel VENUE menggunakan spesifikasi sebelumnya:

```
1  Toyota Park              Bridgeview  IL0
2  Columbus Crew Stadium    Columbus    OH0
3  RFK Stadium              Washington  DC0
4  CommunityAmerica Ballpark Kansas City KS0
5  Gillette Stadium         Foxborough  MA68756
```

Perintah COPY berikut memuat kumpulan data ini ke dalam tabel VENUE:

```
COPY venue
FROM 's3://amzn-s3-demo-bucket/data/venue_fw.txt'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
FIXEDWIDTH 'venueid:3,venuename:25,venuecity:12,venuestate:2,venueseats:6';
```

Memuat data multibyte dari Amazon S3

Jika data Anda menyertakan karakter multibyte non-ASCII (seperti karakter Mandarin atau Cyrillic), Anda harus memuat data ke kolom VARCHAR. Tipe data VARCHAR mendukung UTF-8 karakter empat byte, tetapi tipe data CHAR hanya menerima karakter ASCII byte tunggal. Anda tidak dapat memuat karakter lima byte atau lebih lama ke tabel Amazon Redshift. Untuk informasi lebih lanjut tentang CHAR dan VARCHAR, lihatJenis Data.

Untuk memeriksa pengkodean mana yang digunakan file input, gunakan *`file`* perintah Linux:

```
$ file ordersdata.txt
ordersdata.txt: ASCII English text
$ file uni_ordersdata.dat
```

```
uni_ordersdata.dat: UTF-8 Unicode text
```

Memuat file data terenkripsi dari Amazon S3

Anda dapat menggunakan perintah COPY untuk memuat file data yang diunggah ke Amazon S3 menggunakan enkripsi sisi server, enkripsi sisi klien, atau keduanya.

Perintah COPY mendukung jenis enkripsi Amazon S3 berikut:

- Server-side enkripsi dengan S3-managed kunci Amazon (SSE-S3)
- Server-side enkripsi dengan AWS KMS keys (SSE-KMS)
- Client-side enkripsi menggunakan kunci root simetris sisi klien

Perintah COPY tidak mendukung jenis enkripsi Amazon S3 berikut:

- Server-side enkripsi dengan kunci yang disediakan pelanggan () SSE-C
- Client-side enkripsi menggunakan AWS KMS key
- Client-side enkripsi menggunakan kunci root asimetris yang disediakan pelanggan

Untuk informasi selengkapnya tentang enkripsi Amazon S3, lihat Melindungi Data Menggunakan Server-Side Enkripsi dan Melindungi Data Menggunakan Client-Side Enkripsi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Per MEMBONGKAR intah secara otomatis mengenkripsi file menggunakan SSE-S3. Anda juga dapat membongkar menggunakan SSE-KMS atau enkripsi sisi klien dengan kunci simetris yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Membongkar file data terenkripsi

Perintah COPY secara otomatis mengenali dan memuat file yang dienkripsi menggunakan SSE-S3 dan SSE-KMS. Anda dapat memuat file yang dienkripsi menggunakan kunci root simetris sisi klien dengan menentukan opsi ENCRYPTED dan memberikan nilai kunci. Untuk informasi selengkapnya, lihat Mengunggah data terenkripsi ke Amazon S3.

Untuk memuat file data terenkripsi sisi klien, berikan nilai kunci root menggunakan parameter MASTER_SYMMETRIC_KEY dan sertakan opsi ENCRYPTED.

```
COPY customer FROM 's3://amzn-s3-demo-bucket/encrypted/customer'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
MASTER_SYMMETRIC_KEY '<root_key>'
```

```
ENCRYPTED
DELIMITER '|';
```

Untuk memuat file data terenkripsi yang dikompresi gzip, lzop, atau bzip2, sertakan opsi GZIP, LZOP, atau BZIP2 bersama dengan nilai kunci root dan opsi ENCRYPTED.

```
COPY customer FROM 's3://amzn-s3-demo-bucket/encrypted/customer'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
MASTER_SYMMETRIC_KEY '<root_key>'
ENCRYPTED
DELIMITER '|'
GZIP;
```

Memuat data dari Amazon EMR

Anda dapat menggunakan perintah COPY untuk memuat data secara paralel dari cluster Amazon EMR yang dikonfigurasi untuk menulis file teks ke Hadoop Distributed File System (HDFS) cluster sebagai file dengan lebar tetap, file yang dibatasi karakter, file CSV, atau file. JSON-formatted

Proses untuk memuat data dari Amazon EMR

Bagian ini memandu Anda melalui proses memuat data dari cluster Amazon EMR. Bagian berikut memberikan rincian yang harus Anda selesaikan setiap langkah.

- Langkah 1: Konfigurasikan izin IAM

  Pengguna yang membuat cluster Amazon EMR dan menjalankan perintah Amazon Redshift COPY harus memiliki izin yang diperlukan.
- Langkah 2: Buat cluster Amazon EMR

  Konfigurasikan cluster untuk menampilkan file teks ke Hadoop Distributed File System (HDFS). Anda memerlukan ID cluster Amazon EMR dan DNS publik utama cluster (titik akhir untuk instans Amazon EC2 yang menghosting cluster).
- Langkah 3: Ambil kunci publik cluster Amazon Redshift dan alamat IP node cluster

  Kunci publik memungkinkan node cluster Amazon Redshift untuk membuat koneksi SSH ke host. Anda akan menggunakan alamat IP untuk setiap node cluster untuk mengonfigurasi grup keamanan host untuk mengizinkan akses dari cluster Amazon Redshift Anda menggunakan alamat IP ini.

- Langkah 4: Tambahkan kunci publik cluster Amazon Redshift ke setiap file kunci resmi host Amazon EC2

  Anda menambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host sehingga host akan mengenali cluster Amazon Redshift dan menerima koneksi SSH.

- Langkah 5: Konfigurasikan host untuk menerima semua alamat IP cluster Amazon Redshift

  Ubah grup keamanan instans Amazon EMR untuk menambahkan aturan input untuk menerima alamat IP Amazon Redshift.

- Langkah 6: Jalankan perintah COPY untuk memuat data

  Dari database Amazon Redshift, jalankan perintah COPY untuk memuat data ke tabel Amazon Redshift.

Langkah 1: Konfigurasikan izin IAM

Pengguna yang membuat cluster Amazon EMR dan menjalankan perintah Amazon Redshift COPY harus memiliki izin yang diperlukan.

Untuk mengkonfigurasi izin IAM

1. Tambahkan izin berikut untuk pengguna yang akan membuat cluster Amazon EMR.

```
ec2:DescribeSecurityGroups
ec2:RevokeSecurityGroupIngress
ec2:AuthorizeSecurityGroupIngress
redshift:DescribeClusters
```

2. Tambahkan izin berikut untuk peran IAM atau pengguna yang akan menjalankan perintah COPY.

```
elasticmapreduce:ListInstances
```

3. Tambahkan izin berikut ke peran IAM cluster Amazon EMR.

```
redshift:DescribeClusters
```

Langkah 2: Buat cluster Amazon EMR

Perintah COPY memuat data dari file di Amazon EMR Hadoop Distributed File System (HDFS). Saat Anda membuat cluster Amazon EMR, konfigurasikan cluster untuk menampilkan file data ke HDFS cluster.

Cara membuat cluster Amazon EMR

1. Buat cluster Amazon EMR di Wilayah yang sama AWS dengan cluster Amazon Redshift.

   Jika cluster Amazon Redshift berada di VPC, cluster Amazon EMR harus berada di grup VPC yang sama. Jika cluster Amazon Redshift menggunakan EC2-Classic mode (yaitu, tidak dalam VPC), cluster Amazon EMR juga harus menggunakan mode. EC2-Classic Untuk informasi selengkapnya, lihat Mengelola Cluster di Virtual Private Cloud (VPC) di Panduan Manajemen Amazon Redshift.
2. Konfigurasikan cluster untuk menampilkan file data ke HDFS cluster. Nama file HDFS tidak boleh menyertakan tanda bintang (*) atau tanda tanya (?).

   > **Important**
   >
   > Nama file tidak boleh menyertakan tanda bintang (*) atau tanda tanya (?).
3. Tentukan Tidak untuk Auto-terminate opsi dalam konfigurasi cluster Amazon EMR sehingga cluster tetap tersedia saat perintah COPY berjalan.

   > **Important**
   >
   > Jika salah satu file data diubah atau dihapus sebelum COPY selesai, Anda mungkin memiliki hasil yang tidak terduga, atau operasi COPY mungkin gagal.
4. Perhatikan ID cluster dan DNS publik utama (titik akhir untuk instans Amazon EC2 yang menghosting cluster). Anda akan menggunakan informasi tersebut di langkah selanjutnya.

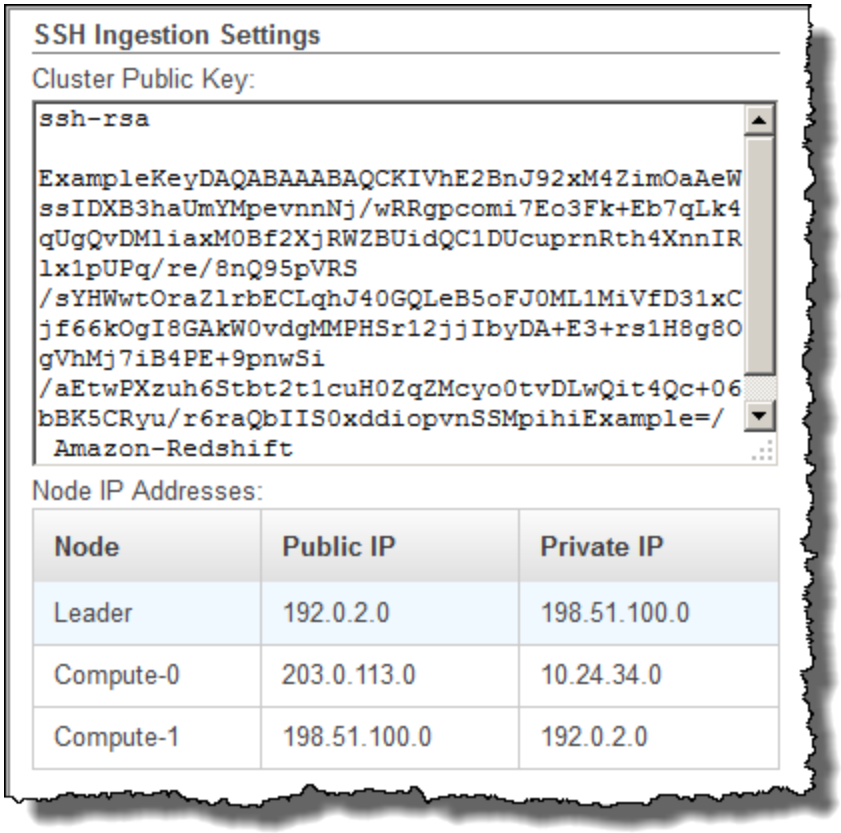
Langkah 3: Ambil kunci publik cluster Amazon Redshift dan alamat IP node cluster

Anda akan menggunakan alamat IP untuk setiap node cluster untuk mengonfigurasi grup keamanan host untuk mengizinkan akses dari cluster Amazon Redshift Anda menggunakan alamat IP ini.

Untuk mengambil kunci publik cluster Amazon Redshift dan alamat IP node cluster untuk cluster Anda menggunakan konsol

1. Akses Konsol Manajemen Amazon Redshift.
2. Pilih tautan Cluster di panel navigasi.
3. Pilih cluster Anda dari daftar.
4. Temukan grup Peng aturan Ingestion SSH.

   Perhatikan Kunci Publik Cluster dan alamat IP Node. Anda akan menggunakannya di langkah selanjutnya.



   Anda akan menggunakan alamat IP pribadi di Langkah 3 untuk mengonfigurasi host Amazon EC2 untuk menerima koneksi dari Amazon Redshift.

Untuk mengambil kunci publik cluster dan alamat IP node cluster untuk cluster Anda menggunakan Amazon Redshift CLI, jalankan perintah describe-cluster. Contoh:

```
aws redshift describe-clusters --cluster-identifier <cluster-identifier>
```

Tanggapan akan mencakup ClusterPublicKey nilai dan daftar alamat IP pribadi dan publik, mirip dengan yang berikut:

```
{
    "Clusters": [
        {
            "VpcSecurityGroups": [],
            "ClusterStatus": "available",
            "ClusterNodes": [
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "LEADER",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-0",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-1",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                }
            ],
            "AutomatedSnapshotRetentionPeriod": 1,
            "PreferredMaintenanceWindow": "wed:05:30-wed:06:00",
            "AvailabilityZone": "us-east-1a",
            "NodeType": "dc2.large",
            "ClusterPublicKey": "ssh-rsa AAAABexamplepublickey...Y3TAl Amazon-
Redshift",
             ...
             ...
}
```

Untuk mengambil kunci publik cluster dan alamat IP node cluster untuk cluster Anda menggunakan Amazon Redshift API, gunakan `DescribeClusters` tindakan tersebut. Untuk informasi selengkapnya, lihat deskripsi-cluster di Panduan Amazon Redshift CLI atau DescribeClusters di Panduan API Amazon Redshift.

Langkah 4: Tambahkan kunci publik cluster Amazon Redshift ke setiap file kunci resmi host Amazon EC2

Anda menambahkan kunci publik cluster ke file kunci resmi masing-masing host untuk semua node cluster Amazon EMR sehingga host akan mengenali Amazon Redshift dan menerima koneksi SSH.

Untuk menambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host

1. Akses host menggunakan koneksi SSH.

   Untuk informasi tentang menghubungkan ke instans menggunakan SSH, lihat Mengh ubungkan ke Instans Anda di Panduan Pengguna Amazon EC2.
2. Salin kunci publik Amazon Redshift dari konsol atau dari teks respons CLI.
3. Salin dan tempel isi kunci publik ke dalam `/home/<ssh_username>/.ssh/authorized_keys` file di host. Sertakan string lengkap, termasuk awalan "`ssh-rsa`" dan akhiran "`Amazon-Redshift`". Contoh:

   ```
   ssh-rsa AAAACTP3isxgGzVWoIWpbVvRCOzYdVifMrh… uA70BnMHCaMiRdmvsDOedZDOedZ Amazon-
   Redshift
   ```

Langkah 5: Konfigurasikan host untuk menerima semua alamat IP cluster Amazon Redshift

Untuk mengizinkan lalu lintas masuk ke instans host, edit grup keamanan dan tambahkan satu aturan Masuk untuk setiap node cluster Amazon Redshift. Untuk Jenis, pilih SSH dengan protokol TCP pada Port 22. Untuk Sumber, masukkan alamat IP pribadi simpul cluster Amazon Redshift yang Anda ambil. Langkah 3: Ambil kunci publik cluster Amazon Redshift dan alamat IP node cluster Untuk informasi tentang menambahkan aturan ke grup keamanan Amazon EC2, lihat Mengotor isasi Lalu Lintas Masuk untuk Instans Anda di Panduan Pengguna Amazon EC2.

Langkah 6: Jalankan perintah COPY untuk memuat data

Jalan MENYALIN kan perintah untuk terhubung ke cluster Amazon EMR dan memuat data ke tabel Amazon Redshift. Cluster Amazon EMR harus terus berjalan sampai perintah COPY selesai. Misalnya, jangan mengkonfigurasi cluster untuk mengakhiri secara otomatis.

> **Important**
>
> Jika salah satu file data diubah atau dihapus sebelum COPY selesai, Anda mungkin memiliki hasil yang tidak terduga, atau operasi COPY mungkin gagal.

Dalam perintah COPY, tentukan ID cluster Amazon EMR dan jalur file HDFS dan nama file.

```
COPY sales
FROM 'emr://myemrclusterid/myoutput/part*' CREDENTIALS
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Anda dapat menggunakan karakter wildcard asterisk (*) dan tanda tanya (?) sebagai bagian dari argumen nama file. Misalnya, `part*` memuat file`part-0000`,`part-0001`, dan sebagainya. Jika Anda hanya menentukan nama folder, COPY mencoba memuat semua file dalam folder.

> **Important**
>
> Jika Anda menggunakan karakter wildcard atau hanya menggunakan nama folder, verifikasi bahwa tidak ada file yang tidak diinginkan akan dimuat atau perintah COPY akan gagal. Misalnya, beberapa proses mungkin menulis file log ke folder output.

Memuat data dari host jarak jauh

Anda dapat menggunakan perintah COPY untuk memuat data secara paralel dari satu atau beberapa host jarak jauh, seperti instans Amazon EC2 atau komputer lain. COPY terhubung ke host jarak jauh menggunakan SSH dan menjalankan perintah pada host jarak jauh untuk menghasilkan output teks.

Host jarak jauh dapat berupa instance Amazon EC2 Linux atau komputer Unix atau Linux lain yang dikonfigurasi untuk menerima koneksi SSH. Panduan ini mengasumsikan host jarak jauh Anda adalah instans Amazon EC2. Jika prosedurnya berbeda untuk komputer lain, panduan akan menunjukkan perbedaannya.

Amazon Redshift dapat terhubung ke beberapa host, dan dapat membuka beberapa koneksi SSH ke setiap host. Amazon Redshift mengirimkan perintah unik melalui setiap koneksi untuk menghasilkan output teks ke output standar host, yang kemudian dibaca oleh Amazon Redshift seperti file teks.

Sebelum Anda mulai

Sebelum Anda mulai, Anda harus memiliki yang berikut:

- Satu atau beberapa mesin host, seperti instans Amazon EC2, yang dapat Anda sambungkan menggunakan SSH.
- Sumber data pada host.

  Anda akan memberikan perintah yang akan dijalankan oleh cluster Amazon Redshift pada host untuk menghasilkan output teks. Setelah cluster terhubung ke host, perintah COPY menjalankan perintah, membaca teks dari output standar host, dan memuat data secara paralel ke dalam tabel Amazon Redshift. Output teks harus dalam bentuk yang dapat dicerna oleh perintah COPY. Untuk informasi selengkapnya, lihat Mempersiapkan data input Anda
- Akses ke host dari komputer Anda.

  Untuk instans Amazon EC2, Anda akan menggunakan koneksi SSH untuk mengakses host. Anda harus mengakses host untuk menambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host.
- Cluster Amazon Redshift yang sedang berjalan.

  Untuk informasi tentang cara meluncurkan cluster, lihat Panduan https://docs.aws.amazon.com/redshift/latest/gsg/ Memulai Amazon Redshift.

Proses pemuatan data

Bagian ini memandu Anda melalui proses memuat data dari host jarak jauh. Bagian berikut memberikan rincian yang harus Anda capai di setiap langkah.

- Langkah 1: Ambil kunci publik cluster dan alamat IP node cluster

  Kunci publik memungkinkan node cluster Amazon Redshift untuk membuat koneksi SSH ke host jarak jauh. Anda akan menggunakan alamat IP untuk setiap node cluster untuk mengonfigurasi grup keamanan host atau firewall untuk mengizinkan akses dari cluster Amazon Redshift Anda menggunakan alamat IP ini.
- Langkah 2: Tambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host

  Anda menambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host sehingga host akan mengenali cluster Amazon Redshift dan menerima koneksi SSH.

- Langkah 3: Konfigurasikan host untuk menerima semua alamat IP cluster Amazon Redshift

  Untuk Amazon EC2, ubah grup keamanan instans untuk menambahkan aturan input untuk menerima alamat IP Amazon Redshift. Untuk host lain, ubah firewall sehingga node Amazon Redshift Anda dapat membuat koneksi SSH ke host jarak jauh.

- Langkah 4: Dapatkan kunci publik untuk tuan rumah

  Anda dapat secara opsional menentukan bahwa Amazon Redshift harus menggunakan kunci publik untuk mengidentifikasi host. Anda harus menemukan kunci publik dan menyalin teks ke file manifes Anda.

- Langkah 5: Buat file manifes

  Manifes adalah file JSON-formatted teks dengan detail yang dibutuhkan Amazon Redshift untuk terhubung ke host dan mengambil data.

- Langkah 6: Unggah file manifes ke bucket Amazon S3

  Amazon Redshift membaca manifes dan menggunakan informasi tersebut untuk terhubung ke host jarak jauh. Jika bucket Amazon S3 tidak berada di Wilayah yang sama dengan cluster Amazon Redshift Anda, Anda harus menggunakan REGION opsi untuk menentukan Wilayah tempat data berada.

- Langkah 7: Jalankan perintah COPY untuk memuat data

  Dari database Amazon Redshift, jalankan perintah COPY untuk memuat data ke tabel Amazon Redshift.

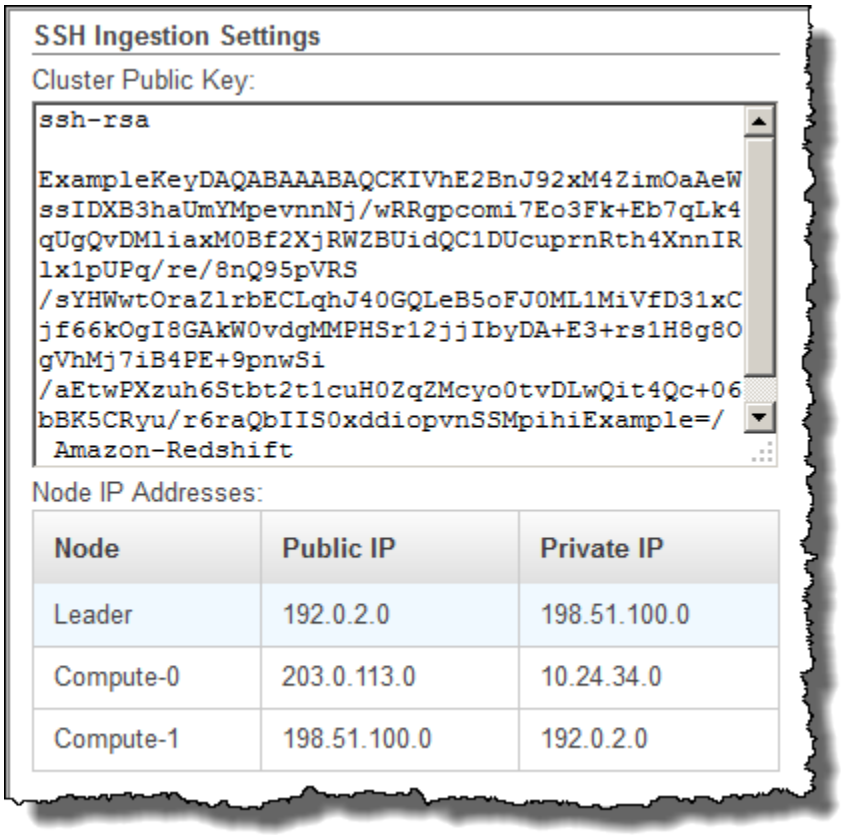
Langkah 1: Ambil kunci publik cluster dan alamat IP node cluster

Anda akan menggunakan alamat IP untuk setiap node cluster untuk mengonfigurasi grup keamanan host untuk mengizinkan akses dari cluster Amazon Redshift Anda menggunakan alamat IP ini.

Untuk mengambil kunci publik cluster dan alamat IP node cluster untuk cluster Anda menggunakan konsol

1. Akses Konsol Manajemen Amazon Redshift.
2. Pilih tautan Cluster di panel navigasi.
3. Pilih cluster Anda dari daftar.
4. Temukan grup Peng aturan Ingestion SSH.

Perhatikan Kunci Publik Cluster dan alamat IP Node. Anda akan menggunakannya di langkah selanjutnya.



Anda akan menggunakan alamat IP di Langkah 3 untuk mengonfigurasi host untuk menerima koneksi dari Amazon Redshift. Tergantung pada jenis host yang Anda sambungkan dan apakah itu dalam VPC, Anda akan menggunakan alamat IP publik atau alamat IP pribadi.

Untuk mengambil kunci publik cluster dan alamat IP node cluster untuk cluster Anda menggunakan Amazon Redshift CLI, jalankan perintah describe-cluster.

Contoh:

```
aws redshift describe-clusters --cluster-identifier <cluster-identifier>
```

Tanggapan akan mencakup ClusterPublicKey dan daftar alamat IP pribadi dan publik, mirip dengan berikut ini:

```
{
    "Clusters": [
```

```
        {
            "VpcSecurityGroups": [],
            "ClusterStatus": "available",
            "ClusterNodes": [
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "LEADER",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-0",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-1",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                }
            ],
            "AutomatedSnapshotRetentionPeriod": 1,
            "PreferredMaintenanceWindow": "wed:05:30-wed:06:00",
            "AvailabilityZone": "us-east-1a",
            "NodeType": "dc2.large",
            "ClusterPublicKey": "ssh-rsa AAAABexamplepublickey...Y3TAl Amazon-
Redshift",
             ...
             ...
}
```

Untuk mengambil kunci publik cluster dan alamat IP node cluster untuk cluster Anda menggunakan Amazon Redshift API, gunakan DescribeClusters tindakan tersebut. Untuk informasi selengkapnya, lihat deskripsi-cluster di Panduan Amazon Redshift CLI atau DescribeClusters di Panduan API Amazon Redshift.

Langkah 2: Tambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host

Anda menambahkan kunci publik cluster ke file kunci resmi masing-masing host sehingga host akan mengenali Amazon Redshift dan menerima koneksi SSH.

Untuk menambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host

1. Akses host menggunakan koneksi SSH.

Untuk informasi tentang menghubungkan ke instans menggunakan SSH, lihat Mengh ubungkan ke Instans Anda di Panduan Pengguna Amazon EC2.

2. Salin kunci publik Amazon Redshift dari konsol atau dari teks respons CLI.
3. Salin dan tempel isi kunci publik ke dalam `/home/<ssh_username>/.ssh/authorized_keys` file di host jarak jauh. Harus `<ssh_username>` cocok dengan nilai untuk bidang “nama pengguna” di file manifes. Sertakan string lengkap, termasuk awalan "`ssh-rsa`" dan akhiran "`Amazon-Redshift`”. Contoh:

```
ssh-rsa AAAACTP3isxgGzVWoIWpbVvRCOzYdVifMrh… uA70BnMHCaMiRdmvsDOedZDOedZ Amazon-
Redshift
```

Langkah 3: Konfigurasikan host untuk menerima semua alamat IP cluster Amazon Redshift

Jika Anda bekerja dengan instans Amazon EC2 atau cluster Amazon EMR, tambahkan aturan masuk ke grup keamanan host untuk mengizinkan lalu lintas dari setiap node cluster Amazon Redshift. Untuk Jenis, pilih SSH dengan protokol TCP pada Port 22. Untuk Sumber, masukkan alamat IP node cluster Amazon Redshift yang Anda ambil. Langkah 1: Ambil kunci publik cluster dan alamat IP node cluster Untuk informasi tentang menambahkan aturan ke grup keamanan Amazon EC2, lihat Mengotor isasi Lalu Lintas Masuk untuk Instans Anda di Panduan Pengguna Amazon EC2.

Gunakan alamat IP pribadi saat:

- Anda memiliki cluster Amazon Redshift yang tidak ada di Virtual Private Cloud (VPC), dan instans Amazon EC2 -Classic, yang keduanya berada di Wilayah yang sama. AWS
- Anda memiliki cluster Amazon Redshift yang ada di VPC, dan instans Amazon EC2 -VPC, keduanya berada di Wilayah yang sama dan di VPC yang sama AWS .

Jika tidak, gunakan alamat IP publik.

Untuk informasi selengkapnya tentang menggunakan Amazon Redshift di VPC, lihat M engelola Cluster di Virtual Private Cloud (VPC) di Panduan Manajemen Amazon Redshift.

Langkah 4: Dapatkan kunci publik untuk tuan rumah

Secara opsional, Anda dapat memberikan kunci publik host dalam file manifes sehingga Amazon Redshift dapat mengidentifikasi host. Perintah COPY tidak memerlukan kunci publik host tetapi, untuk

alasan keamanan, kami sangat menyarankan menggunakan kunci publik untuk membantu mencegah serangan “man-in-the-middle”.

Anda dapat menemukan kunci publik host di lokasi berikut, di `<ssh_host_rsa_key_name>` mana nama unik untuk kunci publik host:

```
:  /etc/ssh/<ssh_host_rsa_key_name>.pub
```

> **Note**
>
> Amazon Redshift hanya mendukung kunci RSA. Amazon Redshift tidak mendukung kunci DSA.

Saat Anda membuat file manifes Anda di Langkah 5, Anda akan menempelkan teks kunci publik ke bidang “Kunci Publik” di entri file manifes.

Langkah 5: Buat file manifes

Perintah COPY dapat terhubung ke beberapa host menggunakan SSH, dan dapat membuat beberapa koneksi SSH ke setiap host. COPY menjalankan perintah melalui setiap koneksi host, dan kemudian memuat output dari perintah secara paralel ke dalam tabel. File manifes adalah file teks dalam format JSON yang digunakan Amazon Redshift untuk terhubung ke host. File manifes menentukan titik akhir host SSH dan perintah yang dijalankan pada host untuk mengembalikan data ke Amazon Redshift. Secara opsional, Anda dapat menyertakan kunci publik host, nama pengguna login, dan tanda wajib untuk setiap entri.

Buat file manifes di komputer lokal Anda. Pada langkah selanjutnya, Anda mengunggah file ke Amazon S3.

File manifes dalam format berikut:

```
{
   "entries": [
     {"endpoint":"<ssh_endpoint_or_IP>",
       "command": "<remote_command>",
       "mandatory":true,
       "publickey": "<public_key>",
       "username": "<host_user_name>"},
     {"endpoint":"<ssh_endpoint_or_IP>",
```

```
        "command": "<remote_command>",
        "mandatory":true,
        "publickey": "<public_key>",
        "username": "host_user_name"}
    ]
}
```

File manifes berisi satu konstruksi “entri” untuk setiap koneksi SSH. Setiap entri mewakili koneksi SSH tunggal. Anda dapat memiliki beberapa koneksi ke satu host atau beberapa koneksi ke beberapa host. Tanda kutip ganda diperlukan seperti yang ditunjukkan, baik untuk nama bidang maupun nilainya. Satu-satunya nilai yang tidak memerlukan tanda kutip ganda adalah nilai Boolean **`true`** atau **`false`** untuk bidang wajib.

Berikut ini menjelaskan bidang dalam file manifes.

titik akhir

Alamat URL atau alamat IP host. Misalnya, "`ec2-111-222-333.compute-1.amazonaws.com`" atau "`22.33.44.56`”

perintah

Perintah yang akan dijalankan oleh host untuk menghasilkan teks atau biner (gzip, lzop, atau bzip2) output. Perintah dapat berupa perintah apa pun yang pengguna “host_user_name” memiliki izin untuk menjalankan. Perintah dapat sesederhana mencetak file, atau dapat menanyakan database atau meluncurkan skrip. Output (file teks, file biner gzip, file biner lzop, atau file biner bzip2) harus dalam bentuk yang dapat dicerna oleh perintah Amazon Redshift COPY. Untuk informasi selengkapnya, lihat Mempersiapkan data input Anda.

kunci publik

(Opsional) Kunci publik dari host. Jika disediakan, Amazon Redshift akan menggunakan kunci publik untuk mengidentifikasi host. Jika kunci publik tidak disediakan, Amazon Redshift tidak akan mencoba identifikasi host. Misalnya, jika kunci publik host jarak jauh adalah:`ssh-rsa AbcCbaxxx…xxxDHKJ root@amazon.com`, masukkan teks berikut di bidang kunci publik:`AbcCbaxxx…xxxDHKJ`.

wajib

(Opsional) Menunjukkan apakah perintah COPY akan gagal jika koneksi gagal. Nilai default-nya `false`. Jika Amazon Redshift tidak berhasil membuat setidaknya satu koneksi, perintah COPY gagal.

nama pengguna

(Opsional) Nama pengguna yang akan digunakan untuk masuk ke sistem host dan menjalankan perintah jarak jauh. Nama login pengguna harus sama dengan login yang digunakan untuk menambahkan kunci publik ke file kunci resmi host di Langkah 2. Nama pengguna default adalah “redshift”.

Contoh berikut menunjukkan manifes lengkap untuk membuka empat koneksi ke host yang sama dan menjalankan perintah yang berbeda melalui setiap koneksi:

```
{
  "entries": [
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata1.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"},
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata2.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"},
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata3.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"},
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata4.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"}
     ]
}
```

Langkah 6: Unggah file manifes ke bucket Amazon S3

Unggah file manifes ke bucket Amazon S3. Jika bucket Amazon S3 tidak berada di Wilayah yang sama AWS dengan cluster Amazon Redshift Anda, Anda harus menggunakan REGION opsi untuk menentukan Wil AWS ayah tempat manifes berada. Untuk informasi tentang membuat bucket

Amazon S3 dan mengunggah file, lihat Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Langkah 7: Jalankan perintah COPY untuk memuat data

Jalan MENYALIN kan perintah untuk terhubung ke host dan memuat data ke tabel Amazon Redshift. Dalam perintah COPY, tentukan jalur objek Amazon S3 eksplisit untuk file manifes dan sertakan opsi SSH. Misalnya,

```
COPY sales
FROM 's3://amzn-s3-demo-bucket/ssh_manifest'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
DELIMITER '|'
SSH;
```

> **Note**
>
> Jika Anda menggunakan kompresi otomatis, perintah COPY melakukan dua pembacaan data, yang berarti menjalankan perintah jarak jauh dua kali. Pembacaan pertama adalah memberikan sampel untuk analisis kompresi, kemudian pembacaan kedua benar-benar memuat data. Jika menjalankan perintah jarak jauh dua kali dapat menyebabkan masalah karena potensi efek samping, Anda harus mematikan kompresi otomatis. Untuk mematikan kompresi otomatis, jalankan perintah COPY dengan opsi COMPDATE disetel ke OFF. Untuk informasi selengkapnya, lihat Memuat tabel dengan kompresi otomatis.

Memuat data dari tabel Amazon DynamoDB

Anda dapat menggunakan perintah COPY untuk memuat tabel dengan data dari satu tabel Amazon DynamoDB.

> **Important**
>
> Tabel Amazon DynamoDB yang menyediakan data harus dibuat di Wilayah yang sama AWS dengan cluster Anda kecuali Anda menggunakan REGION opsi untuk menentukan Wilayah AWS tempat tabel Amazon DynamoDB berada.

Perintah COPY menggunakan arsitektur Amazon Redshift massively parallel processing (MPP) untuk membaca dan memuat data secara paralel dari tabel Amazon DynamoDB. Anda dapat memanfaatkan pemrosesan paralel secara maksimal dengan mengatur gaya distribusi pada tabel Amazon Redshift Anda. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

> **Important**
>
> Saat perintah COPY membaca data dari tabel Amazon DynamoDB, transfer data yang dihasilkan adalah bagian dari throughput yang disediakan tabel tersebut.

Untuk menghindari konsumsi throughput baca yang disediakan dalam jumlah berlebihan, sebaiknya jangan memuat data dari tabel Amazon DynamoDB yang ada di lingkungan produksi. Jika Anda memuat data dari tabel produksi, sebaiknya setel opsi READRATIO jauh lebih rendah daripada persentase rata-rata throughput yang disediakan yang tidak digunakan. Pengaturan READRATIO rendah akan membantu meminimalkan masalah pelambatan. Untuk menggunakan seluruh throughput yang disediakan dari tabel Amazon DynamoDB, setel READRATIO ke 100.

Perintah COPY mencocokkan nama atribut dalam item yang diambil dari tabel DynamoDB dengan nama kolom di tabel Amazon Redshift yang ada dengan menggunakan aturan berikut:

- Kolom tabel Amazon Redshift dicocokkan secara tidak peka huruf besar/kecil dengan atribut item Amazon DynamoDB. Jika item dalam tabel DynamoDB berisi beberapa atribut yang berbeda hanya dalam kasus, seperti Harga dan HARGA, perintah COPY akan gagal.
- Kolom tabel Amazon Redshift yang tidak cocok dengan atribut di tabel Amazon DynamoDB dimuat sebagai NULL atau kosong, tergantung pada nilai yang ditentukan dengan opsi EMPTYASNULL dalam perintah. MENYALIN
- Atribut Amazon DynamoDB yang tidak cocok dengan kolom di tabel Amazon Redshift akan dibuang. Atribut dibaca sebelum dicocokkan, sehingga bahkan atribut yang dibuang menghabiskan sebagian dari throughput yang disediakan tabel tersebut.
- Hanya atribut Amazon DynamoDB dengan tipe data STRING dan NUMBER skalar yang didukung. Jenis data Amazon DynamoDB BINARY dan SET tidak didukung. Jika perintah COPY mencoba memuat atribut dengan tipe data yang tidak didukung, perintah akan gagal. Jika atribut tidak cocok dengan kolom tabel Amazon Redshift, COPY tidak mencoba memuatnya, dan tidak menimbulkan kesalahan.

Perintah COPY menggunakan sintaks berikut untuk memuat data dari tabel Amazon DynamoDB:

```
COPY <redshift_tablename> FROM 'dynamodb://<dynamodb_table_name>'
authorization
readratio '<integer>';
```

Nilai untuk otorisasi adalah AWS kredenSIAL yang diperlukan untuk mengakses tabel Amazon DynamoDB. Jika kredenSIAL ini sesuai dengan pengguna, pengguna tersebut harus memiliki izin untuk MEMINDAI dan MENDESKRIPSI tabel Amazon DynamoDB yang sedang dimuat.

Nilai untuk otorisasi memberikan otor AWS isasi yang dibutuhkan cluster Anda untuk mengakses tabel Amazon DynamoDB. Izin harus menyertakan SCAN dan DESCRIBE untuk tabel Amazon DynamoDB yang sedang dimuat. Untuk informasi lebih lanjut tentang izin yang diperlukan, lihat Izin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY. Metode otentikasi yang lebih disukai adalah menentukan parameter IAM_ROLE dan memberikan Amazon Resource Name (ARN) untuk peran IAM dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Role-based kontrol akses.

Untuk mengotentikasi menggunakan parameter IAM_ROLE, *<aws-account-id>* dan *<role-name>* seperti yang ditunjukkan dalam sintaks berikut.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

Contoh berikut menunjukkan otentikasi menggunakan peran IAM.

```
COPY favoritemovies
FROM 'dynamodb://ProductCatalog'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Untuk informasi selengkapnya tentang opsi otorisasi lainnya, lihat Parameter otorisasi

Jika Anda ingin memvalidasi data Anda tanpa benar-benar memuat tabel, gunakan opsi NOLOAD dengan perintahMENYALIN.

Contoh berikut memuat tabel FAVORITEMOVIES dengan data dari tabel DynamoDB my-favorite-movies-table. Aktivitas baca dapat menghabiskan hingga 50% dari throughput yang disediakan.

```
COPY favoritemovies FROM 'dynamodb://my-favorite-movies-table'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
READRATIO 50;
```

Untuk memaksimalkan throughput, perintah COPY memuat data dari tabel Amazon DynamoDB secara paralel di seluruh node komputasi di cluster.

Throughput yang disediakan dengan kompresi otomatis

Secara default, perintah COPY menerapkan kompresi otomatis setiap kali Anda menentukan tabel target kosong tanpa pengkodean kompresi. Analisis kompresi otomatis awalnya mengambil sampel sejumlah besar baris dari tabel Amazon DynamoDB. Ukuran sampel didasarkan pada nilai parameter COMPROWS. Defaultnya adalah 100.000 baris per irisan.

Setelah pengambilan sampel, baris sampel dibuang dan seluruh tabel dimuat. Akibatnya, banyak baris dibaca dua kali. Untuk informasi selengkapnya tentang cara kerja kompresi otomatis, lihatMemuat tabel dengan kompresi otomatis.

> **Important**
>
> Saat perintah COPY membaca data dari tabel Amazon DynamoDB, termasuk baris yang digunakan untuk pengambilan sampel, transfer data yang dihasilkan adalah bagian dari throughput yang disediakan tabel tersebut.

Memuat data multibyte dari Amazon DynamoDB

Jika data Anda menyertakan karakter multibyte non-ASCII (seperti karakter Mandarin atau Cyrillic), Anda harus memuat data ke kolom VARCHAR. Tipe data VARCHAR mendukung UTF-8 karakter empat byte, tetapi tipe data CHAR hanya menerima karakter ASCII byte tunggal. Anda tidak dapat memuat karakter lima byte atau lebih lama ke tabel Amazon Redshift. Untuk informasi lebih lanjut tentang CHAR dan VARCHAR, lihatJenis Data.

Memverifikasi bahwa data dimuat dengan benar

Setelah operasi pemuatan selesai, kueri tabel STL_LOAD_COMMIT sistem untuk memverifikasi bahwa file yang diharapkan telah dimuat. Jalankan perintah COPY dan memuat verifikasi dalam transaksi yang sama sehingga jika ada masalah dengan beban Anda dapat memutar kembali seluruh transaksi.

Kueri berikut mengembalikan entri untuk memuat tabel di database TICKIT:

```
SELECT query, trim(filename) AS filename, curtime, status
FROM stl_load_commits
WHERE filename like '%tickit%' order by query;
```

```
 query |         filename          |          curtime           | status
-------+---------------------------+----------------------------+--------
 22475 | tickit/allusers_pipe.txt  | 2013-02-08 20:58:23.274186 |      1
 22478 | tickit/venue_pipe.txt     | 2013-02-08 20:58:25.070604 |      1
 22480 | tickit/category_pipe.txt  | 2013-02-08 20:58:27.333472 |      1
 22482 | tickit/date2008_pipe.txt  | 2013-02-08 20:58:28.608305 |      1
 22485 | tickit/allevents_pipe.txt | 2013-02-08 20:58:29.99489  |      1
 22487 | tickit/listings_pipe.txt  | 2013-02-08 20:58:37.632939 |      1
 22489 | tickit/sales_tab.txt      | 2013-02-08 20:58:37.632939 |      1
(6 rows)
```

Memvalidasi data masukan

Untuk memvalidasi data dalam file input Amazon S3 atau tabel Amazon DynamoDB sebelum Anda benar-benar memuat data, gunakan opsi NOLOAD dengan perintah tersebut. MENYALIN Gunakan NOLOAD dengan perintah dan opsi COPY yang sama yang akan Anda gunakan untuk memuat data. NOLOAD memeriksa integritas semua data tanpa memuatnya ke dalam database. Opsi NOLOAD menampilkan kesalahan yang terjadi jika Anda mencoba memuat data.

Misalnya, jika Anda menentukan jalur Amazon S3 yang salah untuk file input, Amazon Redshift akan menampilkan kesalahan berikut.

```
ERROR:  No such file or directory
DETAIL:
-----------------------------------------------
Amazon Redshift error:  The specified key does not exist
code:      2
context:   S3 key being read :
location:  step_scan.cpp:1883
process:   xenmaster [pid=22199]
-----------------------------------------------
```

Untuk memecahkan masalah pesan kesalahan, lihatReferensi kesalahan muat.

Untuk contoh menggunakan opsi NOLOAD, lihat. Perintah COPY dengan opsi NOLOAD

Memuat tabel dengan kompresi otomatis

Anda dapat menerapkan pengkodean kompresi ke kolom dalam tabel secara manual, berdasarkan evaluasi data Anda sendiri. Atau Anda dapat menggunakan perintah COPY dengan COMPUPDATE

disetel ke ON untuk menganalisis dan menerapkan kompresi secara otomatis berdasarkan data sampel.

Anda dapat menggunakan kompresi otomatis saat Anda membuat dan memuat tabel baru. Perintah COPY melakukan analisis kompresi. Anda juga dapat melakukan analisis kompresi tanpa memuat data atau mengubah kompresi pada tabel dengan menjalankan MENGANALISIS KOMPRESI perintah pada tabel yang sudah diisi. Misalnya, Anda dapat menjalankan ANALYZE COMPRESSION ketika Anda ingin menganalisis kompresi pada tabel untuk penggunaan di masa mendatang, sambil mempertahankan pernyataan bahasa definisi data (DDL) yang ada.

Kompresi otomatis menyeimbangkan kinerja keseluruhan saat memilih pengkodean kompresi. Range-restricted pemindaian mungkin berkinerja buruk jika kolom kunci pengurutan dikompresi jauh lebih tinggi daripada kolom lain dalam kueri yang sama. Akibatnya, kompresi otomatis melewatkan fase analisis data pada kolom kunci pengurutan dan mempertahankan jenis pengkodean yang ditentukan pengguna.

Kompresi otomatis memilih penyandian RAW jika Anda belum secara eksplisit mendefinisikan jenis pengkodean. ANALISIS COMPRESSION berperilaku sama. Untuk kinerja kueri yang optimal, pertimbangkan untuk menggunakan RAW untuk kunci pengurutan.

Cara kerja kompresi otomatis

Ketika parameter COMPUPDATE AKTIF, perintah COPY menerapkan kompresi otomatis setiap kali Anda menjalankan perintah COPY dengan tabel target kosong dan semua kolom tabel memiliki pengkodean RAW atau tidak ada pengkodean.

Untuk menerapkan kompresi otomatis ke tabel kosong, terlepas dari pengkodean kompresi saat ini, jalankan perintah COPY dengan opsi COMPDATE disetel ke ON. Untuk mematikan kompresi otomatis, jalankan perintah COPY dengan opsi COMPDATE disetel ke OFF.

Anda tidak dapat menerapkan kompresi otomatis ke tabel yang sudah berisi data.

> **Note**
>
> Analisis kompresi otomatis membutuhkan baris yang cukup dalam data beban (setidaknya 100.000 baris per irisan) untuk menghasilkan sampel yang bermakna.

Kompresi otomatis melakukan operasi ini di latar belakang sebagai bagian dari transaksi beban:

1. Sampel awal baris dimuat dari file input. Ukuran sampel didasarkan pada nilai parameter COMPROWS. Defaultnya adalah 100.000.
2. Opsi kompresi dipilih untuk setiap kolom.
3. Baris sampel dihapus dari tabel.
4. Tabel dibuat ulang dengan pengkodean kompresi yang dipilih.
5. Seluruh file input dimuat dan dikompresi menggunakan pengkodean baru.

Setelah Anda menjalankan perintah COPY, tabel dimuat penuh, dikompresi, dan siap digunakan. Jika Anda memuat lebih banyak data nanti, baris ditambahkan dikompresi sesuai dengan pengkodean yang ada.

Jika Anda hanya ingin melakukan analisis kompresi, jalankan ANALYZE COMPRESSION, yang lebih efisien daripada menjalankan COPY penuh. Kemudian Anda dapat mengevaluasi hasil untuk memutuskan apakah akan menggunakan kompresi otomatis atau membuat ulang tabel secara manual.

Kompresi otomatis hanya didukung untuk perintah COPY. Atau, Anda dapat menerapkan pengkodean kompresi secara manual saat Anda membuat tabel. Untuk informasi tentang pengkodean kompresi manual, lihatKompresi kolom untuk mengurangi ukuran data yang disimpan.

Contoh kompresi otomatis

Dalam contoh ini, asumsikan bahwa database TICKIT berisi salinan tabel LISTING yang disebut BIGLIST, dan Anda ingin menerapkan kompresi otomatis ke tabel ini ketika dimuat dengan sekitar 3 juta baris.

Untuk memuat dan secara otomatis mengompres tabel

1. Pastikan meja kosong. Anda dapat menerapkan kompresi otomatis hanya ke tabel kosong:

   ```
   TRUNCATE biglist;
   ```

2. Muat tabel dengan perintah COPY tunggal. Meskipun tabel kosong, beberapa pengkodean sebelumnya mungkin telah ditentukan. Untuk memudahkan Amazon Redshift melakukan analisis kompresi, setel parameter COMPUPDATE ke ON.

   ```
   COPY biglist FROM 's3://amzn-s3-demo-bucket/biglist.txt'
   IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
   ```

```
DELIMITER '|' COMPUPDATE ON;
```

Karena tidak ada opsi COMPROWS yang ditentukan, ukuran sampel default dan yang disarankan 100.000 baris per irisan digunakan.

3. Lihatlah skema baru untuk tabel BIGLIST untuk meninjau skema pengkodean yang dipilih secara otomatis.

```
SELECT "column", type, encoding
from pg_table_def where tablename = 'biglist';


     Column     |            Type             | Encoding
----------------+-----------------------------+----------
 listid         | integer                     | az64
 sellerid       | integer                     | az64
 eventid        | integer                     | az64
 dateid         | smallint                    | none
 numtickets     | smallint                    | az64
 priceperticket | numeric(8,2)                | az64
 totalprice     | numeric(8,2)                | az64
 listtime       | timestamp without time zone | az64
```

4. Verifikasi bahwa jumlah baris yang diharapkan telah dimuat:

```
select count(*) from biglist;

count
---------
3079952
(1 row)
```

Ketika baris kemudian ditambahkan ke tabel ini menggunakan pernyataan COPY atau INSERT, pengkodean kompresi yang sama diterapkan.

Mengoptimalkan penyimpanan untuk tabel sempit

Jika Anda memiliki tabel dengan kolom yang sangat sedikit tetapi jumlah baris yang sangat besar, tiga kolom identitas metadata tersembunyi (INSERT_XID, DELETE_XID, ROW_ID) akan menghabiskan jumlah ruang disk yang tidak proporsional untuk tabel.

Untuk mengoptimalkan kompresi kolom tersembunyi, muat tabel dalam satu transaksi COPY jika memungkinkan. Jika Anda memuat tabel dengan beberapa perintah COPY terpisah, kolom INSERT_XID tidak akan dikompres dengan baik. Anda harus melakukan operasi vakum jika Anda menggunakan beberapa perintah COPY, tetapi itu tidak akan meningkatkan kompresi INSERT_XID.

Memuat nilai kolom default

Anda dapat secara opsional menentukan daftar kolom dalam perintah COPY Anda. Jika kolom dalam tabel dihilangkan dari daftar kolom, COPY akan memuat kolom dengan nilai yang disediakan oleh opsi DEFAULT yang ditentukan dalam perintah CREATE TABLE, atau dengan NULL jika opsi DEFAULT tidak ditentukan.

Jika COPY mencoba menetapkan NULL ke kolom yang didefinisikan sebagai NOT NULL, perintah COPY gagal. Untuk informasi tentang menetapkan opsi DEFAULT, lihatCREATE TABLE.

Saat memuat dari file data di Amazon S3, kolom dalam daftar kolom harus dalam urutan yang sama dengan bidang dalam file data. Jika bidang dalam file data tidak memiliki kolom yang sesuai dalam daftar kolom, perintah COPY gagal.

Saat memuat dari tabel Amazon DynamoDB, urutan tidak masalah. Setiap bidang dalam atribut Amazon DynamoDB yang tidak cocok dengan kolom di tabel Amazon Redshift akan dibuang.

Batasan berikut berlaku saat menggunakan perintah COPY untuk memuat nilai DEFAULT ke dalam tabel:

- Jika IDENTITY kolom disertakan dalam daftar kolom, opsi EXPLICIT_IDS juga harus ditentukan dalam MENYALIN perintah, atau perintah COPY akan gagal. Demikian pula, jika kolom IDENTITY dihilangkan dari daftar kolom, dan opsi EXPLICIT_IDS ditentukan, operasi COPY akan gagal.
- Karena ekspresi DEFAULT yang dievaluasi untuk kolom tertentu sama untuk semua baris yang dimuat, ekspresi DEFAULT yang menggunakan fungsi RANDOM () akan menetapkan nilai yang sama untuk semua baris.
- Ekspresi DEFAULT yang berisi CURRENT_DATE atau SYSDATE diatur ke stempel waktu transaksi saat ini.

Sebagai contoh, lihat “Memuat data dari file dengan nilai default” diContoh COPY.

Memecahkan masalah beban data

Saat memuat data ke tabel Amazon Redshift, Anda mungkin mengalami kesalahan dari Amazon S3, data input yang tidak valid, dan kesalahan perintah COPY. Bagian berikut memberikan informasi tentang mengidentifikasi dan menyelesaikan kesalahan pemuatan data.

Topik

- Memecahkan masalah integrasi peristiwa S3 dan kesalahan COPY JOB
- S3ServiceException kesalahan
- Tabel sistem untuk pemecahan masalah beban data
- Kesalahan pemuatan karakter multibyte
- Referensi kesalahan muat

Memecahkan masalah integrasi peristiwa S3 dan kesalahan COPY JOB

Gunakan informasi berikut untuk memecahkan masalah umum dengan integrasi acara Amazon S3 dan COPY JOB dengan Amazon Redshift.

Pembuatan integrasi acara S3 gagal

Jika pembuatan integrasi peristiwa S3 gagal, status integrasi adalah`Inactive`. Pastikan bahwa berikut ini benar untuk gudang data Amazon Redshift Anda.

- Anda menambahkan sumber prinsip dan integrasi resmi yang benar untuk namespace target Anda di Amazon Redshift. Lihat Prasyarat untuk membuat integrasi acara S3.
- Anda menambahkan kebijakan berbasis sumber daya yang benar ke bucket Amazon S3 sumber. Lihat Prasyarat untuk membuat integrasi acara S3.

Data Amazon S3 Anda tidak muncul di database target

Jika data dari COPY JOB tidak muncul, periksa yang berikut ini.

- Kueri SYS_COPY_JOB_DETAIL untuk melihat apakah file Amazon S3 telah dimuat, apakah tertunda penyerapan, atau ada kesalahan. Untuk informasi selengkapnya, lihat SYS_COPY_JOB_DETAIL.

- Konsultasikan STL_ERROR atau SYS_COPY_JOB_INFO jika file Amazon S3 tidak ada atau ada waktu tunggu yang tidak terduga. Cari kesalahan kredensi atau apa pun yang menunjukkan integrasi tidak aktif. Untuk informasi selengkapnya, lihat STL_ERROR dan SYS_COPY_JOB_INFO.

S3ServiceException kesalahan

Kes ServiceException alahan s3 yang paling umum disebabkan oleh string kredenSIAL yang diformat secara tidak benar atau salah, memiliki cluster dan bucket Anda di Wil AWS ayah yang berbeda, dan izin Amazon S3 yang tidak mencukupi.

Bagian ini menyediakan informasi pemecahan masalah untuk setiap jenis kesalahan.

String kredensional tidak valid

Jika string kredensial Anda tidak diformat dengan benar, Anda akan menerima pesan kesalahan berikut:

```
ERROR: Invalid credentials. Must be of the format: credentials
'aws_access_key_id=<access-key-id>;aws_secret_access_key=<secret-access-key>
[;token=<temporary-session-token>]'
```

Verifikasi bahwa string kredenSIAL tidak mengandung spasi atau jeda baris, dan tertutup dalam tanda kutip tunggal.

ID kunci akses tidak valid

Jika ID kunci akses Anda tidak ada, Anda akan menerima pesan kesalahan berikut:

```
[Amazon](500310) Invalid operation: S3ServiceException:The AWS Access Key Id you
 provided does not exist in our records.
```

Ini sering merupakan kesalahan salin dan tempel. Pastikan ID kunci akses dimasukkan dengan benar. Juga, jika Anda menggunakan kunci sesi sementara, periksa apakah nilai untuk `token` ditetapkan.

Kunci akses rahasia tidak valid

Jika kunci akses rahasia Anda salah, Anda akan menerima pesan kesalahan berikut:

```
[Amazon](500310) Invalid operation: S3ServiceException:The request signature we
 calculated does not match the signature you provided.
```

```
Check your key and signing method.,Status 403,Error SignatureDoesNotMatch
```

Ini sering merupakan kesalahan salin dan tempel. Verifikasi bahwa kunci akses rahasia dimasukkan dengan benar dan itu adalah kunci yang benar untuk ID kunci akses.

Bucket berada di wilayah yang berbeda

Bucket Amazon S3 yang ditentukan dalam perintah COPY harus berada di Wil AWS ayah yang sama dengan cluster. Jika bucket Amazon S3 dan cluster Anda berada di Wilayah yang berbeda, Anda akan menerima kesalahan yang mirip dengan berikut ini:

```
ERROR: S3ServiceException:The bucket you are attempting to access must be addressed
 using the specified endpoint.
```

Anda dapat membuat bucket Amazon S3 di Wilayah tertentu dengan memilih Wilayah saat membuat bucket menggunakan Amazon S3 Management Console, atau dengan menentukan titik akhir saat Anda membuat bucket menggunakan API atau CLI Amazon S3. Untuk informasi selengkapnya, lihat Mengunggah file ke Amazon S3 untuk digunakan dengan COPY.

Untuk informasi selengkapnya tentang Wilayah Amazon S3, lihat Mengakses Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Atau, Anda dapat menentukan Wilayah menggunakan REGION opsi dengan perintah COPY.

Akses ditolak

Jika pengguna tidak memiliki izin yang memadai, Anda akan menerima pesan kesalahan berikut:

```
ERROR: S3ServiceException:Access Denied,Status 403,Error AccessDenied
```

Salah satu kemungkinan penyebabnya adalah pengguna yang diidentifikasi oleh kredenSIAL tidak memiliki LIST dan akses GET ke bucket Amazon S3. Untuk penyebab lainnya, lihat Memec ahkan Masalah Kesalahan Akses Ditolak (403 Dilarang) di Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk informasi tentang mengelola akses pengguna ke bucket, lihat Identitas dan manajemen akses di Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Tabel sistem untuk pemecahan masalah beban data

Tabel sistem Amazon Redshift berikut dapat membantu dalam memecahkan masalah pemuatan data:

- Kueri STL_LOAD_ERROR untuk menemukan kesalahan yang terjadi selama pemuatan tertentu.
- Kueri STL_FILE_SCAN untuk melihat waktu muat untuk file tertentu atau untuk melihat apakah file tertentu bahkan dibaca.
- Kueri STL_S3CLIENT_ERROR untuk menemukan detail kesalahan yang ditemui saat mentransfer data dari Amazon S3.

Untuk menemukan dan mendiagnosis kesalahan beban

1. Buat tampilan atau tentukan kueri yang mengembalikan detail tentang kesalahan pemuatan. Contoh berikut menggabungkan tabel STL_LOAD_ERRORS ke tabel STV_TBL_PERM untuk mencocokkan ID tabel dengan nama tabel aktual.

   ```
   create view loadview as
   (select distinct tbl, trim(name) as table_name, query, starttime,
   trim(filename) as input, line_number, colname, err_code,
   trim(err_reason) as reason
   from stl_load_errors sl, stv_tbl_perm sp
   where sl.tbl = sp.id);
   ```

2. Setel opsi MAXERRORS dalam perintah COPY Anda ke nilai yang cukup besar untuk mengaktifkan COPY untuk mengembalikan informasi yang berguna tentang data Anda. Jika COPY menemukan kesalahan, pesan kesalahan mengarahkan Anda untuk berkonsultasi dengan tabel STL_LOAD_ERRORS untuk detailnya.
3. Kueri tampilan LOADVIEW untuk melihat detail kesalahan. Contoh:

   ```
   select * from loadview where table_name='venue';
   ```

   ```
     tbl   | table_name | query |         starttime
   --------+------------+-------+---------------------------
    100551 | venue      | 20974 | 2013-01-29 19:05:58.365391

   |     input      | line_number | colname | err_code |       reason
   +----------------+-------------+---------+----------+--------------------
   ```

```
| venue_pipe.txt |          1 |       0 |     1214 | Delimiter not found
```

4. Perbaiki masalah dalam file input atau skrip muat, berdasarkan informasi yang dikembalikan tampilan. Beberapa kesalahan pemuatan khas yang harus diperhatikan meliputi:

   - Ketidakcocokan antara tipe data dalam tabel dan nilai di bidang data input.
   - Ketidakcocokan antara jumlah kolom dalam tabel dan jumlah bidang dalam data input.
   - Tanda kutip tidak cocok. Amazon Redshift mendukung tanda kutip tunggal dan ganda; Namun, tanda kutip ini harus seimbang dengan tepat.
   - Format salah untuk date/time data dalam file input.
   - Out-of-range nilai dalam file input (untuk kolom numerik).
   - Jumlah nilai berbeda untuk kolom melebihi batasan untuk pengkodean kompresi.

Kesalahan pemuatan karakter multibyte

Kolom dengan tipe data CHAR hanya menerima UTF-8 karakter byte tunggal, hingga nilai byte 127, atau 7F hex, yang juga merupakan set karakter ASCII. Kolom VARCHAR menerima UTF-8 karakter multibyte, hingga maksimum empat byte. Untuk informasi selengkapnya, lihat Jenis karakter.

Jika baris dalam data pemuatan Anda berisi karakter yang tidak valid untuk tipe data kolom, COPY mengembalikan kesalahan dan mencatat baris dalam tabel log sistem STL_LOAD_ERRORS dengan nomor kesalahan 1220. Bidang ERR_REASON menyertakan urutan byte, dalam hex, untuk karakter yang tidak valid.

Alternatif untuk memperbaiki karakter yang tidak valid dalam data pemuatan Anda adalah mengganti karakter yang tidak valid selama proses pemuatan. Untuk mengganti UTF-8 karakter yang tidak valid, tentukan opsi ACCEPTINVCHARS dengan perintah COPY. Jika opsi ACCEPTINVCHARS disetel, karakter yang Anda tentukan menggantikan titik kode. Jika opsi ACCEPTINVCHARS tidak disetel, Amazon Redshift menerima karakter sebagai valid. UTF-8 Untuk informasi selengkapnya, lihat ACCEPTINVCHARS.

Daftar titik kode berikut ini valid UTF-8, operasi COPY tidak mengembalikan kesalahan jika opsi ACCEPTINVCHARS tidak disetel. Namun, titik kode ini bukan karakter yang valid. Anda dapat menggunakan ACCEPTINVCHARS opsi untuk mengganti titik kode dengan karakter yang Anda tentukan. Poin kode ini mencakup rentang nilai dari `0xFDD0` ke `0xFDEF` dan nilai hingga`0x10FFFF`, diakhiri dengan `FFFE` atau`FFFF`:

- `0xFFFE`, `0x1FFFE`, `0x2FFFE`, …, `0xFFFFE`, `0x10FFFE`

- `0xFFFF`, `0x1FFFF`, `0x2FFFF`, ..., `0xFFFFF`, `0x10FFFF`

Contoh berikut menunjukkan alasan kesalahan saat COPY mencoba memuat UTF-8 karakter `e0 a1 c7a4` ke kolom CHAR.

```
Multibyte character not supported for CHAR
(Hint: Try using  VARCHAR). Invalid char: e0 a1 c7a4
```

Jika kesalahan terkait dengan tipe data VARCHAR, alasan kesalahan mencakup kode kesalahan serta urutan UTF-8 hex yang tidak valid. Contoh berikut menunjukkan alasan kesalahan ketika COPY mencoba memuat UTF-8 `a4` ke bidang VARCHAR.

```
String contains invalid or unsupported UTF-8 codepoints.
Bad UTF-8 hex sequence: a4 (error 3)
```

Tabel berikut mencantumkan deskripsi dan solusi yang disarankan untuk kesalahan pemuatan VARCHAR. Jika salah satu kesalahan ini terjadi, ganti karakter dengan urutan UTF-8 kode yang valid atau hapus karakter.

Kode kesalahan	Deskripsi
1	Ur UTF-8 utan byte melebihi maksimum empat byte yang didukung oleh VARCHAR.
2	Ur UTF-8 utan byte tidak lengkap. COPY tidak menemukan jumlah byte lanjutan yang diharapkan untuk karakter multibyte sebelum akhir string.
3	Karakter UTF-8 single-byte berada di luar jangkauan. Byte awal tidak boleh 254, 255 atau karakter apa pun antara 128 dan 191 (inklusif).
4	Nilai byte trailing dalam urutan byte berada di luar jangkauan. Byte lanjutan harus antara 128 dan 191 (inklusif).
5	UTF-8 Karakter dicadangkan sebagai pengganti. Titik kode pengganti (U+D800 hingga U+DFFF) tidak valid.
8	Urutan byte melebihi titik UTF-8 kode maksimum.

Kode kesalahan	Deskripsi
9	Ur UTF-8 utan byte tidak memiliki titik kode yang cocok.

Referensi kesalahan muat

Jika ada kesalahan yang terjadi saat memuat data dari file, kueri STL_LOAD_ERROR tabel untuk mengidentifikasi kesalahan dan menentukan penjelasan yang mungkin. Tabel berikut mencantumkan semua kode kesalahan yang mungkin terjadi selama pemuatan data:

Muat kode kesalahan

Kode kesalahan	Deskripsi
1200	Kesalahan penguraian tidak diketahui. Hubungi support.
1201	Pembatas bidang tidak ditemukan dalam file input.
1202	Data input memiliki lebih banyak kolom daripada yang didefinisikan dalam DDL.
1203	Data input memiliki lebih sedikit kolom daripada yang didefinisikan dalam DDL.
1204	Data input melebihi rentang yang dapat diterima untuk tipe data.
1205	Format tanggal tidak valid. Lihat String DATEFORMAT dan TIMEFORMAT untuk format yang valid.
1206	Format stempel waktu tidak valid. Lihat String DATEFORMAT dan TIMEFORMAT untuk format yang valid.
1207	Data berisi nilai di luar kisaran yang diharapkan 0-9.
1208	Kesalahan format tipe data FLOAT.
1209	Kesalahan format tipe data DESIMAL.
1210	Kesalahan format tipe data BOOLEAN.

Kode kesalahan	Deskripsi
1211	Baris input tidak berisi data.
1212	File muat tidak ditemukan.
1213	Bidang yang ditentukan sebagai NOT NULL tidak berisi data.
1214	Pembatas tidak ditemukan.
1215	Kesalahan bidang CHAR.
1216	Baris input tidak valid.
1217	Nilai kolom identitas tidak valid.
1218	Saat menggunakan NULL AS '\ 0', bidang yang berisi terminator null (NUL, atau UTF-8 0000) berisi lebih dari satu byte.
1219	UTF-8 heksadesimal berisi digit yang tidak valid.
1220	String berisi poin UTF-8 kode yang tidak valid atau tidak didukung.
1221	Pengkodean file tidak sama dengan yang ditentukan dalam perintah COPY.
1222	Kesalahan meluap nilai integer.
1223	Jenis data tidak valid.
1224	Input data tidak terbentuk dengan baik format JSON untuk tipe data super.
8001	COPY dengan parameter MANIFEST memerlukan jalur penuh objek Amazon S3.
9005	Kunci akhir tidak valid yang ditentukan.

Buat integrasi acara S3 untuk menyalin file secara otomatis dari bucket Amazon S3

Anda dapat menggunakan pekerjaan penyalinan otomatis untuk memuat data ke tabel Amazon Redshift dari file yang disimpan di Amazon S3. Amazon Redshift mendeteksi ketika file Amazon S3 baru ditambahkan ke jalur yang ditentukan dalam perintah COPY Anda. Perintah COPY kemudian dijalankan secara otomatis tanpa Anda harus membuat pipeline penyerapan data eksternal. Amazon Redshift melacak file mana yang telah dimuat. Amazon Redshift menentukan jumlah file yang digabungkan bersama per perintah COPY. Anda dapat melihat perintah COPY yang dihasilkan dalam tampilan sistem.

Langkah pertama untuk membuat COPY JOB otomatis adalah membuat integrasi acara S3. Saat file baru muncul di bucket sumber Amazon S3, Amazon Redshift kemudian mengelola pemuatan file ke database Anda menggunakan perintah COPY.

Prasyarat untuk membuat integrasi acara S3

Untuk mengatur integrasi acara s3 Anda, konfirmasikan prasyarat berikut telah selesai.

- Bucket Amazon S3 Anda harus memiliki kebijakan bucket yang mengizinkan beberapa izin Amazon S3. Misalnya, kebijakan contoh berikut mengizinkan izin untuk bucket sumber daya `amzn-s3-demo-bucket` yang dihosting*`us-east-1`*. Baik bucket Amazon S3 dan integrasinya sama Wilayah AWS.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Auto-Copy-Policy-01",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
                },
            "Action": [
                "s3:GetBucketNotification",
                "s3:PutBucketNotification",
                "s3:GetBucketLocation"
            ],
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket:*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:redshift:us-
east-1:111122223333:integration:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

- Cluster target yang disediakan Amazon Redshift atau namespace Redshift Serverless Anda harus memiliki izin ke bucket. Konfirmasikan peran IAM yang terkait dengan cluster atau ruang nama tanpa server Anda memiliki kebijakan IAM yang mengizinkan izin yang tepat. Kebijakan harus mengizinkan baik `s3:GetObject` untuk sumber daya bucket seperti *amzn-s3-demo-bucket* dan `s3:ListBucket` untuk sumber daya bucket dan isinya seperti*amzn-s3-demo-bucket*/*.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AutoCopyReadId",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

  Menambahkan kebijakan Anda ke peran IAM yang memiliki hubungan kepercayaan untuk peran tersebut adalah sebagai berikut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "redshift.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Jika gudang data target Anda adalah cluster yang disediakan, Anda dapat mengaitkan peran IAM ke cluster yang disediakan menggunakan konsol Amazon Redshift, tab Izin klaster di detail cluster Anda. Untuk informasi tentang cara mengaitkan peran ke cluster yang disediakan, lihat Mengaitkan peran IAM dengan cluster di Panduan Manajemen Amazon Redshift.

Jika gudang data target Anda adalah Redshift Serverless, Anda dapat mengaitkan peran IAM ke namespace tanpa server menggunakan tab Konsol Redshift Serverless, Keamanan dan enkripsi di detail namespace Anda. Untuk informasi tentang cara mengaitkan peran ke namespace tanpa server Anda, lihat Memberikan izin ke Amazon Redshift Serverless di Panduan Manajemen Amazon Redshift.

- Gudang data Amazon Redshift Anda juga harus memiliki kebijakan sumber daya yang memungkinkan bucket Amazon S3. Jika Anda menggunakan konsol Amazon Redshift, saat Anda membuat integrasi peristiwa s3, Amazon Redshift menyediakan opsi Per baiki agar saya menambahkan kebijakan ini ke gudang data Amazon Redshift Anda. Untuk memperbarui kebijakan sumber daya sendiri, Anda dapat menggunakan perintah put-resour AWS CLI ce-policy. Misalnya, untuk melampirkan kebijakan sumber daya ke cluster yang disediakan Amazon Redshift untuk integrasi peristiwa S3 dengan bucket Amazon S3, jalankan AWS CLI perintah yang serupa dengan berikut ini. Contoh berikut menunjukkan kebijakan untuk namespace cluster yang disediakan di akun pengguna *us-east-1* Wilayah AWS untuk. *123456789012* Ember itu diberi nama*amzn-s3-demo-bucket*.

```
aws redshift put-resource-policy \
```

```
--policy file://rs-rp.json \
--resource-arn "arn:aws:redshift: us-east-1:123456789012:namespace/cc4ffe56-
ad2c-4fd1-a5a2-f29124a56433"
```

Di mana `rs-rp.json` berisi:

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": {
    "Service": "redshift.amazonaws.com"
   },
   "Action": "redshift:AuthorizeInboundIntegration",
   "Resource": "arn:aws:redshift:us-east-1:123456789012:namespace:cc4ffe56-
ad2c-4fd1-a5a2-f29124a56433",
   "Condition": {
    "StringEquals": {
     "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket",
     "aws:SourceAccount": 111122223333
    }
   }
  },
  {
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/myRedshiftRole"
   },
   "Action": "redshift:CreateInboundIntegration",
   "Resource": "arn:aws:redshift:us-east-1:123456789012:namespace:cc4ffe56-
ad2c-4fd1-a5a2-f29124a56433",
   "Condition": {
    "StringEquals": {
     "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket",
     "aws:SourceAccount": 111122223333
    }
   }
  }
 ]
}
```

Untuk melampirkan kebijakan sumber daya ke namespace Redshift Serverless Anda untuk integrasi peristiwa S3 dengan bucket Amazon S3, jalankan AWS CLI perintah yang serupa dengan berikut ini. Contoh berikut menunjukkan kebijakan untuk namespace tanpa server di *us-east-1* Wilayah AWS untuk akun pengguna. *123456789012* Ember itu diberi nama*amzn-s3-demo-bucket*.

```
aws redshift put-resource-policy \
--policy file://rs-rp.json \
--resource-arn "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/
namespace-1"
```

Di mana `rs-rp.json` berisi:

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": {
    "Service": "redshift.amazonaws.com"
   },
   "Action": "redshift:AuthorizeInboundIntegration",
   "Resource": "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/
namespace-1",
   "Condition": {
    "StringEquals": {
     "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket",
     "aws:SourceAccount": 111122223333

    }
   }
  },
  {
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::123456789012:user/myUser"
   },
   "Action": "redshift:CreateInboundIntegration",
```

```
   "Resource": "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/
namespace-1",
   "Condition": {
    "StringEquals": {
     "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket",
     "aws:SourceAccount": 111122223333
    }
   }
  }
 ]
}
```

Membuat integrasi acara S3

Untuk mengatur pekerjaan penyalinan Anda, pertama-tama Anda menentukan integrasi peristiwa S3.

Amazon Redshift console

Cara membuat integrasi acara Amazon S3 di konsol Amazon Redshift

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Di panel navigasi kiri, pilih integrasi acara S3.
3. Pilih Buat integrasi acara Amazon S3 untuk membuka wizard untuk membuat dan integrasi acara S3 untuk digunakan dengan salin otomatis. Bucket Amazon S3 sumber Anda dan gudang data Amazon Redshift target harus berada di tempat yang sama Wilayah AWS. Tentukan informasi berikut saat melalui langkah-langkah untuk membuat integrasi:
   - Nama integrasi — Merupakan pengidentifikasi unik di semua integrasi yang dimiliki oleh Anda Akun AWS saat ini Wilayah AWS.
   - Deskripsi — Adalah teks yang menjelaskan integrasi acara Amazon S3 untuk referensi nanti.
   - Bucket Source S3 - Adalah bucket Amazon S3 saat ini Akun AWS dan Wilayah AWS yang merupakan sumber penyerapan data ke Amazon Redshift.
   - Gudang data Amazon Redshift — Adalah cluster yang disediakan Amazon Redshift target atau kelompok kerja Redshift Serverless yang menerima data dari integrasi.

Jika target Amazon Redshift Anda ada di akun yang sama, Anda dapat memilih target. Jika target berada di akun yang berbeda, Anda menentukan gudang data Amazon Redshift AR N. Target harus memiliki kebijakan sumber daya dengan prinsipal resmi dan sumber integrasi. Jika Anda tidak memiliki kebijakan sumber daya yang benar pada target dan target Anda berada di akun yang sama, Anda dapat memilih opsi Per baiki untuk saya untuk menerapkan kebijakan sumber daya secara otomatis selama proses integrasi buat. Jika target Anda berbeda Akun AWS, Anda perlu menerapkan kebijakan sumber daya di gudang Amazon Redshift secara manual.

4. Masukkan hingga 50 K unci tag dan dengan Nilai opsional — Untuk memberikan metadata tambahan tentang integrasi.
5. Halaman ulasan ditampilkan di mana Anda dapat memilih Buat integrasi acara S3.

AWS CLI

Untuk membuat integrasi acara Amazon S3 menggunakan AWS CLI, gunakan `create-integration` perintah dengan opsi berikut:

- `integration-name` – Tentukan nama untuk integrasi.
- `source-arn`— Tentukan ARN dari bucket sumber Amazon S3.
- `target-arn`— Tentukan namespace ARN dari cluster yang disediakan Amazon Redshift atau target kelompok kerja Redshift Serverless.

Contoh berikut membuat integrasi dengan memberikan nama integrasi, sumber ARN, dan target ARN. Integrasi tidak dienkripsi.

```
aws redshift create-integration \
--integration-name s3-integration \
--source-arn arn:aws:s3:us-east-1::s3-example-bucket \
--target-arn arn:aws:redshift:us-east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
          {
    "IntegrationArn": "arn:aws:redshift:us-east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "IntegrationName": "s3-integration",
    "SourceArn": "arn:aws:s3:::s3-example-bucket",
    "SourceType": "s3-event-notifications",
```

```
    "TargetArn": "arn:aws:redshift:us-
east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Status": "creating",
    "Errors": [],
    "CreateTime": "2024-10-09T19:08:52.758000+00:00",
    "Tags": []
}
```

Anda juga dapat menggunakan AWS CLI perintah berikut untuk mengelola integrasi acara S3 Anda.

- `delete-integration`— Tentukan integrasi ARN untuk menghapus integrasi peristiwa S3.
- `modify-integration`— Tentukan integrasi ARN untuk mengubah nama atau deskripsi (atau keduanya) integrasi peristiwa S3.
- `describe-integrations`— Tentukan integrasi ARN untuk melihat properti integrasi peristiwa S3.

Lihat Panduan Amazon Redshift CLI untuk informasi selengkapnya tentang perintah ini.

Amazon Redshift kemudian membuat integrasi peristiwa S3 dengan sumber dan target terkait, status, dan informasi tentang status pekerjaan penyalinan otomatis terkait. Anda dapat melihat informasi tentang integrasi acara S3 di konsol Amazon Redshift dengan memilih integrasi acara S3, dan memilih integrasi untuk menampilkan detailnya. Integrasi dipisahkan oleh yang dibuat di akun saya dan Dari akun lain. Daftar Di akun saya menunjukkan integrasi di mana sumber dan target berada di akun yang sama. Daftar Dari akun lain menunjukkan integrasi di mana sumbernya dimiliki oleh akun lain.

Jika Anda menghapus integrasi peristiwa S3, status COPY JOB yang sesuai berubah dari 1 (aktif) menjadi `0` (inactive/pending). Namun, COPY JOB yang sesuai tidak secara otomatis dihapus. Jika nanti Anda mencoba membuat COPY JOB dengan nama yang sama, mungkin ada konflik.

Membuat dan memantau COPY JOB

Setelah integrasi dibuat, pada halaman detail integrasi peristiwa S3 untuk integrasi yang Anda buat, pilih Buat pekerjaan penyalinan otomatis untuk membuka editor kueri Amazon Redshift v2 di mana Anda dapat membuat pekerjaan penyalinan otomatis untuk integrasi. Amazon Redshift mencocokkan bucket dalam klausa FROM dalam pernyataan COPY JOB CREATE dengan bucket yang digunakan dalam integrasi peristiwa S3. Untuk informasi tentang cara menggunakan editor kueri

Amazon Redshift v2, lihat Meng kueri database menggunakan editor kueri Amazon Redshift v2 di Panduan Manajemen Amazon Redshift. Misalnya, jalankan perintah COPY berikut di query editor v2 untuk membuat COPY JOB otomatis yang cocok dengan bucket `s3://amzn-s3-demo-bucket/staging-folder` Amazon S3 dengan integrasi acara Amazon S3.

```
COPY public.target_table
FROM 's3://amzn-s3-demo-bucket/staging-folder'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyLoadRoleName'
JOB CREATE my_copy_job_name
AUTO ON;
```

Anda mendefinisikan COPY JOB satu kali. Parameter yang sama digunakan untuk proses di masa mendatang.

Untuk menentukan dan mengelola COPY JOB, Anda harus memiliki izin. Untuk informasi tentang pemberian dan pencabutan izin pada COPY JOB, lihat HIBAH danMENCABUT. Untuk informasi selengkapnya tentang pemberian dan pencabutan izin cakupan untuk COPY JOB, lihat Pemberian izin bercakupan dan. Mencabut izin yang dicakup

Anda mengelola operasi pemuatan menggunakan opsi untuk CREATE, LIST, SHOW, DROP, ALTER, dan RUN. Untuk informasi selengkapnya, lihat SALIN PEKERJAAN.

Anda dapat menanyakan tampilan sistem untuk melihat status dan kemajuan COPY JOB. Tampilan disediakan sebagai berikut:

- SYS_COPY_JOB— berisi baris untuk setiap COPY JOB yang didefinisikan saat ini.
- SYS_COPY_JOB_DETAIL— berisi detail tentang file yang tertunda, kesalahan, dan dicerna untuk setiap COPY JOB.
- SYS_COPY_JOB_INFO— berisi pesan yang dicatat tentang COPY JOB.
- SYS_LOAD_HISTORY- berisi rincian perintah COPY.
- SYS_LOAD_ERROR_DETAIL- berisi rincian kesalahan perintah COPY.
- SVV_COPY_JOB_INTEGRASI— berisi rincian integrasi acara S3.
- STL_LOAD_ERROR- berisi kesalahan dari perintah COPY.
- STL_LOAD_COMMIT- berisi informasi yang digunakan untuk memecahkan masalah pemuatan data perintah COPY.

Untuk informasi tentang pemecahan masalah kesalahan integrasi peristiwa S3, lihatMemecahkan masalah integrasi peristiwa S3 dan kesalahan COPY JOB.

Untuk mendapatkan daftar file yang dimuat oleh COPY JOB, jalankan SQL berikut, tetapi pertama-tama ganti*<job_id>*:

```
SELECT job_id, job_name, data_source, copy_query, filename, status, curtime
FROM sys_copy_job copyjob
JOIN stl_load_commits loadcommit
ON copyjob.job_id = loadcommit.copy_job_id
WHERE job_id = <job_id>;
```

Pertimbangan saat membuat integrasi acara S3 untuk penyalinan otomatis

Pertimbangkan hal berikut saat menggunakan salin otomatis.

- Anda dapat membuat maksimal 200 COPY JOBS untuk setiap cluster atau workgroup dalam sebuah Akun AWS.
- Anda dapat membuat maksimal 50 integrasi acara S3 untuk setiap target Amazon Redshift.
- Anda tidak dapat membuat integrasi peristiwa S3 dengan bucket Amazon S3 sumber yang memiliki titik (.) dalam nama bucket.
- Anda hanya dapat membuat satu integrasi peristiwa S3 antara sumber dan target yang sama. Artinya, hanya ada satu integrasi peristiwa S3 antara bucket Amazon S3 dan gudang data Amazon Redshift pada satu waktu.
- Anda tidak dapat memiliki notifikasi peristiwa yang ada untuk jenis peristiwa `S3_OBJECT_CREATED` yang ditentukan pada bucket Amazon S3 sumber. Namun, setelah integrasi peristiwa S3 dibuat, Anda dapat memperbarui pemberitahuan peristiwa bucket Amazon S3 dengan cakupan yang prefix/suffix lebih sempit. Dengan cara ini, Anda juga dapat mengonfigurasi `S3_OBJECT_CREATED` untuk target lain prefix/suffix ke target lain dan menghindari konflik dengan integrasi peristiwa S3. Jika Anda mengalami masalah karena penyalinan otomatis tidak berjalan seperti yang diharapkan, siapkan AWS CloudTrail log `s3:PutBucketNotificationConfiguration` tindakan pada bucket S3 Anda untuk kerangka waktu yang dimaksud saat Anda menghubungi AWS Dukungan.

Wilayah yang Didukung

Wilayah berikut tersedia untuk penyalinan otomatis.

Region	Auto-copy
Africa (Cape Town)	Available
Asia Pasifik (Hong Kong)	Available
Asia Pasifik (Taipei)	Available
Asia Pasifik (Tokyo)	Available
Asia Pasifik (Seoul)	Available
Asia Pasifik (Osaka)	Available
Asia Pasifik (Mumbai)	Available
Asia Pasifik (Hyderabad)	Available
Asia Pasifik (Singapura)	Available
Asia Pasifik (Sydney)	Available
Asia Pasifik (Jakarta)	Available
Asia Pacific (Melbourne)	Available
Asia Pasifik (Malaysia)	Available
Asia Pasifik (Selandia Baru)	Tidak tersedia
Asia Pasifik (Thailand)	Available
Kanada (Pusat)	Available
Kanada Barat (Calgary)	Available
Tiongkok (Beijing)	Available
Tiongkok (Ningxia)	Available
Eropa (Frankfurt)	Available

Region	Auto-copy
Europe (Zurich)	Available
Eropa (Stockholm)	Available
Europe (Milan)	Available
Eropa (Spanyol)	Available
Eropa (Irlandia)	Available
Eropa (London)	Available
Eropa (Paris)	Available
Israel (Tel Aviv)	Available
Timur Tengah (UAE)	Available
Timur Tengah (Bahrain)	Available
Meksiko (Tengah)	Available
Amerika Selatan (Sao Paulo)	Available
AS Timur (Virginia Utara)	Available
AS Timur (Ohio)	Available
AS Barat (California Utara)	Available
AS Barat (Oregon)	Tersedia
AWS GovCloud (US-East)	Tersedia
AWS GovCloud (US-West)	Tersedia

Pertimbangan saat memulihkan target integrasi peristiwa S3

Saat Anda memulihkan namespace Amazon Redshift Serverless dari snapshot atau titik pemulihan ke namespace tanpa server yang sama, integrasi peristiwa S3 dan COPY JOB terkait akan dipertahankan secara otomatis. Pertimbangan berikut berlaku untuk instans Mac:

- Memulihkan snapshot ke namespace yang berbeda tidak mempertahankan integrasi peristiwa S3.
- Setelah pemulihan, Amazon Redshift secara otomatis mengidentifikasi file yang diunggah ke bucket Amazon S3 sumber setelah snapshot diambil. File apa pun yang ada di Amazon S3 yang belum dimuat akan dicerna secara otomatis. Selama proses ini, penyerapan file baru dari notifikasi peristiwa S3 terus bekerja secara paralel.

  > **Note**
  >
  > Setelah pemulihan, hanya file yang ada di bucket Amazon S3 yang memenuhi syarat untuk dicerna. File yang diunggah setelah snapshot diambil dan kemudian dihapus sebelum pemulihan selesai tidak dicerna.

- Jika integrasi peristiwa S3 dibuat setelah snapshot diambil, integrasi memasuki `NEEDS_ATTENTION` status setelah pemulihan. Untuk mengatasi masalah ini, Anda dapat memulihkan dari snapshot yang lebih baru yang menyertakan integrasi, atau menghapus integrasi.
- Jika integrasi peristiwa S3 dihapus setelah snapshot diambil, integrasi tidak dipulihkan. Status COPY JOB yang sesuai berubah menjadi`PENDING`. Anda dapat membuat integrasi baru ke bucket Amazon S3 yang sama dan membuat ulang COPY JOB. File yang ditambahkan ke S3 antara waktu snapshot dan waktu COPY JOB dibuat ulang tidak dapat dipulihkan atau dicerna kembali. Untuk mengidentifikasi file-file ini, periksa waktu snapshot di konsol dan waktu penghapusan integrasi CloudTrail dan file yang tiba di bucket S3 antara waktu-waktu ini adalah file yang terlewatkan.
- Jika COPY JOB diubah setelah snapshot diambil, namespace yang dipulihkan menggunakan COPY JOB dari snapshot. Amazon Redshift mendamaikan perbedaan apa pun dan memperbarui integrasi untuk mencerminkan konfigurasi COPY JOB yang dipulihkan.
- Untuk memilih tidak mempertahankan integrasi peristiwa S3 selama pemulihan, hapus centang pada kotak Maintain Integrations pada halaman restore di, atau jika Anda menggunakan Konsol Manajemen AWS, setel **`--no-maintain-integration`** parameter saat memanggil operasi API restore-from-snapshot atau restore-from-recovery-point. AWS CLIhttps://docs.aws.amazon.com/redshift-serverless/latest/APIReference/API_RestoreFromRecoveryPoint.html Saat Anda memilih keluar, integrasi masuk ke `FAILED` status setelah pemulihan.

- Fitur ini berlaku untuk Amazon Redshift Serverless hanya ketika dipulihkan ke namespace tanpa server yang sama. Pemulihan snapshot pada cluster yang disediakan tidak mempertahankan integrasi peristiwa S3.

Memuat tabel dengan perintah DML

Amazon Redshift mendukung perintah bahasa manipulasi data standar (DML) (INSERT, UPDATE, dan DELETE) yang dapat Anda gunakan untuk memodifikasi baris dalam tabel. Anda juga dapat menggunakan perintah TRUNCATE untuk melakukan penghapusan massal cepat.

> **Note**
>
> Kami sangat menyarankan Anda untuk menggunakan MENYALIN perintah untuk memuat data dalam jumlah besar. Menggunakan pernyataan INSERT individual untuk mengisi tabel mungkin sangat lambat. Atau, jika data Anda sudah ada di tabel database Amazon Redshift lainnya, gunakan INSERT INTO... PILIH DARI atau BUAT TABEL AS untuk meningkatkan kinerja. Untuk informasi, lihat INSERT atauCREATE TABLE AS.

Jika Anda menyisipkan, memperbarui, atau menghapus sejumlah besar baris dalam tabel, relatif terhadap jumlah baris sebelum perubahan, jalankan perintah ANALYZE dan VACUUM terhadap tabel setelah selesai. Jika sejumlah perubahan kecil menumpuk dari waktu ke waktu di aplikasi Anda, Anda mungkin ingin menjadwalkan perintah ANALYZE dan VACUUM untuk dijalankan secara berkala. Untuk informasi selengkapnya, lihat Menganalisis tabel dan Meja penyedot debu.

Topik

- Memperbarui dan memasukkan data baru

Memperbarui dan memasukkan data baru

Anda dapat secara efisien menambahkan data baru ke tabel yang ada dengan menggunakan perintah MERGE. Lakukan operasi penggabungan dengan membuat tabel pementasan dan kemudian menggunakan salah satu metode yang dijelaskan di bagian ini untuk memperbarui tabel target dari tabel pementasan. Untuk informasi selengkapnya tentang perintah MERGE, lihatMERGE.

Gunakan Gabungkan contoh kumpulan data sampel untuk Amazon Redshift, yang disebut kumpulan data TICKIT. Sebagai prasyarat, Anda dapat mengatur tabel dan data TICKIT dengan mengikuti

petunjuk yang tersedia di Memulai tugas database umum. Informasi lebih rinci tentang kumpulan data sampel ditemukan di database sampel.

Metode penggabungan 1: Mengganti baris yang ada

Jika Anda mengganti semua kolom dalam tabel target, metode tercepat untuk melakukan penggabungan adalah mengganti baris yang ada. Ini memindai tabel target hanya sekali, dengan menggunakan join dalam untuk menghapus baris yang akan diperbarui. Setelah baris dihapus, mereka diganti dengan baris baru dengan operasi sisipan tunggal dari tabel pementasan.

Gunakan metode ini jika semua hal berikut benar:

- Tabel target Anda dan tabel pementasan Anda berisi kolom yang sama.
- Anda bermaksud mengganti semua data di kolom tabel target dengan semua kolom tabel pementasan.
- Anda akan menggunakan semua baris dalam tabel pementasan dalam penggabungan.

Jika salah satu kriteria ini tidak berlaku, gunakan metode Gabung 2: Menentukan daftar kolom tanpa menggunakan MERGE, dijelaskan di bagian berikut.

Jika Anda tidak akan menggunakan semua baris dalam tabel pementasan, filter pernyataan DELETE dan INSERT dengan menggunakan klausa WHERE untuk meninggalkan baris yang tidak berubah. Namun, jika sebagian besar baris dalam tabel pementasan tidak akan berpartisipasi dalam penggabungan, sebaiknya lakukan UPDATE dan INSERT dalam langkah-langkah terpisah, seperti yang dijelaskan nanti di bagian ini.

Metode penggabungan 2: Menentukan daftar kolom tanpa menggunakan MERGE

Gunakan metode ini untuk memperbarui kolom tertentu dalam tabel target alih-alih mengganti seluruh baris. Metode ini membutuhkan waktu lebih lama dari metode sebelumnya karena memerlukan langkah pembaruan tambahan dan tidak menggunakan perintah MERGE. Gunakan metode ini jika salah satu dari berikut ini benar:

- Tidak semua kolom dalam tabel target harus diperbarui.
- Sebagian besar baris dalam tabel pementasan tidak akan digunakan dalam pembaruan.

Topik

- Membuat tabel pementasan sementara
- Melakukan operasi penggabungan dengan mengganti baris yang ada
- Melakukan operasi penggabungan dengan menentukan daftar kolom tanpa menggunakan perintah MERGE
- Gabungkan contoh

Membuat tabel pementasan sementara

Tabel pementasan adalah tabel sementara yang menyimpan semua data yang akan digunakan untuk membuat perubahan pada tabel target, termasuk pembaruan dan sisipan.

Operasi penggabungan memerlukan gabungan antara tabel pementasan dan tabel target. Untuk mengkolokasikan baris penggabungan, atur kunci distribusi tabel pementasan ke kolom yang sama dengan kunci distribusi tabel target. Misalnya, jika tabel target menggunakan kolom kunci asing sebagai kunci distribusinya, gunakan kolom yang sama untuk kunci distribusi tabel pementasan. Jika Anda membuat tabel pementasan dengan menggunakan CREATE TABLE LIKE pernyataan, tabel pementasan akan mewarisi kunci distribusi dari tabel induk. Jika Anda menggunakan pernyataan CREATE TABLE AS, tabel baru tidak mewarisi kunci distribusi. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri

Jika kunci distribusi tidak sama dengan kunci utama dan kunci distribusi tidak diperbarui sebagai bagian dari operasi penggabungan, tambahkan predikat gabungan redundan pada kolom kunci distribusi untuk mengaktifkan gabungan yang dikolokasikan. Contoh:

```
where target.primarykey = stage.primarykey
and target.distkey = stage.distkey
```

Untuk memverifikasi bahwa kueri akan menggunakan gabungan yang dikolokasikan, jalankan kueri dengan EXPLAIN dan periksa DS_DIST_NONE pada semua gabungan. Untuk informasi selengkapnya, lihat Mengevaluasi rencana kueri

Melakukan operasi penggabungan dengan mengganti baris yang ada

Saat Anda menjalankan operasi penggabungan yang dirinci dalam prosedur, letakkan semua langkah kecuali untuk membuat dan melepaskan tabel pementasan sementara dalam satu transaksi. Transaksi akan kembali jika ada langkah yang gagal. Menggunakan satu transaksi juga mengurangi jumlah commit, yang menghemat waktu dan sumber daya.

Untuk melakukan operasi penggabungan dengan mengganti baris yang ada

1. Buat tabel pementasan, lalu isi dengan data yang akan digabungkan, seperti yang ditunjukkan dalam pseudocode berikut.

```
CREATE temp table stage (like target);

INSERT INTO stage
SELECT * FROM source
WHERE source.filter = 'filter_expression';
```

2. Gunakan MERGE untuk melakukan join internal dengan tabel pementasan untuk memperbarui baris dari tabel target yang cocok dengan tabel pementasan, lalu masukkan semua baris yang tersisa ke tabel target yang tidak cocok dengan tabel pementasan.

   Kami menyarankan Anda menjalankan operasi pembaruan dan menyisipkan dalam satu perintah MERGE.

```
MERGE INTO target
USING stage [optional alias] on (target.primary_key = stage.primary_key)
WHEN MATCHED THEN
UPDATE SET col_name1 = stage.col_name1 , col_name2= stage.col_name2, col_name3 =
 {expr}
WHEN NOT MATCHED THEN
INSERT (col_name1 , col_name2, col_name3) VALUES (stage.col_name1, stage.col_name2,
 {expr});
```

3. Jatuhkan meja pementasan.

```
DROP TABLE stage;
```

Melakukan operasi penggabungan dengan menentukan daftar kolom tanpa menggunakan perintah MERGE

Ketika Anda menjalankan operasi penggabungan yang dirinci dalam prosedur, letakkan semua langkah dalam satu transaksi. Transaksi akan kembali jika ada langkah yang gagal. Menggunakan satu transaksi juga mengurangi jumlah commit, yang menghemat waktu dan sumber daya.

Untuk melakukan operasi penggabungan dengan menentukan daftar kolom

1. Letakkan seluruh operasi dalam satu blok transaksi.

```
BEGIN transaction;
…
END transaction;
```

2. Buat tabel pementasan, lalu isi dengan data yang akan digabungkan, seperti yang ditunjukkan dalam pseudocode berikut.

```
create temp table stage (like target);
insert into stage
select * from source
where source.filter = 'filter_expression';
```

3. Perbarui tabel target dengan menggunakan join dalam dengan tabel pementasan.

   - Dalam klausa UPDATE, secara eksplisit daftar kolom yang akan diperbarui.
   - Lakukan gabungan dalam dengan tabel pementasan.
   - Jika kunci distribusi berbeda dari kunci utama dan kunci distribusi tidak diperbarui, tambahkan penggabungan redundan pada kunci distribusi. Untuk memverifikasi bahwa kueri akan menggunakan gabungan yang dikolokasikan, jalankan kueri dengan EXPLAIN dan periksa DS_DIST_NONE pada semua gabungan. Untuk informasi selengkapnya, lihat Mengevaluasi rencana kueri
   - Jika tabel target Anda diurutkan berdasarkan stempel waktu, tambahkan predikat untuk memanfaatkan pemindaian terbatas rentang pada tabel target. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang kueri.
   - Jika Anda tidak akan menggunakan semua baris dalam penggabungan, tambahkan klausa untuk memfilter baris yang ingin Anda ubah. Misalnya, tambahkan filter ketidaksetaraan pada satu atau lebih kolom untuk mengecualikan baris yang belum berubah.
   - Letakkan operasi pembaruan, hapus, dan sisipkan dalam satu blok transaksi sehingga jika ada masalah, semuanya akan diputar kembali.

   Contoh:

```
begin transaction;
```

```
update target
set col1 = stage.col1,
col2 = stage.col2,
col3 = 'expression'
from stage
where target.primarykey = stage.primarykey
and target.distkey = stage.distkey
and target.col3 > 'last_update_time'
and (target.col1 != stage.col1
or target.col2 != stage.col2
or target.col3 = 'filter_expression');
```

4. Hapus baris yang tidak diperlukan dari tabel pementasan dengan menggunakan join dalam dengan tabel target. Beberapa baris dalam tabel target sudah cocok dengan baris yang sesuai di tabel pementasan, dan yang lainnya diperbarui pada langkah sebelumnya. Dalam kedua kasus, mereka tidak diperlukan untuk sisipan.

```
delete from stage
using target
where stage.primarykey = target.primarykey;
```

5. Masukkan baris yang tersisa dari tabel pementasan. Gunakan daftar kolom yang sama dalam klausa VALUES yang Anda gunakan dalam pernyataan UPDATE pada langkah kedua.

```
insert into target
(select col1, col2, 'expression'
from stage);

end transaction;
```

6. Jatuhkan meja pementasan.

```
drop table stage;
```

Gabungkan contoh

Contoh berikut melakukan penggabungan untuk memperbarui tabel SALES. Contoh pertama menggunakan metode yang lebih sederhana untuk menghapus dari tabel target dan kemudian memasukkan semua baris dari tabel pementasan. Contoh kedua memerlukan pembaruan pada kolom tertentu di tabel target, sehingga termasuk langkah pembaruan tambahan.

Gunakan Gabungkan contoh kumpulan data sampel untuk Amazon Redshift, yang disebut kumpulan data TICKIT. Sebagai prasyarat, Anda dapat mengatur tabel dan data TICKIT dengan mengikuti petunjuk yang tersedia di panduan Mem ulai tugas database umum. Informasi lebih rinci tentang kumpulan data sampel ditemukan di database sampel.

Sumber data penggabungan sampel

Contoh di bagian ini menggunakan sumber data sampel yang mencakup pembaruan dan sisipan. Untuk mengatur data sampel, Anda membuat tabel bernama SALES_UPDATE dari tabel SALES dan mengisinya dengan data acak yang mewakili aktivitas penjualan baru untuk bulan Desember. Contoh berikut menggunakan tabel SALES_UPDATE sebagai tabel pementasan.

```
-- Create a sample table as a copy of the SALES table.

create table tickit.sales_update as
select * from tickit.sales;

-- Change every fifth row to have updates.

update tickit.sales_update
set qtysold = qtysold*2,
pricepaid = pricepaid*0.8,
commission = commission*1.1
where saletime > '2008-11-30'
and mod(sellerid, 5) = 0;

-- Add some new rows to have inserts.
-- This example creates a duplicate of every fourth row.

insert into tickit.sales_update
select (salesid + 172456) as salesid, listid, sellerid, buyerid, eventid, dateid,
 qtysold, pricepaid, commission, getdate() as saletime
from tickit.sales_update
where saletime > '2008-11-30'
and mod(sellerid, 4) = 0;
```

Contoh penggabungan yang menggantikan baris yang ada berdasarkan kunci yang cocok

Contoh berikut menggunakan tabel SALES_UPDATE untuk melakukan operasi penggabungan pada tabel SALES dengan data baru untuk aktivitas penjualan Desember. Penggabungan menggantikan baris dalam tabel SALES yang memiliki pembaruan, memodifikasi `pricepaid` kolom `qtysold` dan sambil meninggalkan `commission` dan `saletime` tidak berubah.

```
MERGE into tickit.sales
USING tickit.sales_update sales_update
on ( sales.salesid = sales_update.salesid
and sales.listid = sales_update.listid
and sales_update.saletime > '2008-11-30'
and (sales.qtysold != sales_update.qtysold
or sales.pricepaid != sales_update.pricepaid))
WHEN MATCHED THEN
update SET qtysold = sales_update.qtysold,
pricepaid = sales_update.pricepaid
WHEN NOT MATCHED THEN
INSERT (salesid, listid, sellerid, buyerid, eventid, dateid, qtysold , pricepaid,
 commission, saletime)
values (sales_update.salesid, sales_update.listid, sales_update.sellerid,
 sales_update.buyerid, sales_update.eventid,
sales_update.dateid, sales_update.qtysold , sales_update.pricepaid,
 sales_update.commission, sales_update.saletime);

-- Drop the staging table.
drop table tickit.sales_update;

-- Test to see that commission and salestime were not impacted.
SELECT sales.salesid, sales.commission, sales.salestime, sales_update.commission,
 sales_update.salestime
FROM tickit.sales
INNER JOIN tickit.sales_update sales_update
ON
sales.salesid = sales_update.salesid
AND sales.listid = sales_update.listid
AND sales_update.saletime > '2008-11-30'
AND (sales.commission != sales_update.commission
OR sales.salestime != sales_update.salestime);
```

Contoh penggabungan yang menentukan daftar kolom tanpa menggunakan MERGE

Contoh berikut melakukan operasi penggabungan pada tabel SALES dengan data baru untuk aktivitas Desember, menggunakan tabel SALES_UPDATE sebagai sumber data. Data sampel termasuk pembaruan, sisipan, dan baris yang tidak berubah. Penggabungan memperbarui `pricepaid` kolom `qtysold` dan tetapi tetap `commission` dan `saletime` tidak berubah.

```
-- Create a staging table and populate it with rows from SALES_UPDATE for Dec
create temp table stagesales as select * from sales_update
```

```
where saletime > '2008-11-30';

-- Start a new transaction
begin transaction;

-- Update the target table using an inner join with the staging table
-- The join includes a redundant predicate to collocate on the distribution key -- A
 filter on saletime enables a range-restricted scan on SALES

update sales
set qtysold = stagesales.qtysold,
pricepaid = stagesales.pricepaid
from stagesales
where sales.salesid = stagesales.salesid
and sales.listid = stagesales.listid
and stagesales.saletime > '2008-11-30'
and (sales.qtysold != stagesales.qtysold
or sales.pricepaid != stagesales.pricepaid);

-- Delete matching rows from the staging table
-- using an inner join with the target table

delete from stagesales
using sales
where sales.salesid = stagesales.salesid
and sales.listid = stagesales.listid;

-- Insert the remaining rows from the staging table into the target table
insert into sales
select * from stagesales;

-- End transaction and commit
end transaction;

-- Drop the staging table
drop table stagesales;
```

Melakukan salinan mendalam

Salinan mendalam membuat ulang dan mengisi ulang tabel dengan menggunakan sisipan massal, yang secara otomatis mengurutkan tabel. Jika tabel memiliki Wilayah besar yang tidak disortir, salinan dalam jauh lebih cepat daripada ruang hampa. Kami menyarankan Anda hanya membuat

pembaruan bersamaan selama operasi penyalinan mendalam jika Anda dapat melacaknya. Setelah proses selesai, pindahkan pembaruan delta ke tabel baru. Operasi VACUUM mendukung pembaruan bersamaan secara otomatis.

Anda dapat memilih salah satu metode berikut untuk membuat salinan tabel asli:

- Gunakan tabel asli DDL.

  Jika CREATE TABLE DDL tersedia, ini adalah metode tercepat dan disukai. Jika Anda membuat tabel baru, Anda dapat menentukan semua atribut tabel dan kolom, termasuk kunci primer dan kunci asing. Anda dapat menemukan DDL asli dengan menggunakan fungsi SHOW TABLE.
- Gunakan BUAT TABEL SEPERTI.

  Jika DDL asli tidak tersedia, Anda dapat menggunakan CREATE TABLE LIKE untuk membuat ulang tabel asli. Tabel baru mewarisi penyandian, kunci distribusi, kunci pengurutan, dan atribut not-null dari tabel induk. Tabel baru tidak mewarisi kunci utama dan atribut kunci asing dari tabel induk, tetapi Anda dapat menambahkannya menggunakanALTER TABLE.
- Buat tabel sementara dan potong tabel asli.

  Jika Anda harus mempertahankan kunci utama dan atribut kunci asing dari tabel induk. Jika tabel induk memiliki dependensi, Anda dapat menggunakan CREATE TABLE... AS (CTAS) untuk membuat tabel sementara. Kemudian potong tabel asli dan isi dari tabel sementara.

  Menggunakan tabel sementara meningkatkan kinerja secara signifikan dibandingkan dengan menggunakan tabel permanen, tetapi ada risiko kehilangan data. Tabel sementara secara otomatis dijatuhkan pada akhir sesi di mana ia dibuat. TRUNCATE segera melakukan commit, bahkan jika itu berada di dalam blok transaksi. Jika TRUNCATE berhasil tetapi sesi dimatikan sebelum INSERT berikut selesai, data hilang. Jika kehilangan data tidak dapat diterima, gunakan tabel permanen.

Setelah Anda membuat salinan tabel, Anda mungkin harus memberikan akses ke tabel baru. Anda dapat menggunakan HIBAH untuk menentukan hak akses. Untuk melihat dan memberikan semua hak akses tabel, Anda harus menjadi salah satu dari berikut ini:

- Seorang pengguna super.
- Pemilik tabel yang ingin Anda salin.
- Pengguna dengan hak istimewa ACCESS SYSTEM TABLE untuk melihat hak istimewa tabel, dan dengan hak istimewa pemberian untuk semua izin yang relevan.

Selain itu, Anda mungkin harus memberikan izin penggunaan untuk skema di dalamnya salinan mendalam Anda. Pemberian izin penggunaan diperlukan jika skema deep copy Anda berbeda dari skema tabel asli, dan juga bukan `public` skema. Untuk melihat dan memberikan hak penggunaan, Anda harus menjadi salah satu dari berikut ini:

- Seorang pengguna super.
- Pengguna yang dapat memberikan izin USAGE untuk skema deep copy.

Untuk melakukan salinan mendalam menggunakan tabel asli DDL

1. (Opsional) Buat ulang tabel DDL dengan menjalankan skrip yang disebut`v_generate_tbl_ddl`.
2. Buat salinan tabel menggunakan CREATE TABLE DDL asli.
3. Gunakan pernyataan INSERT INTO... SELECT untuk mengisi salinan dengan data dari tabel asli.
4. Periksa izin yang diberikan pada tabel lama. Anda dapat melihat izin ini di tampilan sistem SVV_RELATION_PRIVILEGES.
5. Jika perlu, berikan izin tabel lama ke tabel baru.
6. Berikan izin penggunaan untuk setiap grup dan pengguna yang memiliki hak istimewa di tabel asli. Langkah ini tidak diperlukan jika tabel salinan mendalam Anda ada dalam `public` skema, atau dalam skema yang sama dengan tabel asli.
7. Jatuhkan tabel asli.
8. Gunakan pernyataan ALTER TABLE untuk mengganti nama salinan ke nama tabel asli.

Contoh berikut melakukan salinan mendalam pada tabel SAMPLE menggunakan duplikat SAMPLE bernama sample_copy.

```
--Create a copy of the original table in the sample_namespace namespace using the
 original CREATE TABLE DDL.
create table sample_namespace.sample_copy ( … );

--Populate the copy with data from the original table in the public namespace.
insert into sample_namespace.sample_copy (select * from public.sample);

--Check SVV_RELATION_PRIVILEGES for the original table's privileges.
select * from svv_relation_privileges where namespace_name = 'public' and relation_name
 = 'sample' order by identity_type, identity_id, privilege_type;
```

```
--Grant the original table's privileges to the copy table.
grant DELETE on table sample_namespace.sample_copy to group group1;
grant INSERT, UPDATE on table sample_namespace.sample_copy to group group2;
grant SELECT on table sample_namespace.sample_copy to user1;
grant INSERT, SELECT, UPDATE on table sample_namespace.sample_copy to user2;

--Grant usage permission to every group and user that has privileges in the original
 table.
grant USAGE on schema sample_namespace to group group1, group group2, user1, user2;

--Drop the original table.
drop table public.sample;

--Rename the copy table to match the original table's name.
alter table sample_namespace.sample_copy rename to sample;
```

Untuk melakukan deep copy menggunakan CREATE TABLE LIKE

1. Buat tabel baru menggunakan CREATE TABLE LIKE.
2. Gunakan pernyataan INSERT INTO... SELECT untuk menyalin baris dari tabel saat ini ke tabel baru.
3. Periksa izin yang diberikan pada tabel lama. Anda dapat melihat izin ini di tampilan sistem SVV_RELATION_PRIVILEGES.
4. Jika perlu, berikan izin tabel lama ke tabel baru.
5. Berikan izin penggunaan untuk setiap grup dan pengguna yang memiliki hak istimewa di tabel asli. Langkah ini tidak diperlukan jika tabel salinan mendalam Anda ada dalam `public` skema, atau dalam skema yang sama dengan tabel asli.
6. Jatuhkan tabel saat ini.
7. Gunakan pernyataan ALTER TABLE untuk mengganti nama tabel baru ke nama tabel asli.

Contoh berikut melakukan salinan mendalam pada tabel SAMPLE menggunakan CREATE TABLE LIKE.

```
--Create a copy of the original table in the sample_namespace namespace using CREATE
 TABLE LIKE.
create table sameple_namespace.sample_copy (like public.sample);

--Populate the copy with data from the original table.
```

```
insert into sample_namespace.sample_copy (select * from public.sample);

--Check SVV_RELATION_PRIVILEGES for the original table's privileges.
select * from svv_relation_privileges where namespace_name = 'public' and relation_name
 = 'sample' order by identity_type, identity_id, privilege_type;

--Grant the original table's privileges to the copy table.
grant DELETE on table sample_namespace.sample_copy to group group1;
grant INSERT, UPDATE on table sample_namespace.sample_copy to group group2;
grant SELECT on table sample_namespace.sample_copy to user1;
grant INSERT, SELECT, UPDATE on table sample_namespace.sample_copy to user2;

--Grant usage permission to every group and user that has privileges in the original
 table.
grant USAGE on schema sample_namespace to group group1, group group2, user1, user2;

--Drop the original table.
drop table public.sample;

--Rename the copy table to match the original table's name.
alter table sample_namespace.sample_copy rename to sample;
```

Untuk melakukan salinan mendalam dengan membuat tabel sementara dan memotong tabel asli

1. Gunakan CREATE TABLE AS untuk membuat tabel sementara dengan baris dari tabel asli.
2. Memotong tabel saat ini.
3. Gunakan pernyataan INSERT INTO... SELECT untuk menyalin baris dari tabel sementara ke tabel asli.
4. Jatuhkan meja sementara.

Contoh berikut melakukan salinan mendalam pada tabel SALES dengan membuat tabel sementara dan memotong tabel asli. Karena tabel asli tetap ada, Anda tidak perlu memberikan izin ke tabel salin.

```
--Create a temp table copy using CREATE TABLE AS.
create temp table salestemp as select * from sales;

--Truncate the original table.
truncate sales;

--Copy the rows from the temporary table to the original table.
```

```
insert into sales (select * from salestemp);

--Drop the temporary table.
drop table salestemp;
```

Menganalisis tabel

Operasi ANALYZE memperbarui metadata statistik yang digunakan perencana kueri untuk memilih rencana yang optimal.

Dalam kebanyakan kasus, Anda tidak perlu menjalankan perintah ANALYZE secara eksplisit. Amazon Redshift memantau perubahan pada beban kerja Anda dan secara otomatis memperbarui statistik di latar belakang. Selain itu, perintah COPY melakukan analisis secara otomatis ketika memuat data ke tabel kosong.

Untuk secara eksplisit menganalisis tabel atau seluruh database, jalankan MENGANALISA perintah.

Analisis otomatis

Amazon Redshift terus memantau database Anda dan secara otomatis melakukan operasi analisis di latar belakang. Untuk meminimalkan dampak terhadap kinerja sistem Anda, analisis otomatis berjalan selama periode ketika beban kerja ringan.

Analisis otomatis diaktifkan secara default. Untuk mematikan analisis otomatis, atur `auto_analyze` parameter **`false`** dengan memodifikasi grup parameter cluster Anda.

Untuk mengurangi waktu pemrosesan dan meningkatkan kinerja sistem secara keseluruhan, Amazon Redshift melewatkan analisis otomatis untuk setiap tabel di mana tingkat modifikasi kecil.

Operasi analisis melewatkan tabel yang memiliki statistik terkini. Jika Anda menjalankan ANALYZE sebagai bagian dari alur kerja ekstrak, transformasi, dan muat (ETL), analisis otomatis akan melewatkan tabel yang memiliki statistik saat ini. Demikian pula, ANALISIS eksplisit melewatkan tabel ketika analisis otomatis telah memperbarui statistik tabel.

Analisis data tabel baru

Secara default, perintah COPY melakukan ANALYZE setelah memuat data ke tabel kosong. Anda dapat memaksa ANALYZE terlepas dari apakah tabel kosong dengan menyetel STATUPDATE ON. Jika Anda menentukan STATUPDATE OFF, ANALISIS tidak dilakukan. Hanya pemilik tabel atau

superuser yang dapat menjalankan perintah ANALYZE atau menjalankan perintah COPY dengan STATUPDATE disetel ke ON.

Amazon Redshift juga menganalisis tabel baru yang Anda buat dengan perintah berikut:

- BUAT TABEL SEBAGAI (CTAS)
- BUAT TABEL TEMP SEBAGAI
- PILIH MENJADI

Amazon Redshift mengembalikan pesan peringatan saat Anda menjalankan kueri terhadap tabel baru yang tidak dianalisis setelah data awalnya dimuat. Tidak ada peringatan yang terjadi saat Anda menanyakan tabel setelah pembaruan atau pemuatan berikutnya. Pesan peringatan yang sama dikembalikan ketika Anda menjalankan perintah EXPLAIN pada kueri yang mereferensikan tabel yang belum dianalisis.

Setiap kali menambahkan data ke tabel yang tidak kosong secara signifikan mengubah ukuran tabel, Anda dapat memperbarui statistik secara eksplisit. Anda melakukannya dengan menjalankan perintah ANALYZE atau dengan menggunakan opsi STATUPDATE ON dengan perintah COPY. Untuk melihat detail tentang jumlah baris yang telah disisipkan atau dihapus sejak ANALYZE terakhir, kueri tabel katalog PG_STATISTIC_INDICATOR sistem.

Anda dapat menentukan ruang lingkup MENGANALISA perintah ke salah satu dari berikut ini:

- Seluruh database saat ini
- Satu tabel
- Satu atau lebih kolom spesifik dalam satu tabel
- Kolom yang kemungkinan akan digunakan sebagai predikat dalam kueri

Perintah ANALYZE mendapatkan sampel baris dari tabel, melakukan beberapa perhitungan, dan menyimpan statistik kolom yang dihasilkan. Secara default, Amazon Redshift menjalankan sampel pass untuk kolom DISTKEY dan pass sampel lain untuk semua kolom lain dalam tabel. Jika Anda ingin menghasilkan statistik untuk subset kolom, Anda dapat menentukan daftar kolom yang dipisahkan koma. Anda dapat menjalankan ANALYZE dengan klausa PREDICATE COLUMNS untuk melewati kolom yang tidak digunakan sebagai predikat.

Operasi ANALYZE sangat padat sumber daya, jadi jalankan hanya pada tabel dan kolom yang benar-benar memerlukan pembaruan statistik. Anda tidak perlu menganalisis semua kolom di semua tabel

secara teratur atau pada jadwal yang sama. Jika data berubah secara substan#ial, analisis kolom yang sering digunakan sebagai berikut:

- Operasi penyortiran dan pengelompokan
- Gabungan
- Predikat kueri

Untuk mengurangi waktu pemrosesan dan meningkatkan kinerja sistem secara keseluruhan, Amazon Redshift melewatkan ANALYZE untuk setiap tabel yang memiliki persentase baris yang diubah rendah, sebagaimana ditentukan oleh parameter. analisis_ambang_persen Secara default, ambang analisis diatur ke 10 persen. Anda dapat mengubah ambang analisis untuk sesi saat ini dengan menjalankan SET perintah.

Kolom yang cenderung tidak memerlukan analisis yang sering adalah kolom yang mewakili fakta dan ukuran dan atribut terkait yang tidak pernah benar-benar ditanyakan, seperti kolom VARCHAR besar. Misalnya, pertimbangkan tabel LISTING di database TICKIT.

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'listing';


column         |        type        | encoding | distkey | sortkey
---------------+--------------------+----------+---------+---------
listid         | integer            | none     | t       | 1
sellerid       | integer            | none     | f       | 0
eventid        | integer            | mostly16 | f       | 0
dateid         | smallint           | none     | f       | 0
numtickets     | smallint           | mostly8  | f       | 0
priceperticket | numeric(8,2)       | bytedict | f       | 0
totalprice     | numeric(8,2)       | mostly32 | f       | 0
listtime       | timestamp with...  | none     | f       | 0
```

Jika tabel ini dimuat setiap hari dengan sejumlah besar catatan baru, kolom LISTID, yang sering digunakan dalam kueri sebagai kunci gabungan, harus dianalisis secara teratur. Jika TOTALPRICE dan LISTTIME adalah batasan yang sering digunakan dalam kueri, Anda dapat menganalisis kolom tersebut dan kunci distribusi pada setiap hari kerja.

```
analyze listing(listid, totalprice, listtime);
```

Misalkan penjual dan peristiwa dalam aplikasi jauh lebih statis, dan ID tanggal mengacu pada serangkaian hari tetap yang hanya mencakup dua atau tiga tahun. Dalam hal ini, nilai unik untuk kolom ini tidak berubah secara signifikan. Namun, jumlah contoh dari setiap nilai unik akan terus meningkat.

Selain itu, pertimbangkan kasus di mana ukuran NUMTICKETS dan PRICEPERTICKET jarang ditanyakan dibandingkan dengan kolom TOTALPRICE. Dalam hal ini, Anda dapat menjalankan perintah ANALYZE di seluruh tabel sekali setiap akhir pekan untuk memperbarui statistik untuk lima kolom yang tidak dianalisis setiap hari:

Kolom predikat

Sebagai alternatif yang nyaman untuk menentukan daftar kolom, Anda dapat memilih untuk menganalisis hanya kolom yang kemungkinan akan digunakan sebagai predikat. Saat Anda menjalankan kueri, kolom apa pun yang digunakan dalam gabungan, kondisi filter, atau grup demi klausa ditandai sebagai kolom predikat dalam katalog sistem. Ketika Anda menjalankan ANALYZE dengan klausa PREDICATE COLUMNS, operasi analisis hanya mencakup kolom yang memenuhi kriteria berikut:

- Kolom ditandai sebagai kolom predikat.
- Kolom adalah kunci distribusi.
- Kolom adalah bagian dari kunci pengurutan.

Jika tidak ada kolom tabel yang ditandai sebagai predikat, ANALYZE menyertakan semua kolom, bahkan ketika KOLOM PREDIKAT ditentukan. Jika tidak ada kolom yang ditandai sebagai kolom predikat, mungkin karena tabel belum ditanyakan.

Anda dapat memilih untuk menggunakan KOLOM PREDIKAT ketika pola kueri beban kerja Anda relatif stabil. Ketika pola kueri bervariasi, dengan kolom yang berbeda sering digunakan sebagai predikat, menggunakan KOLOM PREDIKAT dapat sementara menghasilkan statistik basi. Statistik basi dapat menyebabkan rencana runtime kueri yang kurang optimal dan runtime yang lama. Namun, lain kali Anda menjalankan ANALYZE menggunakan KOLOM PREDIKAT, kolom predikat baru disertakan.

Untuk melihat detail kolom predikat, gunakan SQL berikut untuk membuat tampilan bernama PREDICATE_COLUMNS.

```
CREATE VIEW predicate_columns AS
```

```
WITH predicate_column_info as (
SELECT ns.nspname AS schema_name, c.relname AS table_name, a.attnum as col_num,
 a.attname as col_name,
        CASE
            WHEN 10002 = s.stakind1 THEN array_to_string(stavalues1, '||')
            WHEN 10002 = s.stakind2 THEN array_to_string(stavalues2, '||')
            WHEN 10002 = s.stakind3 THEN array_to_string(stavalues3, '||')
            WHEN 10002 = s.stakind4 THEN array_to_string(stavalues4, '||')
            ELSE NULL::varchar
        END AS pred_ts
   FROM pg_statistic s
   JOIN pg_class c ON c.oid = s.starelid
   JOIN pg_namespace ns ON c.relnamespace = ns.oid
   JOIN pg_attribute a ON c.oid = a.attrelid AND a.attnum = s.staattnum)
SELECT schema_name, table_name, col_num, col_name,
       pred_ts NOT LIKE '2000-01-01%' AS is_predicate,
       CASE WHEN pred_ts NOT LIKE '2000-01-01%' THEN (split_part(pred_ts,
 '||',1))::timestamp ELSE NULL::timestamp END as first_predicate_use,
       CASE WHEN pred_ts NOT LIKE '%||2000-01-01%' THEN (split_part(pred_ts,
 '||',2))::timestamp ELSE NULL::timestamp END as last_analyze
FROM predicate_column_info;
```

Misalkan Anda menjalankan query berikut terhadap tabel LISTING. Perhatikan bahwa LISTID, LISTTIME, dan EVENTID digunakan dalam klausa join, filter, dan group by.

```
select s.buyerid,l.eventid, sum(l.totalprice)
from listing l
join sales s on l.listid = s.listid
where l.listtime > '2008-12-01'
group by l.eventid, s.buyerid;
```

Ketika Anda menanyakan tampilan PREDICATE_COLUMNS, seperti yang ditunjukkan pada contoh berikut, Anda melihat bahwa LISTID, EVENTID, dan LISTTIME ditandai sebagai kolom predikat.

```
select * from predicate_columns
where table_name = 'listing';
```

```
schema_name | table_name | col_num | col_name       | is_predicate |
 first_predicate_use | last_analyze
------------+------------+---------+----------------+--------------
+---------------------+---------------------
```

```
public      | listing    |       1 | listid         | true         | 2017-05-05
 19:27:59 | 2017-05-03 18:27:41
public      | listing    |       2 | sellerid       | false        |
  | 2017-05-03 18:27:41
public      | listing    |       3 | eventid        | true         | 2017-05-16
 20:54:32 | 2017-05-03 18:27:41
public      | listing    |       4 | dateid         | false        |
  | 2017-05-03 18:27:41
public      | listing    |       5 | numtickets     | false        |
  | 2017-05-03 18:27:41
public      | listing    |       6 | priceperticket | false        |
  | 2017-05-03 18:27:41
public      | listing    |       7 | totalprice     | false        |
  | 2017-05-03 18:27:41
public      | listing    |       8 | listtime       | true         | 2017-05-16
 20:54:32 | 2017-05-03 18:27:41
```

Menjaga statistik terkini meningkatkan kinerja kueri dengan memungkinkan perencana kueri untuk memilih rencana yang optimal. Amazon Redshift menyegarkan statistik secara otomatis di latar belakang, dan Anda juga dapat menjalankan perintah ANALYZE secara eksplisit. Jika Anda memilih untuk menjalankan ANALYZE secara eksplisit, lakukan hal berikut:

- Jalankan perintah ANALYZE sebelum menjalankan query.
- Jalankan perintah ANALYZE pada database secara rutin di akhir setiap siklus pemuatan atau pembaruan reguler.
- Jalankan perintah ANALYZE pada setiap tabel baru yang Anda buat dan setiap tabel atau kolom yang ada yang mengalami perubahan signifikan.
- Pertimbangkan untuk menjalankan operasi ANALYZE pada jadwal yang berbeda untuk berbagai jenis tabel dan kolom, tergantung pada penggunaannya dalam kueri dan kecenderungannya untuk berubah.
- Untuk menghemat waktu dan sumber daya cluster, gunakan klausa PREDICATE COLUMNS saat Anda menjalankan ANALYZE.

Anda tidak perlu menjalankan perintah ANALYZE secara eksplisit setelah memulihkan snapshot ke cluster yang disediakan atau namespace tanpa server, atau setelah melanjutkan cluster yang telah dijeda. Amazon Redshift mempertahankan informasi tabel sistem dalam kasus ini, membuat perintah ANALYZE manual tidak diperlukan. Amazon Redshift akan terus menjalankan operasi analisis otomatis sesuai kebutuhan.

Operasi analisis melewatkan tabel yang memiliki statistik terkini. Jika Anda menjalankan ANALYZE sebagai bagian dari alur kerja ekstrak, transformasi, dan muat (ETL), analisis otomatis akan melewatkan tabel yang memiliki statistik saat ini. Demikian pula, ANALISIS eksplisit melewatkan tabel ketika analisis otomatis telah memperbarui statistik tabel.

Analisis riwayat perintah

Sangat berguna untuk mengetahui kapan perintah ANALYZE terakhir dijalankan pada tabel atau database. Saat perintah ANALYZE dijalankan, Amazon Redshift menjalankan beberapa kueri yang terlihat seperti ini:

```
padb_fetch_sample: select * from table_name
```

Kueri STL_ANALYZE untuk melihat riwayat operasi analisis. Jika Amazon Redshift menganalisis tabel menggunakan analisis otomatis, `is_background` kolom disetel ke `t` (true). Jika tidak, itu diatur ke `f` (false). Contoh berikut bergabung dengan STV_TBL_PERM untuk menampilkan nama tabel dan detail runtime.

```
select distinct a.xid, trim(t.name) as name, a.status, a.rows, a.modified_rows,
 a.starttime, a.endtime
from stl_analyze a
join stv_tbl_perm t  on t.id=a.table_id
where name = 'users'
order by starttime;


xid    | name  | status          | rows  | modified_rows | starttime           |
 endtime
-------+-------+-----------------+-------+---------------+---------------------
+---------------------
  1582 | users | Full            | 49990 |         49990 | 2016-09-22 22:02:23 |
 2016-09-22 22:02:28
244287 | users | Full            | 24992 |         74988 | 2016-10-04 22:50:58 |
 2016-10-04 22:51:01
244712 | users | Full            | 49984 |         24992 | 2016-10-04 22:56:07 |
 2016-10-04 22:56:07
245071 | users | Skipped         | 49984 |             0 | 2016-10-04 22:58:17 |
 2016-10-04 22:58:17
245439 | users | Skipped         | 49984 |          1982 | 2016-10-04 23:00:13 |
 2016-10-04 23:00:13
```

```
(5 rows)
```

Atau, Anda dapat menjalankan kueri yang lebih kompleks yang mengembalikan semua pernyataan yang berjalan di setiap transaksi selesai yang menyertakan perintah ANALYZE:

```
select xid, to_char(starttime, 'HH24:MM:SS.MS') as starttime,
datediff(sec,starttime,endtime ) as secs, substring(text, 1, 40)
from svl_statementtext
where sequence = 0
and xid in (select xid from svl_statementtext s where s.text like 'padb_fetch_sample
%' )
order by xid desc, starttime;

xid  |  starttime   | secs |                 substring
-----+--------------+------+-----------------------------------------
1338 | 12:04:28.511 |    4 | Analyze date
1338 | 12:04:28.511 |    1 | padb_fetch_sample: select count(*) from
1338 | 12:04:29.443 |    2 | padb_fetch_sample: select * from date
1338 | 12:04:31.456 |    1 | padb_fetch_sample: select * from date
1337 | 12:04:24.388 |    1 | padb_fetch_sample: select count(*) from
1337 | 12:04:24.388 |    4 | Analyze sales
1337 | 12:04:25.322 |    2 | padb_fetch_sample: select * from sales
1337 | 12:04:27.363 |    1 | padb_fetch_sample: select * from sales
...
```

Meja penyedot debu

Amazon Redshift dapat secara otomatis mengurutkan dan melakukan operasi VACUUM DELETE pada tabel di latar belakang. Untuk membersihkan tabel setelah pemuatan atau serangkaian pembaruan inkremental, Anda juga dapat menjalankan VAKUM perintah, baik terhadap seluruh database atau terhadap tabel individual.

> **Note**
>
> Hanya pengguna dengan izin tabel yang diperlukan yang dapat menyedot meja secara efektif. Jika VACUUM dijalankan tanpa izin tabel yang diperlukan, operasi selesai dengan sukses tetapi tidak berpengaruh. Untuk daftar izin tabel yang valid untuk menjalankan VACUUM secara efektif, lihatVAKUM.

Untuk alasan ini, kami sarankan menyedot debu masing-masing meja sesuai kebutuhan. Kami juga merekomendasikan pendekatan ini karena menyedot debu seluruh database berpotensi merupakan operasi yang mahal.

Pengurutan tabel otomatis

Amazon Redshift secara otomatis mengurutkan data di latar belakang untuk mempertahankan data tabel dalam urutan kunci pengurutannya. Amazon Redshift melacak kueri pemindaian Anda untuk menentukan bagian tabel mana yang akan mendapat manfaat dari penyortiran. Amazon Redshift juga melacak kueri pemindaian dari cluster penskalaan konkurensi. Untuk arsitektur multi cluster yang menggunakan Amazon Redshift Data Sharing, Amazon Redshift juga melacak kueri pemindaian yang berasal dari konsumen clusters/workgroups di jaring data Anda, termasuk di clusters/workgroups seluruh Wilayah yang berbeda. Statistik pemindaian dari cluster utama, cluster penskalaan konkurensi, dan cluster konsumen dikumpulkan untuk menentukan bagian tabel mana yang akan mendapat manfaat dari penyortiran.

Bergantung pada beban pada sistem, Amazon Redshift secara otomatis memulai pengurutan. Urutan otomatis ini mengurangi kebutuhan untuk menjalankan perintah VACUUM untuk menjaga data dalam urutan kunci pengurutan. Jika Anda membutuhkan data yang sepenuhnya diurutkan dalam urutan kunci, misalnya setelah pemuatan data yang besar, maka Anda masih dapat menjalankan perintah VACUUM secara manual. Untuk menentukan apakah tabel Anda akan mendapat manfaat dengan menjalankan VACUUM SORT, pantau `vacuum_sort_benefit` kolom diSVV_TABLE_INFO.

Amazon Redshift melacak kueri pemindaian yang menggunakan tombol pengurutan pada setiap tabel. Amazon Redshift memperkirakan persentase maksimum peningkatan dalam pemindaian dan pemfilteran data untuk setiap tabel (jika tabel diurutkan sepenuhnya). Perkiraan ini terlihat di `vacuum_sort_benefit` kolom diSVV_TABLE_INFO. Anda dapat menggunakan kolom ini, bersama dengan `unsorted` kolom, untuk menentukan kapan kueri dapat memperoleh manfaat dari menjalankan VACUUM SORT secara manual pada tabel. `unsorted`Kolom mencerminkan urutan fisik tabel. `vacuum_sort_benefit`Kolom menentukan dampak menyortir tabel dengan menjalankan VACUUM SORT secara manual.

Misalnya, pertimbangkan kueri berikut:

```
select "table", unsorted,vacuum_sort_benefit from svv_table_info order by 1;
```

```
 table | unsorted | vacuum_sort_benefit
-------+----------+--------------------
 sales |    85.71 |                5.00
 event |    45.24 |               67.00
```

Untuk tabel “penjualan”, meskipun tabel ~ 86% secara fisik tidak disortir, dampak kinerja kueri dari tabel yang 86% tidak disortir hanya 5%. Ini mungkin karena hanya sebagian kecil dari tabel yang diakses oleh kueri, atau sangat sedikit kueri yang mengakses tabel. Untuk tabel “acara”, tabel ~ 45% secara fisik tidak disortir. Tetapi dampak kinerja kueri 67% menunjukkan bahwa sebagian besar tabel diakses oleh kueri, atau jumlah kueri yang mengakses tabel besar. Tabel “acara” berpotensi mendapat manfaat dari menjalankan VACUUM SORT.

Hapus vakum otomatis

Saat Anda melakukan penghapusan, baris ditandai untuk dihapus, tetapi tidak dihapus. Amazon Redshift secara otomatis menjalankan operasi VACUUM DELETE di latar belakang berdasarkan jumlah baris yang dihapus dalam tabel database. Amazon Redshift menjadwalkan VACUUM DELETE untuk berjalan selama periode beban berkurang dan menjeda operasi selama periode beban tinggi.

Topik

- Frekuensi vakum
- Urutkan tahap dan gabungkan tahap
- Ambang vakum
- Jenis vakum
- Meminimalkan waktu vakum

Frekuensi vakum

Anda harus menyedot debu sesering yang diperlukan untuk mempertahankan kinerja kueri yang konsisten. Pertimbangkan faktor-faktor ini saat menentukan seberapa sering menjalankan perintah VACUUM Anda:

- Jalankan VACUUM selama periode waktu ketika Anda mengharapkan aktivitas minimal di cluster, seperti malam hari atau selama jendela administrasi database yang ditunjuk.
- Jalankan perintah VACUUM di luar jendela pemeliharaan. Untuk informasi selengkapnya, lihat Men jadwalkan sekitar jendela pemeliharaan.

- Wilayah besar yang tidak disortir menghasilkan waktu vakum yang lebih lama. Jika Anda menunda penyedot debu, vakum akan memakan waktu lebih lama karena lebih banyak data harus diatur ulang.
- VACUUM adalah operasi I/O intensif, jadi semakin lama waktu yang dibutuhkan untuk penyelesaian vakum Anda, semakin besar dampaknya pada kueri bersamaan dan operasi database lain yang berjalan di cluster Anda.
- VACUUM membutuhkan waktu lebih lama untuk tabel yang menggunakan penyortiran interleaved. Untuk mengevaluasi apakah tabel yang disisipkan harus disortir ulang, kueri tampilan. SVV_KOLUMN_INTERLEAVED_

Urutkan tahap dan gabungkan tahap

Amazon Redshift melakukan operasi vakum dalam dua tahap: pertama, mengurutkan baris di wilayah yang tidak disortir, kemudian, jika perlu, menggabungkan baris yang baru diurutkan di akhir tabel dengan baris yang ada. Saat menyedot debu meja besar, operasi vakum berlangsung dalam serangkaian langkah yang terdiri dari jenis tambahan diikuti dengan penggabungan. Jika operasi gagal atau jika Amazon Redshift offline selama vakum, tabel atau database yang disedot sebagian akan berada dalam keadaan konsisten, tetapi Anda harus memulai ulang operasi vakum secara manual. Urutan inkremental hilang, tetapi baris gabungan yang dilakukan sebelum kegagalan tidak perlu disedot lagi. Jika wilayah yang tidak disortir besar, waktu yang hilang mungkin signifikan. Untuk informasi selengkapnya tentang tahapan pengurutan dan penggabungan, lihatKurangi volume baris gabungan.

Pengguna dapat mengakses tabel saat sedang disedot debu. Anda dapat melakukan kueri dan operasi penulisan saat tabel sedang disedot, tetapi ketika DML dan vakum berjalan secara bersamaan, keduanya mungkin membutuhkan waktu lebih lama. Jika Anda menjalankan pernyataan UPDATE dan DELETE selama vakum, kinerja sistem mungkin berkurang. Penggabungan inkremental memblokir sementara operasi UPDATE dan DELETE bersamaan, dan operasi UPDATE dan DELETE pada gilirannya memblokir sementara langkah-langkah penggabungan tambahan pada tabel yang terpengaruh. Operasi DDL, seperti ALTER TABLE, diblokir sampai operasi vakum selesai dengan tabel.

> **Note**
>
> Berbagai pengubah untuk VACUUM mengontrol cara kerjanya. Anda dapat menggunakannya untuk menyesuaikan operasi vakum untuk kebutuhan saat ini. Misalnya,

menggunakan VACUUM RECLUSTER memperpendek operasi vakum dengan tidak melakukan operasi penggabungan penuh. Untuk informasi selengkapnya, lihat VAKUM.

Ambang vakum

Secara default, VACUUM melewatkan fase pengurutan untuk setiap tabel di mana lebih dari 95 persen baris tabel sudah diurutkan. Melewatkan fase pengurutan dapat secara signifikan meningkatkan kinerja VACUUM. Untuk mengubah ambang pengurutan default untuk satu tabel, sertakan nama tabel dan parameter TO ambang batas PERSEN saat Anda menjalankan perintah VACUUM.

Jenis vakum

Untuk informasi tentang berbagai jenis vakum, lihatVAKUM.

Meminimalkan waktu vakum

Amazon Redshift secara otomatis mengurutkan data dan menjalankan VACUUM DELETE di latar belakang. Ini mengurangi kebutuhan untuk menjalankan perintah VACUUM. Menyedot debu berpotensi memakan waktu. Tergantung pada sifat data Anda, kami merekomendasikan praktik berikut untuk meminimalkan waktu vakum.

Topik

- Putuskan apakah akan mengindeks ulang
- Kurangi ukuran wilayah yang tidak disortir
- Kurangi volume baris gabungan
- Muat data Anda dalam urutan kunci
- Gunakan tabel deret waktu untuk mengurangi data yang disimpan

Putuskan apakah akan mengindeks ulang

Anda sering dapat meningkatkan kinerja kueri secara signifikan dengan menggunakan gaya pengurutan yang disisipkan, tetapi seiring waktu kinerja dapat menurun jika distribusi nilai dalam kolom kunci pengurutan berubah.

Saat pertama kali memuat tabel selisipan kosong menggunakan COPY atau CREATE TABLE AS, Amazon Redshift secara otomatis membuat indeks yang disisipkan. Jika Anda awalnya memuat tabel

yang disisipkan menggunakan INSERT, Anda perlu menjalankan VACUUM REINDEX setelahnya untuk menginisialisasi indeks yang disisipkan.

Seiring waktu, saat Anda menambahkan baris dengan nilai kunci pengurutan baru, kinerja mungkin menurun jika distribusi nilai dalam kolom kunci pengurutan berubah. Jika baris baru Anda terutama berada dalam kisaran nilai kunci pengurutan yang ada, Anda tidak perlu mengindeks ulang. Jalankan VACUUM SORT ONLY atau VACUUM FULL untuk mengembalikan urutan pengurutan.

Mesin kueri dapat menggunakan urutan pengurutan untuk secara efisien memilih blok data mana yang perlu dipindai untuk memproses kueri. Untuk pengurutan yang disisipkan, Amazon Redshift menganalisis nilai kolom kunci pengurutan untuk menentukan urutan pengurutan yang optimal. Jika distribusi nilai kunci berubah, atau miring, saat baris ditambahkan, strategi pengurutan tidak akan lagi optimal, dan manfaat kinerja penyortiran akan menurun. Untuk menganalisis ulang distribusi kunci pengurutan, Anda dapat menjalankan VACUUM REINDEX. Operasi reindex memakan waktu, jadi untuk memutuskan apakah tabel akan mendapat manfaat dari indeks ulang, kueri tampilan. SVV_KOLUMN_INTERLEAVED_

Misalnya, kueri berikut menunjukkan detail untuk tabel yang menggunakan kunci pengurutan yang disisipkan.

```
select tbl as tbl_id, stv_tbl_perm.name as table_name,
col, interleaved_skew, last_reindex
from svv_interleaved_columns, stv_tbl_perm
where svv_interleaved_columns.tbl = stv_tbl_perm.id
and interleaved_skew is not null;


 tbl_id | table_name | col | interleaved_skew | last_reindex
--------+------------+-----+------------------+---------------------
 100048 | customer   |   0 |             3.65 | 2015-04-22 22:05:45
 100068 | lineorder  |   1 |             2.65 | 2015-04-22 22:05:45
 100072 | part       |   0 |             1.65 | 2015-04-22 22:05:45
 100077 | supplier   |   1 |             1.00 | 2015-04-22 22:05:45
(4 rows)
```

Nilai untuk `interleaved_skew` adalah rasio yang menunjukkan jumlah kemiringan. Nilai 1 berarti tidak ada kemiringan. Jika kemiringan lebih besar dari 1,4, VACUUM REINDEX biasanya akan meningkatkan kinerja kecuali kemiringan melekat pada himpunan yang mendasarinya.

Anda dapat menggunakan nilai tanggal `last_reindex` untuk menentukan berapa lama sejak indeks ulang terakhir.

Kurangi ukuran wilayah yang tidak disortir

Wilayah yang tidak disortir tumbuh ketika Anda memuat sejumlah besar data baru ke dalam tabel yang sudah berisi data atau ketika Anda tidak menyedot tabel sebagai bagian dari operasi pemeliharaan rutin Anda. Untuk menghindari operasi vakum yang berjalan lama, gunakan praktik berikut:

- Jalankan operasi vakum pada jadwal reguler.

  Jika Anda memuat tabel secara bertahap (seperti pembaruan harian yang mewakili persentase kecil dari jumlah total baris dalam tabel), menjalankan VACUUM secara teratur akan membantu memastikan bahwa operasi vakum individu berjalan dengan cepat.

- Jalankan beban terbesar terlebih dahulu.

  Jika Anda perlu memuat tabel baru dengan beberapa operasi COPY, jalankan beban terbesar terlebih dahulu. Saat Anda menjalankan beban awal ke tabel baru atau terpotong, semua data dimuat langsung ke wilayah yang diurutkan, sehingga tidak diperlukan vakum.

- Memotong tabel alih-alih menghapus semua baris.

  Menghapus baris dari tabel tidak merebut kembali ruang yang ditempati baris sampai Anda melakukan operasi vakum; Namun, memotong tabel mengosongkan tabel dan mengambil kembali ruang disk, sehingga tidak diperlukan vakum. Atau, jatuhkan tabel dan buat kembali.

- Potong atau jatuhkan tabel tes.

  Jika Anda memuat sejumlah kecil baris ke dalam tabel untuk tujuan pengujian, jangan hapus baris setelah selesai. Sebagai gantinya, potong tabel dan muat ulang baris tersebut sebagai bagian dari operasi beban produksi berikutnya.

- Lakukan salinan mendalam.

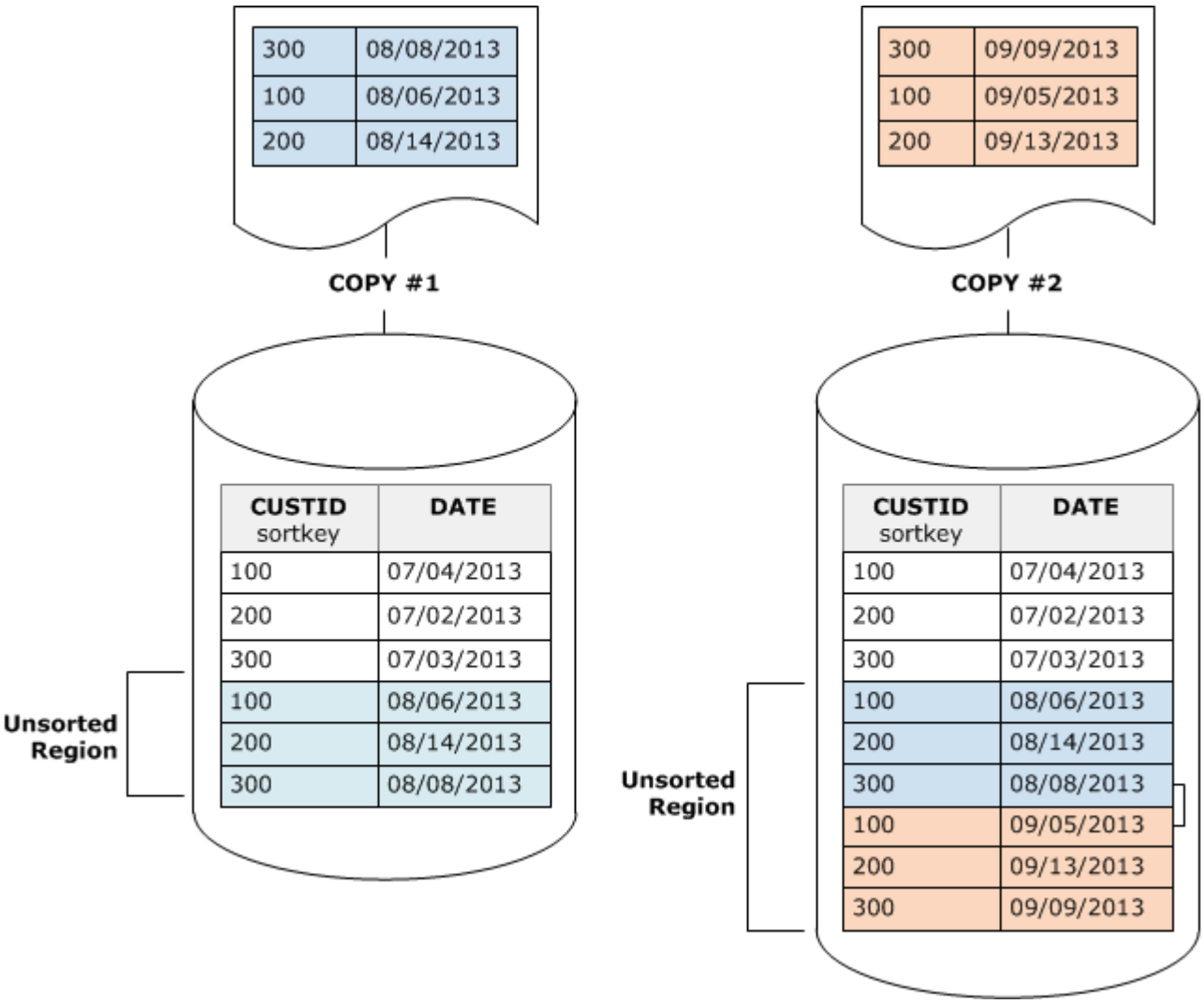
  Jika tabel yang menggunakan tabel kunci pengurutan majemuk memiliki wilayah besar yang tidak disortir, salinan mendalam jauh lebih cepat daripada ruang hampa. Salinan mendalam membuat ulang dan mengisi ulang tabel dengan menggunakan sisipan massal, yang secara otomatis mengurutkan ulang tabel. Jika tabel memiliki wilayah besar yang tidak disortir, salinan dalam jauh lebih cepat daripada ruang hampa. Tukarnya adalah Anda tidak dapat membuat pembaruan bersamaan selama operasi penyalinan mendalam, yang dapat Anda lakukan selama vakum. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang kueri.

Kurangi volume baris gabungan

Jika operasi vakum perlu menggabungkan baris baru ke dalam wilayah yang diurutkan tabel, waktu yang dibutuhkan untuk vakum akan meningkat saat tabel tumbuh lebih besar. Anda dapat meningkatkan kinerja vakum dengan mengurangi jumlah baris yang harus digabungkan.

Sebelum ruang hampa, tabel terdiri dari wilayah yang diurutkan di kepala tabel, diikuti oleh wilayah yang tidak disortir, yang tumbuh setiap kali baris ditambahkan atau diperbarui. Ketika satu set baris ditambahkan oleh operasi COPY, kumpulan baris baru diurutkan pada kunci pengurutan karena ditambahkan ke wilayah yang tidak disortir di akhir tabel. Baris baru diurutkan dalam kumpulannya sendiri, tetapi tidak di dalam wilayah yang tidak disortir.

Diagram berikut menggambarkan wilayah yang tidak disortir setelah dua operasi COPY berturut-turut, di mana kunci pengurutan adalah CUSTID. Untuk kesederhanaan, contoh ini menunjukkan kunci pengurutan majemuk, tetapi prinsip yang sama berlaku untuk kunci pengurutan yang disisipkan, kecuali bahwa dampak dari wilayah yang tidak disortir lebih besar untuk tabel yang disisipkan.



Sebuah vakum mengembalikan urutan tabel dalam dua tahap:

1. Urutkan wilayah yang tidak disortir ke wilayah yang baru diurutkan.

   Tahap pertama relatif murah, karena hanya wilayah yang tidak disortir yang ditulis ulang. Jika rentang nilai kunci pengurutan dari wilayah yang baru diurutkan lebih tinggi dari rentang yang ada, hanya baris baru yang perlu ditulis ulang, dan ruang hampa selesai. Misalnya, jika wilayah yang diurutkan berisi nilai ID 1 hingga 500 dan operasi penyalinan berikutnya menambahkan nilai kunci lebih besar dari 500, maka hanya wilayah yang tidak disortir yang perlu ditulis ulang.

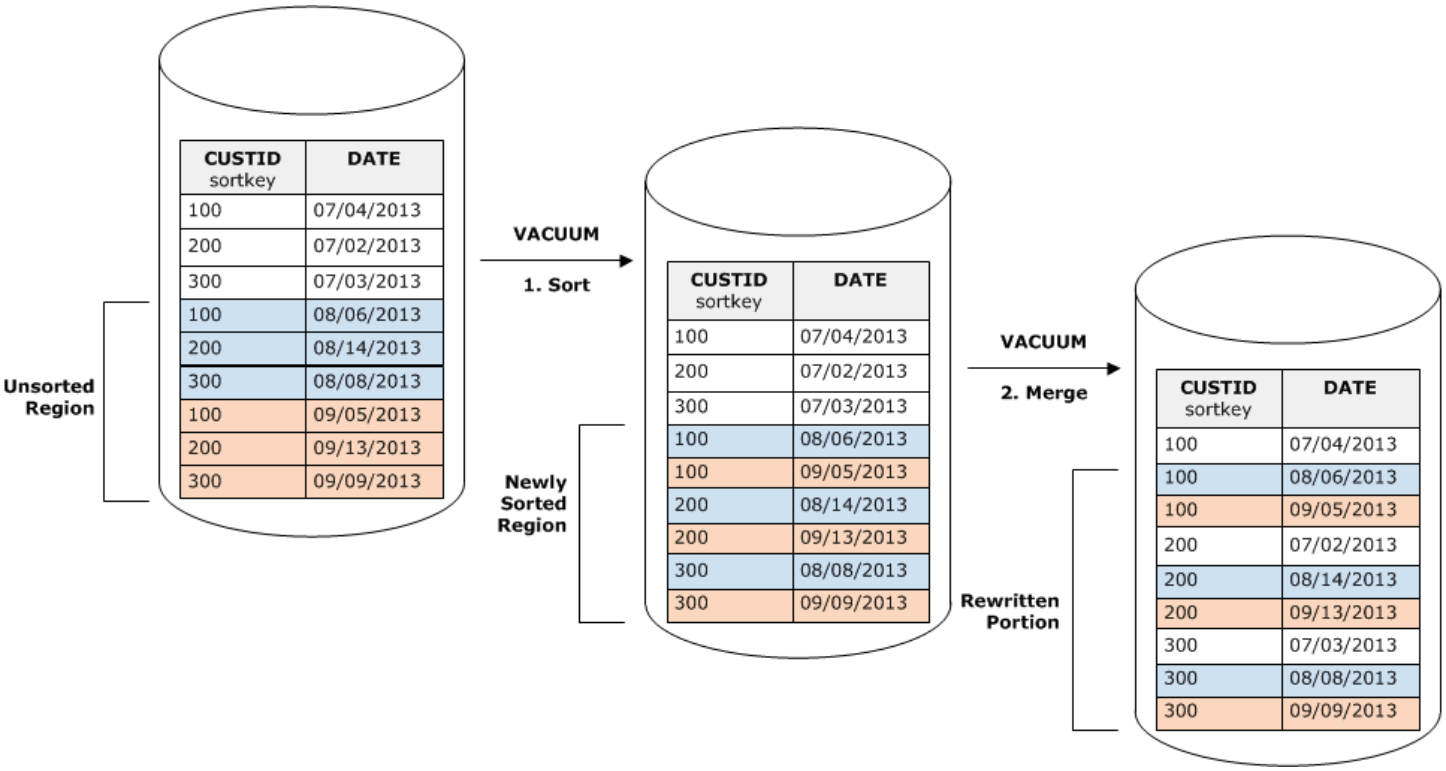
2. Gabungkan wilayah yang baru diurutkan dengan wilayah yang diurutkan sebelumnya.

   Jika kunci di wilayah yang baru diurutkan tumpang tindih dengan kunci di wilayah yang diurutkan, maka VACUUM perlu menggabungkan baris. Mulai dari awal wilayah yang baru diurutkan (pada

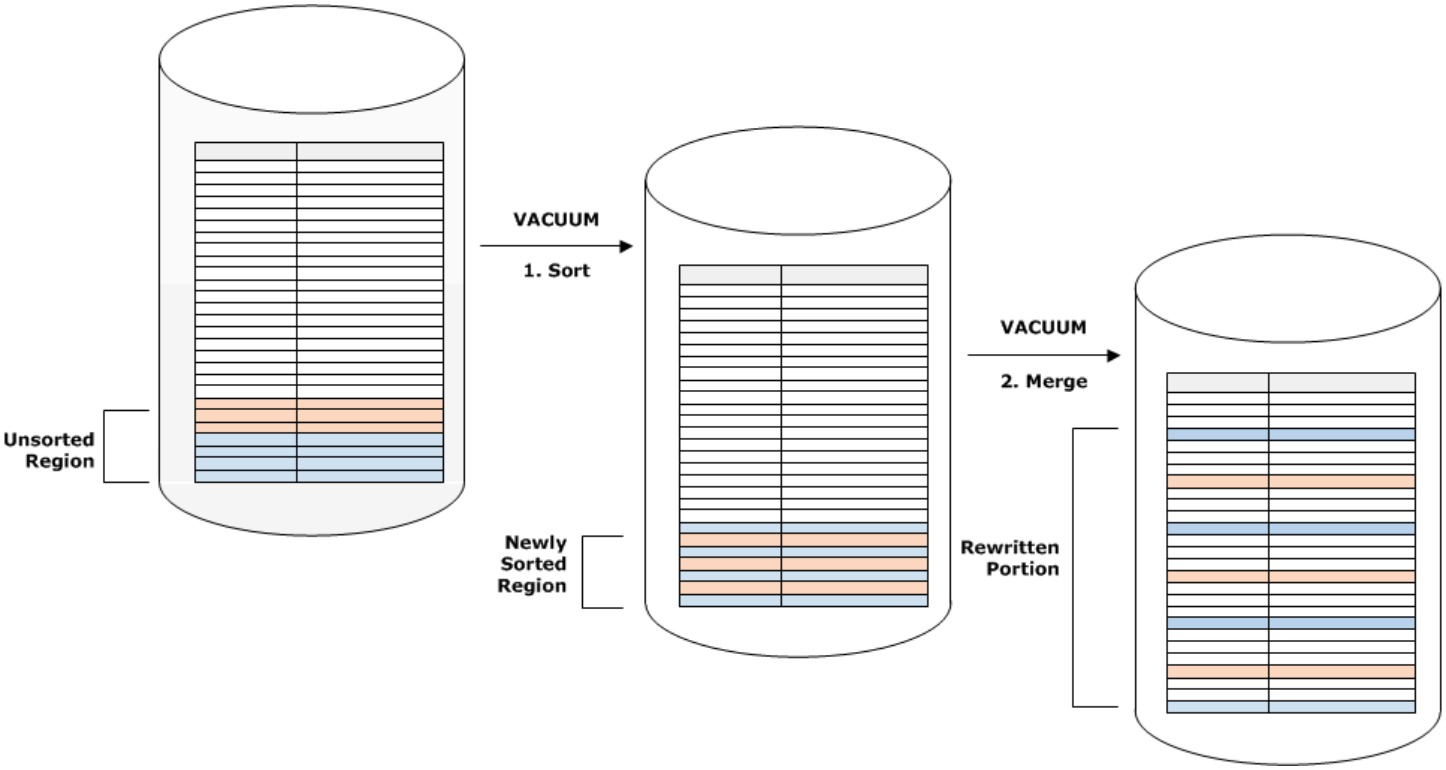
kunci pengurutan terendah), ruang hampa menulis baris gabungan dari wilayah yang diurutkan sebelumnya dan wilayah yang baru diurutkan ke dalam kumpulan blok baru.

Sejauh mana rentang kunci pengurutan baru tumpang tindih dengan kunci pengurutan yang ada menentukan sejauh mana wilayah yang diurutkan sebelumnya perlu ditulis ulang. Jika kunci yang tidak disortir tersebar di seluruh rentang pengurutan yang ada, ruang hampa mungkin perlu menulis ulang bagian tabel yang ada.

Diagram berikut menunjukkan bagaimana ruang hampa akan mengurutkan dan menggabungkan baris yang ditambahkan ke tabel di mana CUSTID adalah kunci pengurutan. Karena setiap operasi penyalinan menambahkan satu set baris baru dengan nilai kunci yang tumpang tindih dengan kunci yang ada, hampir seluruh tabel perlu ditulis ulang. Diagram menunjukkan pengurutan dan penggabungan tunggal, tetapi dalam praktiknya, ruang hampa besar terdiri dari serangkaian langkah pengurutan dan penggabungan inkremental.



Jika rentang kunci pengurutan dalam satu set baris baru tumpang tindih dengan kisaran kunci yang ada, biaya tahap penggabungan terus tumbuh sebanding dengan ukuran tabel saat tabel tumbuh sementara biaya tahap pengurutan tetap sebanding dengan ukuran wilayah yang tidak disortir. Dalam kasus seperti itu, biaya tahap penggabungan menutupi biaya tahap pengurutan, seperti yang ditunjukkan diagram berikut.



Untuk menentukan proporsi tabel yang digabungkan kembali, kueri SVV_VACUUM_SUMMARY setelah operasi vakum selesai. Kueri berikut menunjukkan efek dari enam penyedot debu berturut-turut karena CUSTSALES tumbuh lebih besar dari waktu ke waktu.

```
select * from svv_vacuum_summary
where table_name = 'custsales';


 table_name | xid  | sort_      | merge_     | elapsed_   | row_  | sortedrow_ | block_
  | max_merge_
            |      | partitions | increments | time       | delta | delta      | delta
  | partitions
 -----------+------+------------+------------+------------+-------+------------
+---------+---------------
  custsales | 7072 |          3 |          2 |  143918314 |     0 |   88297472 |   1524
  |      47
  custsales | 7122 |          3 |          3 |  164157882 |     0 |   88297472 |    772
  |      47
  custsales | 7212 |          3 |          4 |  187433171 |     0 |   88297472 |    767
  |      47
  custsales | 7289 |          3 |          4 |  255482945 |     0 |   88297472 |    770
  |      47
```

```
 custsales | 7420 |          3 |          5 |  316583833 |     0 |   88297472 |    769
 |      47
 custsales | 9007 |          3 |          6 |  306685472 |     0 |   88297472 |    772
 |      47
(6 rows)
```

Kolom merge_increments memberikan indikasi jumlah data yang digabungkan untuk setiap operasi vakum. Jika jumlah kenaikan penggabungan selama vakum berturut-turut meningkat sebanding dengan pertumbuhan ukuran tabel, ini menunjukkan bahwa setiap operasi vakum menggabungkan kembali semakin banyak baris dalam tabel karena wilayah yang ada dan yang baru diurutkan tumpang tindih.

Muat data Anda dalam urutan kunci

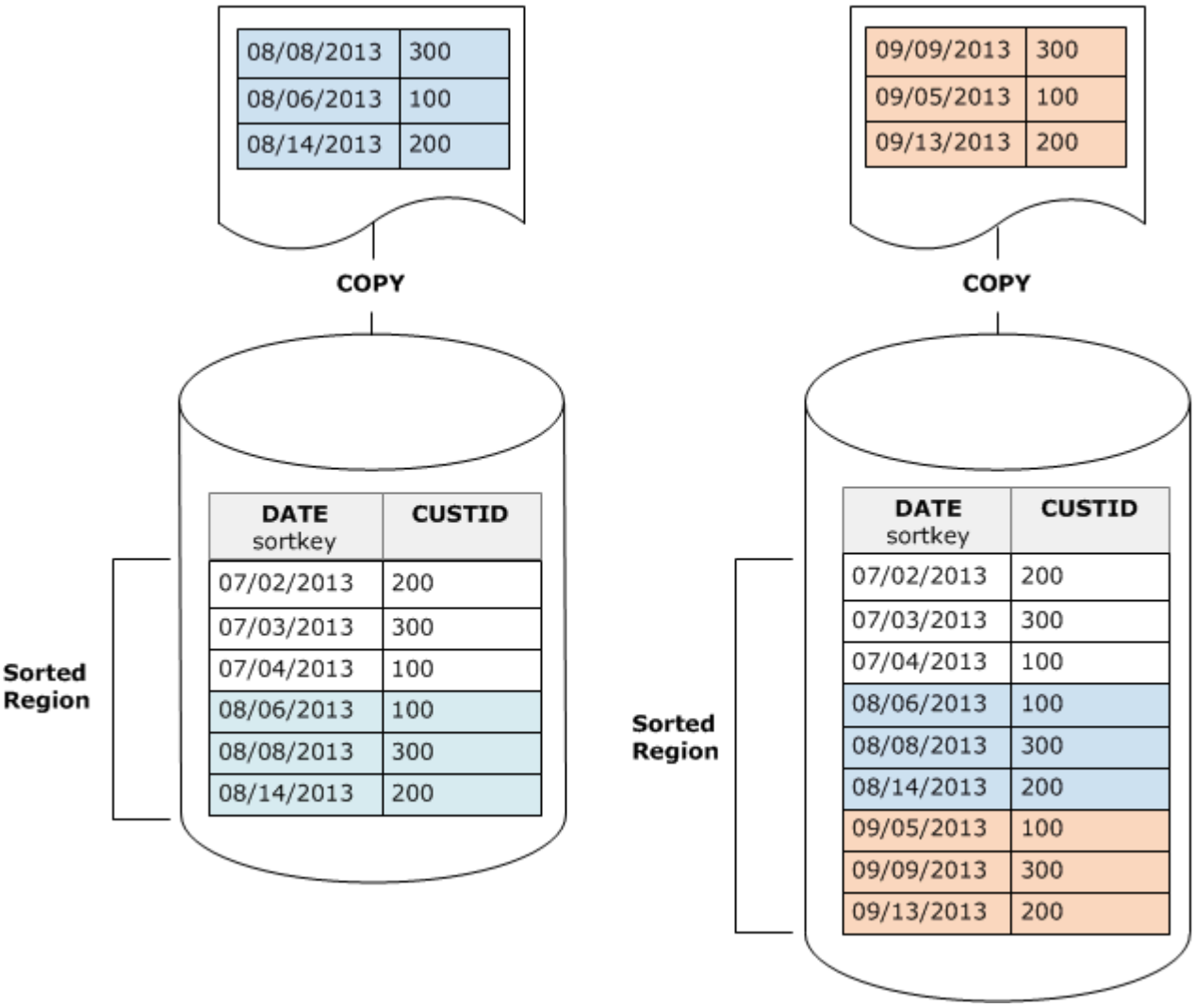
Jika Anda memuat data Anda dalam urutan kunci dengan menggunakan perintah COPY, Anda dapat mengurangi atau bahkan menghapus kebutuhan untuk menyedot debu.

COPY secara otomatis menambahkan baris baru ke wilayah tabel yang diurutkan ketika semua hal berikut benar:

- Tabel menggunakan kunci pengurutan majemuk dengan hanya satu kolom pengurutan.
- Kolom pengurutan BUKAN NULL.
- Tabel 100 persen diurutkan atau kosong.
- Semua baris baru lebih tinggi dalam urutan urutan daripada baris yang ada, termasuk baris yang ditandai untuk dihapus. Dalam contoh ini, Amazon Redshift menggunakan delapan byte pertama dari kunci pengurutan untuk menentukan urutan.
- Perintah COPY tidak memicu optimasi beban tertentu. Saat memuat volume data yang besar, Amazon Redshift mungkin mengoptimalkan kinerja dengan membuat partisi baru yang diurutkan daripada menambahkan baris ke wilayah yang diurutkan tabel.

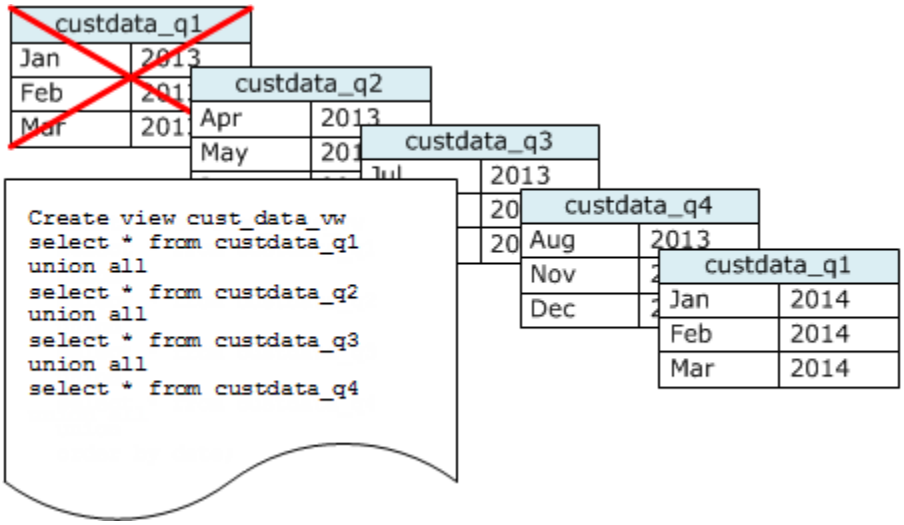
Misalnya, Anda memiliki tabel yang merekam peristiwa pelanggan menggunakan ID pelanggan dan waktu. Jika Anda mengurutkan berdasarkan ID pelanggan, kemungkinan rentang kunci pengurutan baris baru yang ditambahkan oleh beban tambahan akan tumpang tindih dengan rentang yang ada, seperti yang ditunjukkan pada contoh sebelumnya, yang mengarah ke operasi vakum yang mahal.

Jika Anda mengatur kunci pengurutan ke kolom stempel waktu, baris baru Anda akan ditambahkan dalam urutan urutan di akhir tabel, seperti yang ditunjukkan diagram berikut, mengurangi atau bahkan menghilangkan kebutuhan untuk menyedot debu.



Gunakan tabel deret waktu untuk mengurangi data yang disimpan

Jika Anda menyimpan data untuk periode waktu bergulir, gunakan serangkaian tabel, seperti yang digambarkan diagram berikut.



Buat tabel baru setiap kali Anda menambahkan satu set data, lalu hapus tabel tertua dalam seri. Anda mendapatkan keuntungan ganda:

- Anda menghindari biaya tambahan untuk menghapus baris, karena operasi DROP TABLE jauh lebih efisien daripada DELETE massal.
- Jika tabel diurutkan berdasarkan stempel waktu, tidak diperlukan vakum. Jika setiap tabel berisi data selama satu bulan, ruang hampa paling banyak harus menulis ulang data senilai satu bulan, bahkan jika tabel tidak diurutkan berdasarkan stempel waktu.

Anda dapat membuat tampilan UNION ALL untuk digunakan dengan melaporkan kueri yang menyembunyikan fakta bahwa data disimpan dalam beberapa tabel. Jika kueri memfilter pada kunci pengurutan, perencana kueri dapat secara efisien melewati semua tabel yang tidak digunakan. UNION ALL bisa kurang efisien untuk jenis kueri lain, jadi Anda harus mengevaluasi kinerja kueri dalam konteks semua kueri yang menggunakan tabel.

Mengelola operasi penulisan bersamaan

Beberapa aplikasi tidak hanya memerlukan query dan loading bersamaan, tetapi juga kemampuan untuk menulis ke beberapa tabel atau tabel yang sama secara bersamaan. Dalam konteks ini, secara bersamaan berarti tumpang tindih, tidak dijadwalkan untuk berjalan pada waktu yang bersamaan. Dua transaksi dianggap bersamaan jika yang kedua dimulai sebelum commit pertama. Operasi bersamaan dapat berasal dari sesi yang berbeda yang dikendalikan oleh pengguna yang sama atau oleh pengguna yang berbeda.

Amazon Redshift mendukung jenis aplikasi ini dengan memungkinkan tabel dibaca saat sedang dimuat atau dimodifikasi secara bertahap. Kueri hanya melihat versi komited terbaru, atau snapshot, dari data, daripada menunggu versi berikutnya untuk dikomit. Jika Anda ingin kueri tertentu menunggu komit dari operasi tulis lain, Anda harus menjadwalkannya sesuai.

> **Note**
>
> Amazon Redshift mendukung perilaku komit otomatis default di mana masing-masing menjalankan perintah SQL secara terpisah melakukan commit satu per satu. Jika Anda menyertakan satu set perintah dalam blok transaksi (ditentukan oleh AKHIR pernyataan MULAI dan), blok tersebut akan melakukan commit sebagai satu transaksi, sehingga Anda dapat memutarnya kembali jika perlu. Pengecualian untuk perilaku ini adalah perintah TRUNCATE dan VACUUM, yang secara otomatis melakukan semua perubahan yang belum dilakukan dalam transaksi saat ini.
> Beberapa klien SQL mengeluarkan perintah BEGIN dan COMMIT secara otomatis, sehingga klien mengontrol apakah sekelompok pernyataan dijalankan sebagai transaksi atau setiap pernyataan individu dijalankan sebagai transaksinya sendiri. Periksa dokumentasi untuk antarmuka yang Anda gunakan. Misalnya, saat menggunakan driver Amazon Redshift JDBC, JDBC `PreparedStatement` dengan string kueri yang berisi beberapa perintah SQL (terpisah titik koma) menjalankan semua pernyataan sebagai transaksi tunggal. Sebaliknya, jika Anda menggunakan SQL Workbench/J dan mengatur AUTO COMMIT ON, maka jika Anda menjalankan beberapa pernyataan, setiap pernyataan berjalan sebagai transaksinya sendiri.

Topik berikut menjelaskan beberapa konsep utama dan kasus penggunaan yang melibatkan transaksi, snapshot database, pembaruan, dan perilaku bersamaan.

Topik

- Tingkat isolasi di Amazon Redshift
- Menulis dan men read/write goperasikan
- Contoh penulisan bersamaan
- Memecahkan masalah kesalahan isolasi yang dapat diserialisasi

Tingkat isolasi di Amazon Redshift

Operasi penulisan bersamaan didukung di Amazon Redshift dengan cara yang protektif, menggunakan kunci tulis pada tabel dan prinsip isolasi yang dapat diserialisasi. Isolasi yang dapat diserialisasi mempertahankan ilusi bahwa transaksi yang berjalan melawan tabel adalah satu-satunya transaksi yang berjalan melawan tabel itu.

Database Amazon Redshift mendukung operasi penulisan bersamaan dengan meminta setiap operasi menggunakan versi berkomitmen terbaru, atau snapshot, dari data mereka pada awal transaksi. Snapshot database dibuat dalam transaksi pada kejadian pertama sebagian besar pernyataan SELECT, perintah DML seperti COPY, DELETE, INSERT, UPDATE, dan TRUNCATE, dan perintah DDL berikut:

- ALTER TABLE (untuk menambah atau melepaskan kolom)
- CREATE TABLE
- MEJA DROP
- MEMOTONG TABEL

Tidak ada transaksi lain yang dapat mengubah snapshot ini, artinya transaksi terisolasi satu sama lain. Artinya, transaksi bersamaan tidak terlihat satu sama lain dan tidak dapat mendeteksi perubahan satu sama lain.

Eksekusi transaksi secara bersamaan harus menghasilkan hasil yang sama dengan eksekusi serial transaksi tersebut. Jika tidak ada eksekusi serial dari transaksi tersebut yang dapat menghasilkan hasil yang sama, transaksi yang menjalankan pernyataan yang mungkin merusak kemampuan serialisasi dihentikan dan diputar kembali.

Misalnya, pengguna mencoba menjalankan dua transaksi bersamaan, T1 dan T2. Menjalankan T1 dan T2 harus menghasilkan hasil yang sama dengan setidaknya salah satu skenario berikut:

- T1 dan T2 berjalan secara serial dalam urutan itu.
- T2 dan T1 berjalan secara serial dalam urutan itu.

Tingkat isolasi di Amazon Redshift mencegah masalah berikut:

- Pembacaan kotor - Pembacaan kotor terjadi ketika transaksi membaca data yang belum dilakukan. Misalnya, transaksi 1 memperbarui baris. Transaksi 2 membaca baris yang diperbarui sebelum T1

melakukan pembaruan. Jika T1 mengembalikan perubahan, T2 akan membaca data dalam baris yang tidak terikat yang sekarang dianggap Amazon Redshift tidak pernah ada.

- Non-repeatable membaca - Pembacaan yang tidak dapat diulang terjadi ketika satu transaksi membaca baris yang sama dua kali tetapi mendapatkan data yang berbeda setiap kali. Misalnya, transaksi 1 membaca baris. Transaksi 2 memperbarui atau menghapus baris tersebut dan melakukan pembaruan atau penghapusan. Jika T1 membaca ulang baris, ia mengambil nilai baris yang berbeda atau menemukan bahwa baris telah dihapus.
- Hantu — Hantu adalah baris yang cocok dengan kriteria pencarian tetapi awalnya tidak terlihat. Misalnya, transaksi 1 membaca satu set baris yang memenuhi kriteria pencariannya. Transaksi 2 menghasilkan baris baru dalam pernyataan UPDATE atau INSERT yang cocok dengan kriteria pencarian untuk T1. Jika T1 menjalankan kembali pernyataan pencariannya, ia mendapatkan kumpulan baris yang berbeda.

ISOLASI SNAPSHOT dan SERIALIZABLE

Isolasi SNAPSHOT dan SERIALIZABLE adalah dua tingkat isolasi serialisasi yang tersedia di Amazon Redshift.

Isolasi SNAPSHOT adalah tingkat isolasi default saat membuat cluster yang disediakan dan grup kerja tanpa server, memungkinkan Anda memproses volume data yang lebih besar daripada isolasi SERIALIZABLE dalam waktu yang lebih singkat.

Isolasi SERIALIZABLE membutuhkan lebih banyak waktu, tetapi menerapkan batasan yang lebih ketat pada transaksi bersamaan. Tingkat isolasi ini mencegah masalah seperti anomali penulisan dengan hanya mengizinkan satu transaksi dilakukan, sementara membatalkan semua transaksi bersamaan lainnya dengan kesalahan pelanggaran isolasi yang dapat diserialisasi.

Berikut ini adalah contoh garis waktu tentang bagaimana dua operasi penulisan bersamaan akan ditangani saat menggunakan isolasi SNAPSHOT. Pernyataan UPDATE setiap pengguna diizinkan untuk melakukan komit karena mereka tidak bertentangan dengan mencoba memperbarui baris yang sama.

Waktu	Tindakan pengguna 1	Tindakan pengguna 2
1	MULAI;	

Waktu	Tindakan pengguna 1	Tindakan pengguna 2
2		MULAI;

3	PILIH * DARI Angka;
	`digits`
	`------`
	`0`
	`1`	

4		PILIH * DARI Angka;
		`digits`
		`------`
		`0`
		`1`

5	PERBAR Nomor SET digit = 0 DIMANA digit = 1;	

Waktu	Tindakan pengguna 1	Tindakan pengguna 2

6	PILIH * DARI Angka;
	`digits`
	`------`
	`0`
	`0`	

7	BERKOMEN;	
8		Perbarui Nomor SET digit = 1 DIMANA digit = 0;

9		PILIH * DARI Angka;
		`digits`
		`------`
		`1`
		`1`

10		BERKOMITMEN;

Waktu	Tindakan pengguna 1	Tindakan pengguna 2

11	PILIH * DARI Angka;
	`digits`
	`------`
	`1`
	`0`	

12		PILIH * DARI Angka;
		`digits`
		`------`
		`1`
		`0`

Jika skenario yang sama dijalankan menggunakan isolasi yang dapat diserialisasi, maka Amazon Redshift mengakhiri pengguna 2 karena pelanggaran serialisasi dan mengembalikan kesalahan. 1023 Untuk informasi selengkapnya, lihat Memecahkan masalah kesalahan isolasi yang dapat diserialisasi. Dalam hal ini, hanya pengguna 1 yang dapat melakukan komit dengan sukses.

Pertimbangan-pertimbangan

Saat menggunakan level isolasi di Amazon Redshift, pertimbangkan hal berikut:

- Kueri tampilan katalog STV_DB_ISOLATION_LEVEL untuk melihat tingkat isolasi yang digunakan database Anda. Untuk informasi selengkapnya, lihat STV_DB_LEVEL_ISOLASI.
- Kueri tampilan PG_DATABASE_INFO untuk melihat berapa banyak transaksi bersamaan yang didukung untuk database Anda. Untuk informasi selengkapnya, lihat PG_DATABASE_INFO.

- Tabel katalog sistem (PG) dan tabel sistem Amazon Redshift lainnya tidak terkunci dalam transaksi. Oleh karena itu, perubahan pada objek database yang muncul dari operasi DDL dan TRUNCATE terlihat pada komit ke setiap transaksi bersamaan.

  Misalnya, misalkan tabel A ada di database ketika dua transaksi bersamaan, T1 dan T2, dimulai. Misalkan T2 mengembalikan daftar tabel dengan memilih dari tabel katalog PG_TABLES. Kemudian T1 menjatuhkan tabel A dan melakukan commit, dan kemudian T2 mencantumkan tabel lagi. Tabel A sekarang tidak lagi terdaftar. Jika T2 mencoba menanyakan tabel yang dihapus, Amazon Redshift mengembalikan kesalahan “relasi tidak ada”. Kueri katalog yang mengembalikan daftar tabel ke T2 atau memeriksa bahwa tabel A ada tidak tunduk pada aturan isolasi yang sama seperti operasi yang dilakukan pada tabel pengguna.

  Transaksi untuk pembaruan pada tabel ini berjalan dalam mode isolasi berkomitmen baca.
- PG-prefix tabel katalog tidak mendukung isolasi SNAPSHOT.

Menulis dan men read/write goperasikan

Anda dapat mengelola perilaku spesifik operasi penulisan bersamaan dengan memutuskan kapan dan bagaimana menjalankan berbagai jenis perintah. Perintah berikut relevan dengan diskusi ini:

- perintah COPY, yang melakukan pemuatan (awal atau inkremental)
- perintah INSERT yang menambahkan satu atau lebih baris sekaligus
- Perintah UPDATE, yang mengubah baris yang ada
- perintah DELETE, yang menghapus baris

Operasi COPY dan INSERT adalah operasi tulis murni. Operasi DELETE dan UP read/write DATE adalah operasi (untuk baris yang akan dihapus atau diperbarui, mereka harus dibaca terlebih dahulu). Hasil operasi penulisan bersamaan bergantung pada perintah spesifik yang dijalankan secara bersamaan.

Operasi UPDATE dan DELETE berperilaku berbeda karena mereka bergantung pada tabel awal yang dibaca sebelum mereka melakukan penulisan apa pun. Mengingat bahwa transaksi bersamaan tidak terlihat satu sama lain, baik UPdates dan Delete harus membaca snapshot data dari komit terakhir. Ketika UPDATE atau DELETE pertama melepaskan kuncinya, UPDATE atau DELETE kedua perlu menentukan apakah data yang akan digunakan berpotensi basi. Itu tidak akan basi, karena transaksi kedua tidak mendapatkan snapshot datanya sampai setelah transaksi pertama melepaskan kuncinya.

Situasi kebuntuan potensial untuk transaksi penulisan bersamaan yang melibatkan beberapa tabel

Ketika transaksi melibatkan pembaruan lebih dari satu tabel, selalu ada kemungkinan transaksi yang berjalan secara bersamaan menjadi menemui jalan buntu ketika keduanya mencoba menulis ke kumpulan tabel yang sama. Transaksi melepaskan semua kunci tabelnya sekaligus ketika melakukan komit atau memutar kembali; itu tidak melepaskan kunci satu per satu.

Misalnya, anggaplah transaksi T1 dan T2 dimulai kira-kira pada waktu yang sama. Jika T1 mulai menulis ke tabel A dan T2 mulai menulis ke tabel B, kedua transaksi dapat dilanjutkan tanpa konflik. Namun, jika T1 selesai menulis ke tabel A dan perlu mulai menulis ke tabel B, itu tidak akan dapat melanjutkan karena T2 masih memegang kunci pada B. Demikian pula, jika T2 selesai menulis ke tabel B dan perlu mulai menulis ke tabel A, itu tidak akan dapat melanjutkan juga karena T1 masih memegang kunci pada A. Karena tidak ada transaksi yang dapat melepaskan kuncinya sampai semua operasi penulisannya dilakukan, tidak ada transaksi yang dapat dilanjutkan. Untuk menghindari kebuntuan semacam ini, Anda perlu menjadwalkan operasi penulisan bersamaan dengan hati-hati. Misalnya, Anda harus selalu memperbarui tabel dalam urutan yang sama dalam transaksi dan, jika menentukan kunci, kunci tabel dalam urutan yang sama sebelum Anda melakukan operasi DML apa pun.

Situasi kebuntuan potensial untuk transaksi penulisan bersamaan yang melibatkan satu tabel

Dalam lingkungan isolasi snapshot, kebuntuan dapat terjadi saat menjalankan transaksi penulisan bersamaan pada tabel yang sama. Kebuntuan isolasi snapshot terjadi ketika pernyataan INSERT atau COPY bersamaan berbagi kunci dan membuat kemajuan, dan pernyataan lain perlu melakukan operasi (operasi UPDATE, DELETE, MERGE, atau DDL) yang memerlukan kunci eksklusif pada tabel yang sama.

Pertimbangkan skenario berikut:

Transaksi 1 (T1):

```
INSERT/COPY INTO table_A;
```

Transaksi 2 (T2):

```
INSERT/COPY INTO table_A;
```

```
            <UPDATE/DELETE/MERGE/DDL statement> table_A
```

Kebuntuan dapat terjadi ketika beberapa transaksi dengan operasi INSERT atau COPY berjalan secara bersamaan pada tabel yang sama dengan kunci bersama, dan salah satu transaksi tersebut mengikuti operasi penulisan murni dengan operasi yang memerlukan kunci eksklusif, seperti pernyataan UPDATE, MERGE, DELETE, atau DDL.

Untuk menghindari kebuntuan dalam situasi ini, Anda dapat memisahkan pernyataan yang memerlukan kunci eksklusif (UPDATE/MERGE/DELETE/DDL pernyataan) ke transaksi yang berbeda sehingga INSERT/COPY pernyataan apa pun dapat berkembang secara bersamaan, dan pernyataan yang memerlukan kunci eksklusif dapat dieksekusi setelahnya. Atau, untuk transaksi dengan INSERT/COPY pernyataan dan MERGE/UPDATE/MERGE pernyataan pada tabel yang sama, Anda dapat menyertakan logika coba ulang dalam aplikasi Anda untuk mengatasi potensi kebuntuan.

Contoh penulisan bersamaan

Contoh kode semu berikut menunjukkan bagaimana transaksi dilanjutkan atau menunggu ketika mereka dijalankan secara bersamaan.

Contoh penulisan bersamaan dengan isolasi yang dapat diserialkan

Operasi COPY bersamaan ke dalam tabel yang sama dengan isolasi yang dapat diserialkan

Transaksi 1 menyalin baris ke tabel LISTING:

```
begin;
copy listing from ...;
end;
```

Transaksi 2 dimulai secara bersamaan dalam sesi terpisah dan mencoba menyalin lebih banyak baris ke tabel LISTING. Transaksi 2 harus menunggu sampai transaksi 1 melepaskan kunci tulis pada tabel LISTING, kemudian dapat dilanjutkan.

```
begin;
[waits]
copy listing from ;
end;
```

Perilaku yang sama akan terjadi jika salah satu atau kedua transaksi berisi perintah INSERT alih-alih perintah COPY.

Operasi DELETE bersamaan dari tabel yang sama dengan isolasi yang dapat diserialkan

Transaksi 1 menghapus baris dari tabel:

```
begin;
delete from listing where ...;
end;
```

Transaksi 2 dimulai secara bersamaan dan mencoba menghapus baris dari tabel yang sama. Ini akan berhasil karena menunggu transaksi 1 selesai sebelum mencoba menghapus baris.

```
begin
[waits]
delete from listing where ;
end;
```

Perilaku yang sama akan terjadi jika salah satu atau kedua transaksi berisi perintah UPDATE ke tabel yang sama alih-alih perintah DELETE.

Transaksi bersamaan dengan campuran operasi baca dan tulis dengan isolasi serial

Dalam contoh ini, transaksi 1 menghapus baris dari tabel USERS, memuat ulang tabel, menjalankan kueri COUNT (*), dan kemudian ANALYSIS, sebelum melakukan:

```
begin;
delete one row from USERS table;
copy ;
select count(*) from users;
analyze ;
end;
```

Sementara itu, transaksi 2 dimulai. Transaksi ini mencoba menyalin baris tambahan ke tabel USERS, menganalisis tabel, dan kemudian menjalankan kueri COUNT (*) yang sama dengan transaksi pertama:

```
begin;
[waits]
copy users from ...;
select count(*) from users;
analyze;
```

```
end;
```

Transaksi kedua akan berhasil karena harus menunggu yang pertama selesai. Kueri COUNT-nya akan mengembalikan hitungan berdasarkan beban yang telah diselesaikan.

Contoh penulisan bersamaan dengan isolasi snapshot

Operasi COPY bersamaan ke tabel yang sama dengan isolasi snapshot

Transaksi 1 menyalin baris ke tabel LISTING:

```
begin;
copy listing from ...;
end;
```

Transaksi 2 dimulai secara bersamaan dalam sesi terpisah dan mencoba menyalin lebih banyak baris ke tabel LISTING. Transaksi 2 dapat berlangsung secara bersamaan sampai salah satu transaksi perlu menulis data ke tabel target`listing`, di mana mereka akan berjalan secara berurutan.

```
begin;
//When the COPY statement from T1 needs to write data to the table, the COPY statement
 from T2 waits.
copy listing from ...;
end;
```

Perilaku yang sama akan terjadi jika salah satu atau kedua transaksi berisi perintah INSERT alih-alih perintah COPY.

Operasi DELETE bersamaan dari tabel yang sama dengan isolasi snapshot

Operasi DELETE atau UPDATE bersamaan dari tabel yang sama dengan isolasi snapshot berjalan sama seperti operasi yang dijalankan dengan isolasi yang dapat diserialisasi.

Transaksi bersamaan dengan campuran operasi baca dan tulis dengan isolasi snapshot

Transaksi bersamaan yang dijalankan dengan campuran operasi dengan isolasi snapshot berjalan sama seperti transaksi dengan campuran operasi yang dijalankan dengan isolasi yang dapat diserialisasi.

Memecahkan masalah kesalahan isolasi yang dapat diserialisasi

ERROR:1023 DETAIL: Pelanggaran isolasi yang dapat diserialkan pada tabel di Redshift

Saat Amazon Redshift mendeteksi kesalahan isolasi yang dapat diserialisasi, Anda akan melihat pesan kesalahan seperti berikut ini.

```
ERROR:1023 DETAIL: Serializable isolation violation on table in Redshift
```

Untuk mengatasi kesalahan isolasi yang dapat diserialisasi, Anda dapat mencoba metode berikut:

- Coba lagi transaksi yang dibatalkan.

  Amazon Redshift mendeteksi bahwa beban kerja bersamaan tidak dapat diserialkan. Ini menunjukkan kesenjangan dalam logika aplikasi, yang biasanya dapat diatasi dengan mencoba kembali transaksi yang mengalami kesalahan. Jika masalah berlanjut, coba salah satu metode lainnya.
- Pindahkan operasi apa pun yang tidak harus berada dalam transaksi atom yang sama di luar transaksi.

  Metode ini berlaku ketika operasi individu di dalam dua transaksi saling mereferensikan satu sama lain dengan cara yang dapat mempengaruhi hasil transaksi lainnya. Misalnya, dua sesi berikut masing-masing memulai transaksi.

  ```
  Session1_Redshift=# begin;
  ```

  ```
  Session2_Redshift=# begin;
  ```

  Hasil dari pernyataan SELECT di setiap transaksi mungkin dipengaruhi oleh pernyataan INSERT di yang lain. Dengan kata lain, misalkan Anda menjalankan pernyataan berikut secara serial, dalam urutan apa pun. Dalam setiap kasus, hasilnya adalah salah satu pernyataan SELECT yang mengembalikan satu baris lebih banyak daripada jika transaksi dijalankan secara bersamaan. Tidak ada urutan di mana operasi dapat berjalan secara serial yang menghasilkan hasil yang sama seperti ketika dijalankan secara bersamaan. Dengan demikian, operasi terakhir yang dijalankan menghasilkan kesalahan isolasi yang dapat diserialisasi.

  ```
  Session1_Redshift=# select * from tab1;
  ```

```
Session1_Redshift=# insert into tab2 values (1);
```

```
Session2_Redshift=# insert into tab1 values (1);
Session2_Redshift=# select * from tab2;
```

Dalam banyak kasus, hasil pernyataan SELECT tidak penting. Dengan kata lain, atomisitas operasi dalam transaksi tidak penting. Dalam kasus ini, pindahkan pernyataan SELECT di luar transaksi mereka, seperti yang ditunjukkan dalam contoh berikut.

```
Session1_Redshift=# begin;
Session1_Redshift=# insert into tab1 values (1)
Session1_Redshift=# end;
Session1_Redshift=# select * from tab2;
```

```
Session2_Redshift # select * from tab1;
Session2_Redshift=# begin;
Session2_Redshift=# insert into tab2 values (1)
Session2_Redshift=# end;
```

Dalam contoh-contoh ini, tidak ada referensi silang dalam transaksi. Kedua pernyataan INSERT tidak saling mempengaruhi. Dalam contoh ini, setidaknya ada satu urutan di mana transaksi dapat berjalan secara serial dan menghasilkan hasil yang sama seolah-olah dijalankan secara bersamaan. Ini berarti bahwa transaksi dapat diserialisasi.

- Paksa serialisasi dengan mengunci semua tabel di setiap sesi.

  Per KUNCI intah memblokir operasi yang dapat mengakibatkan kesalahan isolasi yang dapat diserialisasi. Saat Anda menggunakan perintah LOCK, pastikan untuk melakukan hal berikut:

  - Kunci semua tabel yang terpengaruh oleh transaksi, termasuk yang terpengaruh oleh pernyataan SELECT read-only di dalam transaksi.
  - Kunci tabel dalam urutan yang sama, terlepas dari urutan operasi yang dilakukan.
  - Kunci semua tabel di awal transaksi, sebelum melakukan operasi apa pun.
- Gunakan isolasi snapshot untuk transaksi bersamaan

  Gunakan perintah ALTER DATABASE dengan isolasi snapshot. Untuk informasi selengkapnya tentang parameter SNAPSHOT untuk ALTER DATABASE, lihatParameter.

ERROR:1018 DETAIL: Hubungan tidak ada

Saat menjalankan operasi Amazon Redshift bersamaan di sesi yang berbeda, Anda akan melihat pesan galat seperti berikut ini.

```
ERROR: 1018 DETAIL: Relation does not exist.
```

Transaksi di Amazon Redshift mengikuti isolasi snapshot. Setelah transaksi dimulai, Amazon Redshift mengambil snapshot dari database. Untuk seluruh siklus hidup transaksi, transaksi beroperasi pada status database seperti yang tercermin dalam snapshot. Jika transaksi dibaca dari tabel yang tidak ada di snapshot, itu akan menampilkan pesan kesalahan 1018 yang ditampilkan sebelumnya. Bahkan ketika transaksi bersamaan lainnya membuat tabel setelah transaksi mengambil snapshot, transaksi tidak dapat membaca dari tabel yang baru dibuat.

Untuk mengatasi kesalahan isolasi serialisasi ini, Anda dapat mencoba memindahkan awal transaksi ke titik di mana Anda tahu tabel ada.

Jika tabel dibuat oleh transaksi lain, titik ini setidaknya setelah transaksi itu dilakukan. Juga, pastikan bahwa tidak ada transaksi bersamaan yang dilakukan yang mungkin telah menjatuhkan tabel.

```
session1 = # BEGIN;
session1 = # DROP TABLE A;
session1 = # COMMIT;
```

```
session2 = # BEGIN;
```

```
session3 = # BEGIN;
session3 = # CREATE TABLE A (id INT);
session3 = # COMMIT;
```

```
session2 = # SELECT * FROM A;
```

Operasi terakhir yang dijalankan sebagai operasi baca oleh session2 menghasilkan kesalahan isolasi yang dapat diserialisasi. Kesalahan ini terjadi ketika session2 mengambil snapshot dan tabel telah dihapus oleh sesi yang dikomitmen1. Dengan kata lain, meskipun sesi serentakan3 telah membuat tabel, session2 tidak melihat tabel karena tidak ada di snapshot.

Untuk mengatasi kesalahan ini, Anda dapat menyusun ulang sesi sebagai berikut.

```
session1 = # BEGIN;
session1 = # DROP TABLE A;
session1 = # COMMIT;
```

```
session3 = # BEGIN;
session3 = # CREATE TABLE A (id INT);
session3 = # COMMIT;
```

```
session2 = # BEGIN;
session2 = # SELECT * FROM A;
```

Sekarang ketika session2 mengambil snapshot, session3 telah dilakukan, dan tabel ada di database. Session2 dapat membaca dari tabel tanpa kesalahan apa pun.

Tutorial: Memuat data dari Amazon S3

Dalam tutorial ini, Anda berjalan melalui proses memuat data ke tabel database Amazon Redshift Anda dari file data dalam bucket Amazon S3 dari awal hingga akhir.

Dalam tutorial ini, Anda akan melakukan hal-hal berikut:

- Unduh file data yang menggunakan format nilai yang dipisahkan koma (CSV), dibatasi karakter, dan lebar tetap.
- Buat bucket Amazon S3 dan kemudian unggah file data ke bucket.
- Luncurkan cluster Amazon Redshift dan buat tabel database.
- Gunakan perintah COPY untuk memuat tabel dari file data di Amazon S3.
- Memecahkan masalah kesalahan pemuatan dan ubah perintah COPY Anda untuk memperbaiki kesalahan.

Prasyarat

Anda memerlukan prasyarat berikut:

- AWS Akun untuk meluncurkan cluster Amazon Redshift dan membuat bucket di Amazon S3.
- Kre AWS denSIAL Anda (peran IAM) untuk memuat data pengujian dari Amazon S3. Jika Anda memerlukan peran IAM baru, buka Membuat peran IAM. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html

- Klien SQL seperti editor kueri konsol Amazon Redshift.

Tutorial ini dirancang agar dapat diambil dengan sendirinya. Selain tutorial ini, kami sarankan menyelesaikan tutorial berikut untuk mendapatkan pemahaman yang lebih lengkap tentang cara merancang dan menggunakan database Amazon Redshift:

- Panduan Memulai Amazon Redshift memandu Anda melalui proses pembuatan cluster Amazon Redshift dan memuat data sampel.

Gambaran umum

Anda dapat menambahkan data ke tabel Amazon Redshift Anda dengan menggunakan perintah INSERT atau dengan menggunakan perintah COPY. Pada skala dan kecepatan gudang data Amazon Redshift, perintah COPY berkali-kali lebih cepat dan lebih efisien daripada perintah INSERT.

Perintah COPY menggunakan arsitektur Amazon Redshift massively parallel processing (MPP) untuk membaca dan memuat data secara paralel dari beberapa sumber data. Anda dapat memuat dari file data di Amazon S3, Amazon EMR, atau host jarak jauh apa pun yang dapat diakses melalui koneksi Secure Shell (SSH). Atau Anda dapat memuat langsung dari tabel Amazon DynamoDB.

Dalam tutorial ini, Anda menggunakan perintah COPY untuk memuat data dari Amazon S3. Banyak prinsip yang disajikan di sini berlaku untuk memuat dari sumber data lain juga.

Untuk mempelajari lebih lanjut tentang menggunakan perintah COPY, lihat sumber daya berikut:

- Praktik terbaik Amazon Redshift untuk memuat data
- Memuat data dari Amazon EMR
- Memuat data dari host jarak jauh
- Memuat data dari tabel Amazon DynamoDB

Langkah 1: Buat cluster

Jika Anda sudah memiliki cluster yang ingin Anda gunakan, Anda dapat melewati langkah ini.

Untuk latihan dalam tutorial ini, gunakan cluster empat simpul.

Untuk membuat klaster DB

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.

   Menggunakan menu navigasi, pilih dasbor cluster yang disediakan.

   > **Important**
   >
   > Pastikan Anda memiliki izin yang diperlukan untuk melakukan operasi cluster. Untuk informasi tentang pemberian izin yang diperlukan, lihat Mengo torisasi Amazon Redshift untuk mengakses AWS layanan.

2. Di kanan atas, pilih AWS Wilayah tempat Anda ingin membuat cluster. Untuk keperluan tutorial ini, pilih AS Barat (Oregon).
3. Pada menu navigasi, pilih Cluster, lalu pilih Buat cluster. Halaman Buat cluster muncul.
4. Pada halaman Buat cluster masukkan parameter untuk cluster Anda. Pilih nilai Anda sendiri untuk parameter, kecuali mengubah nilai-nilai berikut:

   - Pilih **`dc2.large`** untuk jenis simpul.
   - Pilih **`4`** untuk Jumlah node.
   - Di bagian Izin klaster, pilih peran IAM dari Peran IAM yang tersedia. Peran ini harus menjadi salah satu yang Anda buat sebelumnya dan yang memiliki akses ke Amazon S3. Kemudian pilih Associate IAM role untuk menambahkannya ke daftar peran IAM terkait untuk cluster.
5. Pilih Buat klaster.

Ikuti langkah-langkah Panduan Memulai Amazon Redshift untuk menyambung ke cluster Anda dari klien SQL dan menguji koneksi. Anda tidak perlu menyelesaikan langkah-langkah Memulai yang tersisa untuk membuat tabel, mengunggah data, dan mencoba kueri contoh.

Langkah 2: Unduh file data

Pada langkah ini, Anda mengunduh satu set file data sampel ke komputer Anda. Pada langkah berikutnya, Anda mengunggah file ke bucket Amazon S3.

Untuk mengunduh file data

1. Unduh file zip: LoadingDataSampleFiles.zip.

2. Ekstrak file ke folder di komputer Anda.
3. Pastikan folder Anda berisi file-file berikut.

```
customer-fw-manifest
customer-fw.tbl-000
customer-fw.tbl-000.bak
customer-fw.tbl-001
customer-fw.tbl-002
customer-fw.tbl-003
customer-fw.tbl-004
customer-fw.tbl-005
customer-fw.tbl-006
customer-fw.tbl-007
customer-fw.tbl.log
dwdate-tab.tbl-000
dwdate-tab.tbl-001
dwdate-tab.tbl-002
dwdate-tab.tbl-003
dwdate-tab.tbl-004
dwdate-tab.tbl-005
dwdate-tab.tbl-006
dwdate-tab.tbl-007
part-csv.tbl-000
part-csv.tbl-001
part-csv.tbl-002
part-csv.tbl-003
part-csv.tbl-004
part-csv.tbl-005
part-csv.tbl-006
part-csv.tbl-007
```

Langkah 3: Unggah file ke bucket Amazon S3

Pada langkah ini, Anda membuat bucket Amazon S3 dan mengunggah file data ke bucket.

Untuk mengunggah file ke bucket Amazon S3

1. Buat bucket di Amazon S3.

   Untuk informasi selengkapnya tentang membuat bucket, lihat Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

a. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.

b. Pilih Buat bucket.

c. Pilih Wilayah AWS.

   Buat bucket di Wilayah yang sama dengan cluster Anda. Jika cluster Anda berada di Wilayah AS Barat (Oregon), pilih Wilayah AS Barat (Oregon) (us-west-2).

d. Di kotak Nama Bucket pada kotak dialog Buat bucket, masukkan nama bucket.

   Nama bucket yang Anda pilih harus unik di antara semua nama bucket yang ada di Amazon S3. Salah satu cara untuk membantu membuat nama bucket Anda unik adalah dengan menawalkannya dengan nama organisasi Anda. Nama bucket harus mematuhi aturan tertentu. Untuk informasi selengkapnya, buka Pembatasan dan batasan Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

e. Pilih default yang disarankan untuk opsi lainnya.

f. Pilih Buat bucket.

   Ketika Amazon S3 berhasil membuat bucket Anda, konsol akan menampilkan bucket kosong Anda di panel Buckets.

2. Buat folder.

   a. Pilih nama bucket baru.

   b. Pilih tombol Buat Folder.

   c. Beri nama folder baru**`load`**.

   > **Note**
   >
   > Ember yang Anda buat tidak ada di kotak pasir. Dalam latihan ini, Anda menambahkan objek ke ember nyata. Anda dibebankan jumlah nominal untuk waktu Anda menyimpan objek dalam ember. Untuk informasi selengkapnya tentang harga Amazon S3, buka halaman harga Amazon S3.

3. Unggah file data ke bucket Amazon S3 baru.

   a. Pilih nama folder data.

   b. Di wizard Unggah, pilih Tambahkan file.

Ikuti petunjuk konsol Amazon S3 untuk mengunggah semua file yang Anda unduh dan ekstrak,

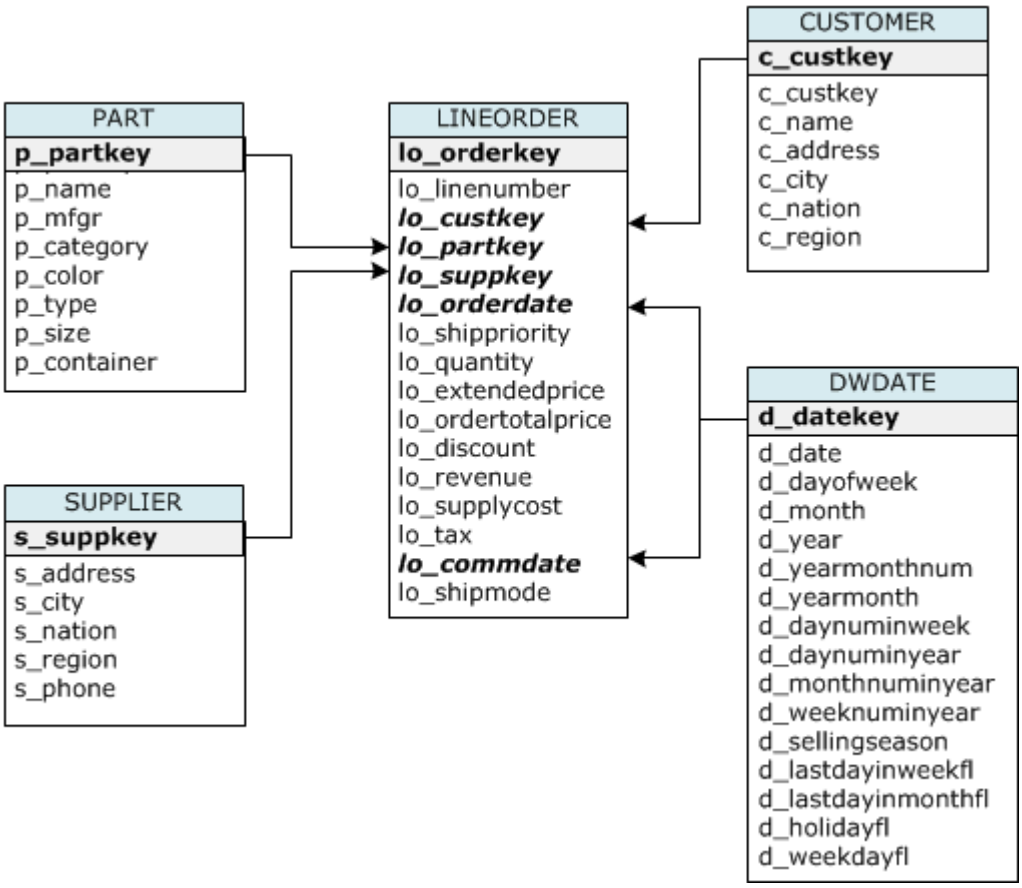
c. Pilih Unggah.

Kredensibilitas Pengguna

Perintah Amazon Redshift COPY harus memiliki akses untuk membaca objek file di bucket Amazon S3. Jika Anda menggunakan kredenSIAL pengguna yang sama untuk membuat bucket Amazon S3 dan menjalankan perintah Amazon Redshift COPY, perintah COPY memiliki semua izin yang diperlukan. Jika ingin menggunakan kredentif pengguna yang berbeda, Anda dapat memberikan akses dengan menggunakan kontrol akses Amazon S3. Perintah Amazon Redshift COPY memerlukan setidaknya ListBucket dan GetObject izin untuk mengakses objek file di bucket Amazon S3. Untuk informasi selengkapnya tentang mengendalikan akses ke sumber daya Amazon S3, buka M engelola izin akses ke sumber daya Amazon S3 Anda.

Langkah 4: Buat tabel sampel

Untuk tutorial ini, Anda menggunakan satu set tabel berdasarkan skema Star Schema Benchmark (SSB). Diagram berikut menunjukkan model data SSB.



Tabel SSB mungkin sudah ada di database saat ini. Jika demikian, jatuhkan tabel untuk menghapusnya dari database sebelum Anda membuatnya menggunakan perintah CREATE TABLE pada langkah berikutnya. Tabel yang digunakan dalam tutorial ini mungkin memiliki atribut yang berbeda dari tabel yang ada.

Untuk membuat tabel sampel

1. Untuk melepaskan tabel SSB, jalankan perintah berikut di klien SQL Anda.

```
drop table part cascade;
drop table supplier;
drop table customer;
drop table dwdate;
drop table lineorder;
```

2. Jalankan perintah CREATE TABLE berikut di klien SQL Anda.

```
CREATE TABLE part
(
```

```
  p_partkey     INTEGER NOT NULL,
  p_name        VARCHAR(22) NOT NULL,
  p_mfgr        VARCHAR(6),
  p_category    VARCHAR(7) NOT NULL,
  p_brand1      VARCHAR(9) NOT NULL,
  p_color       VARCHAR(11) NOT NULL,
  p_type        VARCHAR(25) NOT NULL,
  p_size        INTEGER NOT NULL,
  p_container   VARCHAR(10) NOT NULL
);

CREATE TABLE supplier
(
  s_suppkey   INTEGER NOT NULL,
  s_name      VARCHAR(25) NOT NULL,
  s_address   VARCHAR(25) NOT NULL,
  s_city      VARCHAR(10) NOT NULL,
  s_nation    VARCHAR(15) NOT NULL,
  s_region    VARCHAR(12) NOT NULL,
  s_phone     VARCHAR(15) NOT NULL
);

CREATE TABLE customer
(
  c_custkey      INTEGER NOT NULL,
  c_name         VARCHAR(25) NOT NULL,
  c_address      VARCHAR(25) NOT NULL,
  c_city         VARCHAR(10) NOT NULL,
  c_nation       VARCHAR(15) NOT NULL,
  c_region       VARCHAR(12) NOT NULL,
  c_phone        VARCHAR(15) NOT NULL,
  c_mktsegment   VARCHAR(10) NOT NULL
);

CREATE TABLE dwdate
(
  d_datekey            INTEGER NOT NULL,
  d_date               VARCHAR(19) NOT NULL,
  d_dayofweek          VARCHAR(10) NOT NULL,
  d_month              VARCHAR(10) NOT NULL,
  d_year               INTEGER NOT NULL,
  d_yearmonthnum       INTEGER NOT NULL,
  d_yearmonth          VARCHAR(8) NOT NULL,
  d_daynuminweek       INTEGER NOT NULL,
```

```
  d_daynuminmonth      INTEGER NOT NULL,
  d_daynuminyear       INTEGER NOT NULL,
  d_monthnuminyear     INTEGER NOT NULL,
  d_weeknuminyear      INTEGER NOT NULL,
  d_sellingseason      VARCHAR(13) NOT NULL,
  d_lastdayinweekfl    VARCHAR(1) NOT NULL,
  d_lastdayinmonthfl   VARCHAR(1) NOT NULL,
  d_holidayfl          VARCHAR(1) NOT NULL,
  d_weekdayfl          VARCHAR(1) NOT NULL
);
CREATE TABLE lineorder
(
  lo_orderkey          INTEGER NOT NULL,
  lo_linenumber        INTEGER NOT NULL,
  lo_custkey           INTEGER NOT NULL,
  lo_partkey           INTEGER NOT NULL,
  lo_suppkey           INTEGER NOT NULL,
  lo_orderdate         INTEGER NOT NULL,
  lo_orderpriority     VARCHAR(15) NOT NULL,
  lo_shippriority      VARCHAR(1) NOT NULL,
  lo_quantity          INTEGER NOT NULL,
  lo_extendedprice     INTEGER NOT NULL,
  lo_ordertotalprice   INTEGER NOT NULL,
  lo_discount          INTEGER NOT NULL,
  lo_revenue           INTEGER NOT NULL,
  lo_supplycost        INTEGER NOT NULL,
  lo_tax               INTEGER NOT NULL,
  lo_commitdate        INTEGER NOT NULL,
  lo_shipmode          VARCHAR(10) NOT NULL
);
```

Langkah 5: Jalankan perintah COPY

Anda menjalankan perintah COPY untuk memuat setiap tabel dalam skema SSB. Contoh perintah COPY menunjukkan pemuatan dari format file yang berbeda, menggunakan beberapa opsi perintah COPY, dan memecahkan masalah kesalahan pemuatan.

Sintaks perintah COPY

Sintaks MENYALIN perintah dasar adalah sebagai berikut.

```
COPY table_name [ column_list ] FROM data_source CREDENTIALS access_credentials
 [options]
```

Untuk menjalankan perintah COPY, Anda memberikan nilai-nilai berikut.

Nama tabel

Tabel target untuk perintah COPY. Tabel harus sudah ada dalam basis data. Tabel bisa bersifat sementara atau persisten. Perintah COPY menambahkan data input baru ke baris yang ada dalam tabel.

Daftar kolom

Secara default, COPY memuat bidang dari data sumber ke kolom tabel secara berurutan. Anda dapat secara opsional menentukan daftar kolom, yaitu daftar nama kolom yang dipisahkan koma, untuk memetakan bidang data ke kolom tertentu. Anda tidak menggunakan daftar kolom dalam tutorial ini. Untuk informasi selengkapnya, lihat Column List referensi perintah COPY.

Sumber data

Anda dapat menggunakan perintah COPY untuk memuat data dari bucket Amazon S3, cluster Amazon EMR, host jarak jauh menggunakan koneksi SSH, atau tabel Amazon DynamoDB. Untuk tutorial ini, Anda memuat dari file data dalam bucket Amazon S3. Saat memuat dari Amazon S3, Anda harus memberikan nama bucket dan lokasi file data. Untuk melakukan ini, berikan jalur objek untuk file data atau lokasi file manifes yang secara eksplisit mencantumkan setiap file data dan lokasinya.

- Awalan kunci

  Objek yang disimpan di Amazon S3 secara unik diidentifikasi oleh kunci objek, yang mencakup nama bucket, nama folder, jika ada, dan nama objek. A walan kunci mengacu pada sekumpulan objek dengan awalan yang sama. Jalur objek adalah awalan kunci yang digunakan perintah COPY untuk memuat semua objek yang berbagi awalan kunci. Misalnya, awalan kunci `custdata.txt` dapat merujuk ke satu file atau ke satu set file, termasuk `custdata.txt.001custdata.txt.002`,, dan sebagainya.
- Berkas manifes

  Dalam beberapa kasus, Anda mungkin perlu memuat file dengan awalan yang berbeda, misalnya dari beberapa bucket atau folder. Di tempat lain, Anda mungkin perlu mengecualikan file yang

berbagi awalan. Dalam kasus ini, Anda dapat menggunakan file manifes. File manifes secara eksplisit mencantumkan setiap file pemuatan dan kunci objek uniknya. Anda menggunakan file manifes untuk memuat tabel PART nanti dalam tutorial ini.

Kredensial

Untuk mengakses AWS sumber daya yang berisi data yang akan dimuat, Anda harus memberikan kredentif AWS akses untuk pengguna dengan hak istimewa yang memadai. Kredensibilitas ini mencakup peran IAM Nama Sumber Daya Amazon (ARN). Untuk memuat data dari Amazon S3, kredenSIAL harus menyertakan ListBucket dan GetObject izin. KredenSIAL tambahan diperlukan jika data Anda dienkripsi. Untuk informasi selengkapnya, lihat Parameter otorisasi referensi perintah COPY. Untuk informasi selengkapnya tentang mengelola akses, buka M engelola izin akses ke sumber daya Amazon S3 Anda.

Pilihan

Anda dapat menentukan sejumlah parameter dengan perintah COPY untuk menentukan format file, mengelola format data, mengelola kesalahan, dan mengontrol fitur lainnya. Dalam tutorial ini, Anda menggunakan opsi perintah COPY berikut dan fitur:

- Awalan kunci

  Untuk informasi tentang cara memuat dari beberapa file dengan menentukan awalan kunci, lihatMuat tabel PART menggunakan NULL AS.
- Format CSV

  Untuk informasi tentang cara memuat data dalam format CSV, lihatMuat tabel PART menggunakan NULL AS.
- NULL SEBAGAI

  Untuk informasi tentang cara memuat PART menggunakan opsi NULL AS, lihatMuat tabel PART menggunakan NULL AS.
- Character-delimited format

  Untuk informasi tentang cara menggunakan opsi DELIMITER, lihat. Pilihan DELIMITER dan REGION
- WILAYAH

  Untuk informasi tentang cara menggunakan opsi REGION, lihatPilihan DELIMITER dan REGION.

- Fixed-format Lebar

  Untuk informasi tentang cara memuat tabel CUSTOMER dari data lebar tetap, lihat. Muat tabel CUSTOMER menggunakan MANIFEST

- MAXERROR

  Untuk informasi tentang cara menggunakan opsi MAXERROR, lihatMuat tabel CUSTOMER menggunakan MANIFEST.

- ACCEPTINVCHARS

  Untuk informasi tentang cara menggunakan opsi ACCEPTINVCHARS, lihat. Muat tabel CUSTOMER menggunakan MANIFEST

- MANIFESTASI

  Untuk informasi tentang cara menggunakan opsi MANIFEST, lihatMuat tabel CUSTOMER menggunakan MANIFEST.

- FORMAT TANGGAL

  Untuk informasi tentang cara menggunakan opsi DATEFORMAT, lihatMuat tabel DWDATE menggunakan DATEFORMAT.

- GZIP, LZOP dan BZIP2

  Untuk informasi tentang cara mengompres file Anda, lihatMuat beberapa file data.

- COMPACTDATE

  Untuk informasi tentang cara menggunakan opsi COMPDATE, lihat. Muat beberapa file data

- Beberapa file

  Untuk informasi tentang cara memuat beberapa file, lihatMuat beberapa file data.

Memuat tabel SSB

Anda menggunakan perintah COPY berikut untuk memuat setiap tabel dalam skema SSB. Perintah untuk setiap tabel menunjukkan opsi COPY yang berbeda dan teknik pemecahan masalah.

Untuk memuat tabel SSB, ikuti langkah-langkah ini:

1. Ganti nama bucket dan AWS credentials

2. Muat tabel PART menggunakan NULL AS
3. Muat tabel CUSTOMER menggunakan MANIFEST
4. Muat tabel DWDATE menggunakan DATEFORMAT

Ganti nama bucket dan AWS credentials

Perintah COPY dalam tutorial ini disajikan dalam format berikut.

```
copy table from 's3://<your-bucket-name>/load/key_prefix'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
options;
```

Untuk setiap perintah COPY, lakukan hal berikut:

1. Ganti *<your-bucket-name>* dengan nama bucket di Wilayah yang sama dengan cluster Anda.

   Langkah ini mengasumsikan bucket dan cluster berada di Wilayah yang sama. Atau, Anda dapat menentukan Wilayah menggunakan REGION opsi dengan perintah COPY.
2. Ganti *<aws-account-id>* dan *<role-name>* dengan peran Anda sendiri Akun AWS dan IAM. Segmen string kredensional yang tertutup dalam tanda kutip tunggal tidak boleh berisi spasi atau jeda baris. Perhatikan bahwa ARN mungkin sedikit berbeda dalam format dari sampel. Yang terbaik adalah menyalin ARN untuk peran dari konsol IAM, untuk memastikan bahwa itu akurat, saat Anda menjalankan perintah COPY.

Muat tabel PART menggunakan NULL AS

Pada langkah ini, Anda menggunakan opsi CSV dan NULL AS untuk memuat tabel PART.

Perintah COPY dapat memuat data dari beberapa file secara paralel, yang jauh lebih cepat daripada memuat dari satu file. Untuk menunjukkan prinsip ini, data untuk setiap tabel dalam tutorial ini dibagi menjadi delapan file, meskipun file sangat kecil. Pada langkah selanjutnya, Anda membandingkan perbedaan waktu antara memuat dari satu file dan memuat dari beberapa file. Untuk informasi selengkapnya, lihat Memuat file data.

Awalan kunci

Anda dapat memuat dari beberapa file dengan menentukan awalan kunci untuk kumpulan file, atau dengan mencantumkan file secara eksplisit dalam file manifes. Pada langkah ini, Anda menggunakan

awalan kunci. Pada langkah selanjutnya, Anda menggunakan file manifes. Awalan `'s3://amzn-s3-demo-bucket/load/part-csv.tbl'` kunci memuat kumpulan file berikut dalam `load` folder.

```
part-csv.tbl-000
part-csv.tbl-001
part-csv.tbl-002
part-csv.tbl-003
part-csv.tbl-004
part-csv.tbl-005
part-csv.tbl-006
part-csv.tbl-007
```

Format CSV

CSV, yang merupakan singkatan dari nilai yang dipisahkan koma, adalah format umum yang digunakan untuk mengimpor dan mengekspor data spreadsheet. CSV lebih fleksibel daripada format yang dipisahkan koma karena Anda dapat menyertakan string yang dikutip dalam bidang. Karakter tanda kutip default untuk COPY from CSV format adalah tanda kutip ganda ("), tetapi Anda dapat menentukan karakter tanda kutip lain dengan menggunakan opsi QUOTE AS. Saat Anda menggunakan karakter tanda kutip di dalam bidang, hindari karakter dengan karakter tanda kutip tambahan.

Kutipan berikut dari file CSV-formatted data untuk tabel PART menunjukkan string tertutup dalam tanda kutip ganda (`"LARGE ANODIZED BRASS"`). Ini juga menunjukkan string yang tertutup dalam dua tanda kutip ganda dalam string yang dikutip (`"MEDIUM ""BURNISHED"" TIN"`).

```
15,dark sky,MFGR#3,MFGR#47,MFGR#3438,indigo,"LARGE ANODIZED BRASS",45,LG CASE
22,floral beige,MFGR#4,MFGR#44,MFGR#4421,medium,"PROMO, POLISHED BRASS",19,LG DRUM
23,bisque slate,MFGR#4,MFGR#41,MFGR#4137,firebrick,"MEDIUM ""BURNISHED"" TIN",42,JUMBO
 JAR
```

Data untuk tabel PART berisi karakter yang menyebabkan COPY gagal. Dalam latihan ini, Anda memecahkan masalah kesalahan dan memperbaikinya.

Untuk memuat data dalam format CSV, tambahkan `csv` ke perintah COPY Anda. Jalankan perintah berikut untuk memuat tabel PART.

```
copy part from 's3://<your-bucket-name>/load/part-csv.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
csv;
```

Anda mungkin mendapatkan pesan kesalahan yang mirip dengan berikut ini.

```
An error occurred when executing the SQL command:
copy part from 's3://amzn-s3-demo-bucket/load/part-csv.tbl'
credentials' ...

ERROR: Load into table 'part' failed.  Check 'stl_load_errors' system table for
 details. [SQL State=XX000]

Execution time: 1.46s

1 statement(s) failed.
1 statement(s) failed.
```

Untuk mendapatkan informasi lebih lanjut tentang kesalahan, kueri tabel STL_LOAD_ERRORS. Kueri berikut menggunakan fungsi SUBSTRING untuk mempersingkat kolom agar mudah dibaca dan menggunakan LIMIT 10 untuk mengurangi jumlah baris yang dikembalikan. Anda dapat menyesuaikan nilai `substring(filename,22,25)` untuk memungkinkan panjang nama bucket Anda.

```
select query, substring(filename,22,25) as filename,line_number as line,
substring(colname,0,12) as column, type, position as pos, substring(raw_line,0,30) as
 line_text,
substring(raw_field_value,0,15) as field_text,
substring(err_reason,0,45) as reason
from stl_load_errors
order by query desc
limit 10;
```

```
 query  |    filename      | line |  column   |    type    | pos |
--------+------------------------+-----------+------------+------------+-----+----
 333765 | part-csv.tbl-000 |    1 |           |            |   0 |

 line_text        | field_text |                  reason
------------------+------------+-----------------------------------------------
 15,NUL next,     |            | Missing newline: Unexpected character 0x2c f
```

NULL SEBAGAI

File `part-csv.tbl` data menggunakan karakter terminator NUL (`\x000`atau`\x0`) untuk menunjukkan nilai NULL.

> **Note**
>
> Meskipun ejaan sangat mirip, NUL dan NULL tidak sama. NUL adalah UTF-8 karakter dengan codepoint `x000` yang sering digunakan untuk menunjukkan akhir catatan (EOR). NULL adalah nilai SQL yang mewakili tidak adanya data.

Secara default, COPY memperlakukan karakter terminator NUL sebagai karakter EOR dan mengakhiri catatan, yang sering mengakibatkan hasil yang tidak terduga atau kesalahan. Tidak ada metode standar tunggal untuk menunjukkan NULL dalam data teks. Dengan demikian, dengan opsi perintah NULL AS COPY, Anda dapat menentukan karakter mana yang akan diganti dengan NULL saat memuat tabel. Dalam contoh ini, Anda ingin COPY memperlakukan karakter terminator NUL sebagai nilai NULL.

> **Note**
>
> Kolom tabel yang menerima nilai NULL harus dikonfigurasi sebagai nullable. Artinya, tidak boleh menyertakan batasan NOT NULL dalam spesifikasi CREATE TABLE.

Untuk memuat PART menggunakan opsi NULL AS, jalankan perintah COPY berikut.

```
copy part from 's3://<your-bucket-name>/load/part-csv.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
csv
null as '\000';
```

Untuk memverifikasi bahwa COPY memuat nilai NULL, jalankan perintah berikut untuk memilih hanya baris yang berisi NULL.

```
select p_partkey, p_name, p_mfgr, p_category from part where p_mfgr is null;
```

```
 p_partkey |  p_name  | p_mfgr | p_category
-----------+----------+--------+------------
        15 | NUL next |        | MFGR#47
        81 | NUL next |        | MFGR#23
       133 | NUL next |        | MFGR#44
(2 rows)
```

Pilihan DELIMITER dan REGION

Opsi DELIMITER dan REGION penting untuk memahami cara memuat data.

Character-delimited format

Bidang dalam file yang dibatasi karakter dipisahkan oleh karakter tertentu, seperti karakter pipa (|), koma (,) atau tab (\ t). Character-delimited file dapat menggunakan karakter ASCII tunggal, termasuk salah satu karakter ASCII yang tidak dicetak, sebagai pembatas. Anda menentukan karakter pembatas dengan menggunakan opsi DELIMITER. Pembatas default adalah karakter pipa (|).

Kutipan berikut dari data untuk tabel SUPPLIER menggunakan format dibatasi pipa.

```
1|1|257368|465569|41365|19950218|2-HIGH|0|17|2608718|9783671|4|2504369|92072|2|
19950331|TRUCK
1|2|257368|201928|8146|19950218|2-HIGH|0|36|6587676|9783671|9|5994785|109794|6|
19950416|MAIL
```

WILAYAH

Bila memungkinkan, Anda harus menemukan data muat Anda di Wilayah yang sama AWS dengan cluster Amazon Redshift Anda. Jika data dan cluster Anda berada di Wilayah yang sama, Anda mengurangi latensi dan menghindari biaya transfer data lintas wilayah. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk memuat data.

Jika Anda harus memuat data dari AWS Wilayah yang berbeda, gunakan opsi AWS REGION untuk menentukan Wilayah tempat data beban berada. Jika Anda menentukan Wilayah, semua data pemuatan, termasuk file manifes, harus berada di Region bernama. Untuk informasi selengkapnya, lihat REGION.

Misalnya, jika cluster Anda berada di Wilayah AS Timur (Virginia Utara), dan bucket Amazon S3 Anda terletak di Wilayah AS Barat (Oregon), perintah COPY berikut menunjukkan cara memuat tabel SUPPLIER dari data yang dibatasi pipa.

```
copy supplier from 's3://amzn-s3-demo-bucket/ssb/supplier.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '|'
gzip
region 'us-west-2';
```

Muat tabel CUSTOMER menggunakan MANIFEST

Pada langkah ini, Anda menggunakan opsi FIXEDWIDTH, MAXERROR, ACCEPTINVCHARS, dan MANIFEST untuk memuat tabel CUSTOMER.

Data sampel untuk latihan ini berisi karakter yang menyebabkan kesalahan saat COPY mencoba memuatnya. Anda menggunakan opsi MAXERRORS dan tabel sistem STL_LOAD_ERRORS untuk memecahkan masalah kesalahan pemuatan dan kemudian menggunakan opsi ACCEPTINVCHARS dan MANIFEST untuk menghilangkan kesalahan.

Fixed-Width Format

Fixed-width format mendefinisikan setiap bidang sebagai jumlah karakter tetap, bukan memisahkan bidang dengan pembatas. Kutipan berikut dari data untuk tabel CUSTOMER menggunakan format lebar tetap.

```
1   Customer#000000001   IVhzIApeRb            MOROCCO  0MOROCCO  AFRICA      25-705
2   Customer#000000002   XSTf4,NCwDVaWNe6tE    JORDAN   6JORDAN   MIDDLE EAST 23-453
3   Customer#000000003   MG9kdTD               ARGENTINA5ARGENTINAAMERICA     11-783
```

Urutan label/width pasangan harus sesuai dengan urutan kolom tabel dengan tepat. Untuk informasi selengkapnya, lihat FIXEDWIDTH.

String spesifikasi lebar tetap untuk data tabel CUSTOMER adalah sebagai berikut.

```
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
c_region :12, c_phone:15,c_mktsegment:10'
```

Untuk memuat tabel CUSTOMER dari data dengan lebar tetap, jalankan perintah berikut.

```
copy customer
from 's3://<your-bucket-name>/load/customer-fw.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
 c_region :12, c_phone:15,c_mktsegment:10';
```

Anda akan mendapatkan pesan kesalahan, mirip dengan yang berikut ini.

```
An error occurred when executing the SQL command:
copy customer
from 's3://amzn-s3-demo-bucket/load/customer-fw.tbl'
```

```
credentials'...

ERROR: Load into table 'customer' failed.  Check 'stl_load_errors' system table for
 details. [SQL State=XX000]

Execution time: 2.95s

1 statement(s) failed.
```

MAXERROR

Secara default, pertama kali COPY menemukan kesalahan, perintah gagal dan mengembalikan pesan kesalahan. Untuk menghemat waktu selama pengujian, Anda dapat menggunakan opsi MAXERROR untuk menginstruksikan COPY untuk melewati sejumlah kesalahan tertentu sebelum gagal. Karena Anda dapat mengharapkan kesalahan saat pertama kali Anda menguji pemuatan data tabel CUSTOMER, tambahkan `maxerror 10` ke perintah COPY.

Untuk menguji menggunakan opsi FIXEDWIDTH dan MAXERROR, jalankan perintah berikut.

```
copy customer
from 's3://<your-bucket-name>/load/customer-fw.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
 c_region :12, c_phone:15,c_mktsegment:10'
maxerror 10;
```

Kali ini, alih-alih pesan kesalahan, Anda mendapatkan pesan peringatan yang mirip dengan berikut ini.

```
Warnings:
Load into table 'customer' completed, 112497 record(s) loaded successfully.
Load into table 'customer' completed, 7 record(s) could not be loaded.  Check
 'stl_load_errors' system table for details.
```

Peringatan menunjukkan bahwa COPY mengalami tujuh kesalahan. Untuk memeriksa kesalahan, kueri tabel STL_LOAD_ERRORS, seperti yang ditunjukkan pada contoh berikut.

```
select query, substring(filename,22,25) as filename,line_number as line,
substring(colname,0,12) as column, type, position as pos, substring(raw_line,0,30) as
 line_text,
substring(raw_field_value,0,15) as field_text,
```

```
substring(err_reason,0,45) as error_reason
from stl_load_errors
order by query desc, filename
limit 7;
```

Hasil kueri STL_LOAD_ERRORS akan terlihat mirip dengan berikut ini.

```
 query  |         filename          | line |  column   |    type    | pos |
 line_text           | field_text |              error_reason
--------+---------------------------+------+-----------
+------------+-----+-------------------------------+------------
+--------------------------------------------
 334489 | customer-fw.tbl.log       |    2 | c_custkey | int4       |  -1 | customer-
fw.tbl               | customer-f | Invalid digit, Value 'c', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    6 | c_custkey | int4       |  -1 | Complete
                  | Complete   | Invalid digit, Value 'C', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    3 | c_custkey | int4       |  -1 | #Total rows
                  | #Total row | Invalid digit, Value '#', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    5 | c_custkey | int4       |  -1 | #Status
                  | #Status    | Invalid digit, Value '#', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    1 | c_custkey | int4       |  -1 | #Load file
                  | #Load file | Invalid digit, Value '#', Pos 0, Type: Integ
 334489 | customer-fw.tbl000        |    1 | c_address | varchar    |  34 | 1
 Customer#000000001  | .Mayag.ezR | String contains invalid or unsupported UTF8
 334489 | customer-fw.tbl000        |    1 | c_address | varchar    |  34 | 1
 Customer#000000001  | .Mayag.ezR | String contains invalid or unsupported UTF8
(7 rows)
```

Dengan memeriksa hasilnya, Anda dapat melihat bahwa ada dua pesan di `error_reasons` kolom:

- ```
  Invalid digit, Value '#', Pos 0, Type: Integ
  ```

  Kesalahan ini disebabkan oleh `customer-fw.tbl.log` file. Masalahnya adalah itu adalah file log, bukan file data, dan tidak boleh dimuat. Anda dapat menggunakan file manifes untuk menghindari memuat file yang salah.

- ```
  String contains invalid or unsupported UTF8
  ```

  Tipe data VARCHAR mendukung UTF-8 karakter multibyte hingga tiga byte. Jika data muat berisi karakter yang tidak didukung atau tidak valid, Anda dapat menggunakan opsi ACCEPTINVCHARS untuk mengganti setiap karakter yang tidak valid dengan karakter alternatif yang ditentukan.

Masalah lain dengan beban lebih sulit dideteksi — beban menghasilkan hasil yang tidak terduga. Untuk menyelidiki masalah ini, jalankan perintah berikut untuk menanyakan tabel CUSTOMER.

```
select c_custkey, c_name, c_address
from customer
order by c_custkey
limit 10;
```

```
 c_custkey |          c_name           |         c_address
-----------+---------------------------+---------------------------
         2 | Customer#000000002        | XSTf4,NCwDVaWNe6tE
         2 | Customer#000000002        | XSTf4,NCwDVaWNe6tE
         3 | Customer#000000003        | MG9kdTD
         3 | Customer#000000003        | MG9kdTD
         4 | Customer#000000004        | XxVSJsL
         4 | Customer#000000004        | XxVSJsL
         5 | Customer#000000005        | KvpyuHCplrB84WgAi
         5 | Customer#000000005        | KvpyuHCplrB84WgAi
         6 | Customer#000000006        | sKZz0CsnMD7mp4Xd0YrBvx
         6 | Customer#000000006        | sKZz0CsnMD7mp4Xd0YrBvx
(10 rows)
```

Baris harus unik, tetapi ada duplikat.

Cara lain untuk memeriksa hasil yang tidak terduga adalah memverifikasi jumlah baris yang dimuat. Dalam kasus kami, 100000 baris seharusnya telah dimuat, tetapi pesan beban melaporkan memuat 112497 catatan. Baris tambahan dimuat karena COPY memuat file asing,`customer-fw.tbl0000.bak`.

Dalam latihan ini, Anda menggunakan file manifes untuk menghindari memuat file yang salah.

ACCEPTINVCHARS

Secara default, ketika COPY menemukan karakter yang tidak didukung oleh tipe data kolom, ia melewatkan baris dan mengembalikan kesalahan. Untuk informasi tentang UTF-8 karakter yang tidak valid, lihatKesalahan pemuatan karakter multibyte.

Anda dapat menggunakan opsi MAXERRORS untuk mengabaikan kesalahan dan melanjutkan pemuatan, lalu kueri STL_LOAD_ERRORS untuk menemukan karakter yang tidak valid, dan kemudian memperbaiki file data. Namun, MAXERRORS paling baik digunakan untuk memecahkan masalah beban dan umumnya tidak boleh digunakan di lingkungan produksi.

Opsi ACCEPTINVCHARS biasanya merupakan pilihan yang lebih baik untuk mengelola karakter yang tidak valid. ACCEPTINVCHARS menginstruksikan COPY untuk mengganti setiap karakter yang tidak valid dengan karakter valid yang ditentukan dan melanjutkan operasi pemuatan. Anda dapat menentukan karakter ASCII yang valid, kecuali NULL, sebagai karakter pengganti. Karakter pengganti default adalah tanda tanya (?). COPY menggantikan karakter multibyte dengan string pengganti dengan panjang yang sama. Misalnya, karakter 4-byte akan diganti dengan`'????'`.

COPY mengembalikan jumlah baris yang berisi UTF-8 karakter tidak valid. Ini juga menambahkan entri ke tabel sistem STL_REPLACEMENTS untuk setiap baris yang terpengaruh, hingga maksimum 100 baris per irisan node. Karakter tambahan yang tidak valid UTF-8 juga diganti, tetapi peristiwa pengganti tersebut tidak direkam.

ACCEPTINVCHARS hanya valid untuk kolom VARCHAR.

Untuk langkah ini, Anda menambahkan ACCEPTINVCHARS dengan karakter pengganti. `'^'`

MANIFESTASI

Saat Anda MENYALIN dari Amazon S3 menggunakan awalan kunci, ada risiko Anda mungkin memuat tabel yang tidak diinginkan. Misalnya, `'s3://amzn-s3-demo-bucket/load/` folder berisi delapan file data yang berbagi awalan kuncicustomer-fw.tbl:customer-fw.tbl0000,customer-fw.tbl0001, dan seterusnya. Namun, folder yang sama juga berisi file asing `customer-fw.tbl.log` dancustomer-fw.tbl-0001.bak.

Untuk memastikan bahwa Anda memuat semua file yang benar, dan hanya file yang benar, gunakan file manifes. Manifes adalah file teks dalam format JSON yang secara eksplisit mencantumkan kunci objek unik untuk setiap file sumber yang akan dimuat. Objek file dapat berada di folder yang berbeda atau bucket yang berbeda, tetapi mereka harus berada di wilayah yang sama. Untuk informasi selengkapnya, lihat MANIFEST.

Berikut ini menunjukkan `customer-fw-manifest` teks.

```
{
  "entries": [
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-000"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-001"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-002"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-003"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-004"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-005"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-006"},
```

```
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-007"}
    ]
}
```

Untuk memuat data untuk tabel CUSTOMER menggunakan file manifes

1. Buka file `customer-fw-manifest` di editor teks.
2. Ganti *<your-bucket-name>* dengan nama bucket Anda.
3. Simpan file tersebut.
4. Unggah file ke folder muat di bucket Anda.
5. Jalankan perintah COPY berikut.

```
copy customer from 's3://<your-bucket-name>/load/customer-fw-manifest'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
 c_region :12, c_phone:15,c_mktsegment:10'
maxerror 10
acceptinvchars as '^'
manifest;
```

Muat tabel DWDATE menggunakan DATEFORMAT

Pada langkah ini, Anda menggunakan opsi DELIMITER dan DATEFORMAT untuk memuat tabel DWDATE.

Saat memuat kolom DATE dan TIMESTAMP, COPY mengharapkan format default, yaitu YYYY-MM-DD untuk tanggal dan stempel waktu YYYY-MM-DD HH:MI:SS . Jika data muat tidak menggunakan format default, Anda dapat menggunakan DATEFORMAT dan TIMEFORMAT untuk menentukan format.

Kutipan berikut menunjukkan format tanggal dalam tabel DWDATE. Perhatikan bahwa format tanggal di kolom dua tidak konsisten.

```
19920104 1992-01-04          Sunday  January 1992 199201 Jan1992 1 4 4 1...
19920112 January 12, 1992 Monday  January 1992 199201 Jan1992 2 12 12 1...
19920120 January 20, 1992 Tuesday     January 1992 199201 Jan1992 3 20 20 1...
```

FORMAT TANGGAL

Anda hanya dapat menentukan satu format tanggal. Jika data muat berisi format yang tidak konsisten, mungkin dalam kolom yang berbeda, atau jika formatnya tidak diketahui pada waktu pemuatan, Anda menggunakan DATEFORMAT dengan `'auto'` argumen. Ketika `'auto'` ditentukan, COPY mengenali format tanggal atau waktu yang valid dan mengubahnya ke format default. `'auto'`Opsi mengenali beberapa format yang tidak didukung saat menggunakan string DATEFORMAT dan TIMEFORMAT. Untuk informasi selengkapnya, lihat Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT.

Untuk memuat tabel DWDATE, jalankan perintah COPY berikut.

```
copy dwdate from 's3://<your-bucket-name>/load/dwdate-tab.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '\t'
dateformat 'auto';
```

Muat beberapa file data

Anda dapat menggunakan opsi GZIP dan COMPUPDATE untuk memuat tabel.

Anda dapat memuat tabel dari satu file data atau beberapa file. Lakukan ini untuk membandingkan waktu muat untuk kedua metode.

GZIP, LZOP dan BZIP2

Anda dapat mengompres file Anda menggunakan format kompresi gzip, lzop, atau bzip2. Saat memuat dari file terkompresi, COPY membuka kompres file selama proses pemuatan. Mengompresi file Anda menghemat ruang penyimpanan dan mempersingkat waktu unggah.

COMPACTDATE

Ketika COPY memuat tabel kosong tanpa pengkodean kompresi, itu menganalisis data beban untuk menentukan pengkodean optimal. Kemudian mengubah tabel untuk menggunakan pengkodean tersebut sebelum memulai pemuatan. Proses analisis ini membutuhkan waktu, tetapi terjadi, paling banyak, sekali per tabel. Untuk menghemat waktu, Anda dapat melewati langkah ini dengan mematikan COMPUPDATE. Untuk mengaktifkan evaluasi waktu SALINAN yang akurat, Anda menonaktifkan COMPUPDATE untuk langkah ini.

Beberapa File

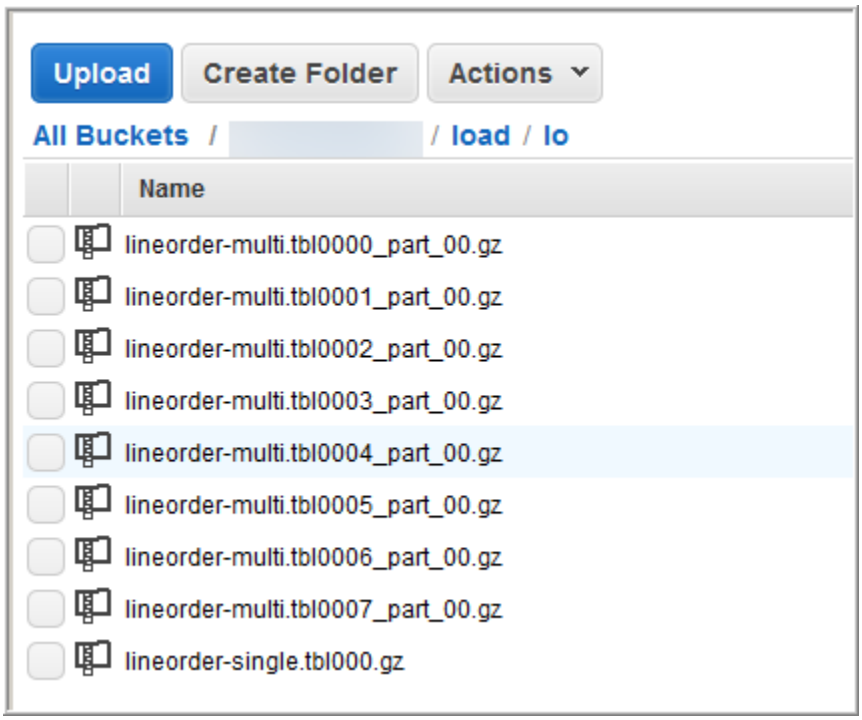
Perintah COPY dapat memuat data dengan sangat efisien ketika memuat dari beberapa file secara paralel, bukan dari satu file. Anda dapat membagi data menjadi file sehingga jumlah file adalah

kelipatan dari jumlah irisan di cluster Anda. Jika Anda melakukannya, Amazon Redshift membagi beban kerja dan mendistribusikan data secara merata di antara irisan. Jumlah irisan per node tergantung pada ukuran node cluster. Untuk informasi selengkapnya tentang jumlah irisan yang dimiliki setiap ukuran node, lihat Tentang cluster dan node di Panduan Manajemen Amazon Redshift.

Misalnya, node komputasi di cluster Anda dalam tutorial ini dapat memiliki dua irisan masing-masing, sehingga cluster empat simpul memiliki delapan irisan. Pada langkah sebelumnya, data beban terkandung dalam delapan file, meskipun file tersebut sangat kecil. Anda dapat membandingkan perbedaan waktu antara memuat dari satu file besar dan memuat dari beberapa file.

Bahkan file yang berisi 15 juta catatan dan menempati sekitar 1, 2 GB sangat kecil dalam skala Amazon Redshift. Tetapi mereka cukup untuk menunjukkan keunggulan kinerja memuat dari beberapa file.

Gambar berikut menunjukkan file data untuk LINEORDER.



Untuk mengevaluasi kinerja COPY dengan beberapa file

1. Dalam uji lab, perintah berikut dijalankan untuk COPY dari satu file. Perintah ini menunjukkan ember fiktif.

```
copy lineorder from 's3://amzn-s3-demo-bucket/load/lo/lineorder-single.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
```

```
gzip
compupdate off
region 'us-east-1';
```

2. Hasilnya adalah sebagai berikut. Perhatikan waktu eksekusi.

```
Warnings:
Load into table 'lineorder' completed, 14996734 record(s) loaded successfully.

0 row(s) affected.
copy executed successfully

Execution time: 51.56s
```

3. Kemudian perintah berikut untuk COPY dari beberapa file dijalankan.

```
copy lineorder from 's3://amzn-s3-demo-bucket/load/lo/lineorder-multi.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
gzip
compupdate off
region 'us-east-1';
```

4. Hasilnya adalah sebagai berikut. Perhatikan waktu eksekusi.

```
Warnings:
Load into table 'lineorder' completed, 14996734 record(s) loaded successfully.

0 row(s) affected.
copy executed successfully

Execution time: 17.7s
```

5. Bandingkan waktu eksekusi.

   Dalam percobaan kami, waktu untuk memuat 15 juta catatan berkurang dari 51,56 detik menjadi 17,7 detik, pengurangan 65,7 persen.

   Hasil ini didasarkan pada penggunaan cluster empat simpul. Jika cluster Anda memiliki lebih banyak node, penghematan waktu dikalikan. Untuk cluster Amazon Redshift yang khas, dengan puluhan hingga ratusan node, perbedaannya bahkan lebih dramatis. Jika Anda memiliki cluster node tunggal, ada sedikit perbedaan antara waktu eksekusi.

Langkah 6: Vakum dan analisis database

Setiap kali Anda menambahkan, menghapus, atau memodifikasi sejumlah besar baris, Anda harus menjalankan perintah VACUUM dan kemudian perintah ANALYZE. Sebuah vakum memulihkan ruang dari baris yang dihapus dan mengembalikan urutan pengurutan. Perintah ANALYZE memperbarui metadata statistik, yang memungkinkan pengoptimal kueri menghasilkan rencana kueri yang lebih akurat. Untuk informasi selengkapnya, lihat Meja penyedot debu.

Jika Anda memuat data dalam urutan kunci, penyedot debu cepat. Dalam tutorial ini, Anda menambahkan sejumlah besar baris, tetapi Anda menambahkannya ke tabel kosong. Karena itu, tidak perlu menggunakan, dan Anda tidak menghapus baris apa pun. COPY secara otomatis memperbarui statistik setelah memuat tabel kosong, jadi statistik Anda harus diperbarui. Namun, untuk urusan tata graha yang baik, Anda menyelesaikan tutorial ini dengan menyedot debu dan menganalisis database Anda.

Untuk menyedot debu dan menganalisis database, jalankan perintah berikut.

```
vacuum;
analyze;
```

Langkah 7: Bersihkan sumber daya Anda

Cluster Anda terus menambah biaya selama masih berjalan. Setelah menyelesaikan tutorial ini, Anda harus mengembalikan lingkungan Anda ke keadaan sebelumnya dengan mengikuti langkah-langkah di Langkah 5: Cabut akses dan hapus cluster sampel Anda di Panduan Mem ulai Amazon Redshift.

Jika Anda ingin menyimpan cluster, tetapi memulihkan penyimpanan yang digunakan oleh tabel SSB, jalankan perintah berikut.

```
drop table part;
drop table supplier;
drop table customer;
drop table dwdate;
drop table lineorder;
```

Selanjutnya

Ringkasan

Ringkasan

Dalam tutorial ini, Anda mengunggah file data ke Amazon S3 dan kemudian menggunakan perintah COPY untuk memuat data dari file ke tabel Amazon Redshift.

Anda memuat data menggunakan format berikut:

- Character-delimited
- CSV
- Fixed-width

Anda menggunakan tabel sistem STL_LOAD_ERRORS untuk memecahkan masalah kesalahan pemuatan, dan kemudian menggunakan opsi REGION, MANIFEST, MAXERROR, ACCEPTINVCHARS, DATEFORMAT, dan NULL AS untuk menyelesaikan kesalahan.

Anda menerapkan praktik terbaik berikut untuk memuat data:

- Gunakan perintah COPY untuk memuat data
- Memuat file data
- Gunakan satu perintah COPY untuk memuat dari beberapa file
- Mengompresi file data Anda
- Verifikasi file data sebelum dan sesudah pemuatan

Untuk informasi selengkapnya tentang praktik terbaik Amazon Redshift, lihat tautan berikut:

- Praktik terbaik Amazon Redshift untuk memuat data
- Praktik terbaik Amazon Redshift untuk merancang tabel
- Praktik terbaik Amazon Redshift untuk merancang kueri

Membongkar data di Amazon Redshift

Untuk membongkar data dari tabel database ke sekumpulan file di bucket Amazon S3, Anda dapat menggunakan perintah dengan MEMBONGKAR pernyataan SELECT. Anda dapat membongkar data teks dalam format terbatas atau format lebar tetap, terlepas dari format data yang digunakan untuk memuatnya. Anda juga dapat menentukan apakah akan membuat file GZIP terkompresi.

Anda dapat membatasi akses yang dimiliki pengguna ke bucket Amazon S3 Anda dengan menggunakan kredensil keamanan sementara.

Topik

- Membongkar data ke Amazon S3
- Membongkar file data terenkripsi
- Membongkar data dalam format lebar terbatas atau tetap
- Memuat ulang data yang dibongkar

Membongkar data ke Amazon S3

Amazon Redshift membagi hasil pernyataan pilih di satu set file, satu atau beberapa file per irisan node, untuk menyederhanakan pemuatan ulang data secara paralel. Atau, Anda dapat menentukan yang MEMBONGKAR harus menulis hasil serial ke satu atau lebih file dengan menambahkan opsi PARALLEL OFF. Anda dapat membatasi ukuran file di Amazon S3 dengan menentukan parameter MAXFILESIZE. UNLOAD secara otomatis mengenkripsi file data menggunakan enkripsi sisi server Amazon S3 (). SSE-S3

Anda dapat menggunakan pernyataan pilih apa pun dalam perintah UNLOAD yang didukung Amazon Redshift, kecuali untuk pilih yang menggunakan klausa LIMIT di seleksi luar. Misalnya, Anda dapat menggunakan pernyataan pilih yang menyertakan kolom tertentu atau yang menggunakan klausa where untuk menggabungkan beberapa tabel. Jika kueri Anda berisi tanda kutip (melampirkan nilai literal, misalnya), Anda harus menghindarinya dalam teks kueri (\ '). Untuk informasi selengkapnya, lihat referensi SELECT perintah. Untuk informasi selengkapnya tentang menggunakan klausa LIMIT, lihat Catatan penggunaan perintah UNLOAD.

Misalnya, perintah UNLOAD berikut mengirimkan konten tabel VENUE ke bucket Amazon `s3://amzn-s3-demo-bucket/tickit/unload/` S3.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Nama file yang dibuat oleh contoh sebelumnya termasuk awalan 'venue_'.

```
venue_0000_part_00
venue_0001_part_00
venue_0002_part_00
venue_0003_part_00
```

Secara default, UNLOAD menulis data secara paralel dengan beberapa file, sesuai dengan jumlah irisan dalam cluster. Untuk menulis data ke satu file, tentukan PARALLEL OFF. UNLOAD menulis data secara serial, diurutkan secara mutlak sesuai dengan klausa ORDER BY, jika digunakan. Ukuran maksimum untuk file data adalah 6,2 GB. Jika ukuran data lebih besar dari maksimum, UNLOAD membuat file tambahan, masing-masing hingga 6,2 GB.

Contoh berikut menulis konten VENUE ke satu file. Hanya satu file yang diperlukan karena ukuran file kurang dari 6,2 GB.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off;
```

> **Note**
>
> Perintah UNLOAD dirancang untuk menggunakan pemrosesan paralel. Sebaiknya biarkan PARALLEL diaktifkan untuk sebagian besar kasus, terutama jika file akan digunakan untuk memuat tabel menggunakan perintah COPY.

Dengan asumsi total ukuran data untuk VENUE adalah 5 GB, contoh berikut menulis isi VENUE ke 50 file, masing-masing berukuran 100 MB.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off
```

```
maxfilesize 100 mb;
```

Jika Anda menyertakan awalan dalam string jalur Amazon S3, UNLOAD akan menggunakan awalan tersebut untuk nama file.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Anda dapat membuat file manifes yang mencantumkan file bongkar muat dengan menentukan opsi MANIFEST dalam perintah UNLOAD. Manifes adalah file teks dalam format JSON yang secara eksplisit mencantumkan URL setiap file yang ditulis ke Amazon S3.

Contoh berikut mencakup opsi manifes.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

Contoh berikut menunjukkan manifes untuk empat file bongkar.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0000_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0001_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0002_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0003_part_00"}
  ]
}
```

File manifes dapat digunakan untuk memuat file yang sama dengan menggunakan COPY dengan opsi MANIFEST. Untuk informasi selengkapnya, lihat Menggunakan manifes untuk menentukan file data.

Setelah Anda menyelesaikan operasi UNLOAD, konfirmasikan bahwa data telah dibongkar dengan benar dengan menavigasi ke bucket Amazon S3 tempat UNLOAD menulis file. Anda akan melihat satu atau lebih file bernomor per irisan, dimulai dengan angka nol. Jika Anda menentukan opsi MANIFEST, Anda juga akan melihat file yang diakhiri dengan '`manifest`'. Contoh:

```
amzn-s3-demo-bucket/tickit/venue_0000_part_00
amzn-s3-demo-bucket/tickit/venue_0001_part_00
amzn-s3-demo-bucket/tickit/venue_0002_part_00
amzn-s3-demo-bucket/tickit/venue_0003_part_00
amzn-s3-demo-bucket/tickit/venue_manifest
```

Anda dapat secara terprogram mendapatkan daftar file yang ditulis ke Amazon S3 dengan memanggil operasi daftar Amazon S3 setelah UNLOAD selesai. Anda juga dapat menanyakan STL_UNLOAD_LOG.

Query berikut mengembalikan pathname untuk file yang dibuat oleh UNLOAD. PG_LAST_QUERY_IDFungsi mengembalikan query terbaru.

```
select query, substring(path,0,40) as path
from stl_unload_log
where query=2320
order by path;

query |             path
-------+--------------------------------------
  2320 | s3://amzn-s3-demo-bucket/venue0000_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0001_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0002_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0003_part_00
(4 rows)
```

Jika jumlah datanya sangat besar, Amazon Redshift mungkin membagi file menjadi beberapa bagian per irisan. Contoh:

```
venue_0000_part_00
venue_0000_part_01
venue_0000_part_02
venue_0001_part_00
venue_0001_part_01
venue_0001_part_02
...
```

Perintah UNLOAD berikut mencakup string yang dikutip dalam pernyataan select, sehingga tanda kutip dilolоskan (). =\'OH\' '

```
unload ('select venuename, venuecity from venue where venuestate=\'OH\' ')
```

```
to 's3://amzn-s3-demo-bucket/tickit/venue/ '
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Secara default, UNLOAD akan gagal daripada menimpa file yang ada di bucket tujuan. Untuk menimpa file yang ada, termasuk file manifes, tentukan opsi ALLOWOVERWRITE.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
allowoverwrite;
```

Membongkar file data terenkripsi

UNLOAD secara otomatis membuat file menggunakan enkripsi sisi server Amazon S3 AWS dengan kunci enkripsi terkelola (). SSE-S3 Anda juga dapat menentukan enkripsi sisi server dengan AWS Key Management Service kunci (SSE-KMS) atau enkripsi sisi klien dengan kunci yang dikelola pelanggan. UNLOAD tidak mendukung enkripsi sisi server Amazon S3 menggunakan kunci yang disediakan pelanggan. Untuk informasi selengkapnya, silakan lihat Melindungi data menggunakan enkripsi sisi server.

Untuk membongkar ke Amazon S3 menggunakan enkripsi sisi server dengan kunci, gunakan parameter KMS_KEY_ID untuk AWS KMS memberikan ID kunci seperti yang ditunjukkan pada contoh berikut.

```
unload ('select venuename, venuecity from venue')
to 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
KMS_KEY_ID '1234abcd-12ab-34cd-56ef-1234567890ab'
encrypted;
```

Jika Anda ingin memberikan kunci enkripsi Anda sendiri, Anda dapat membuat file data terenkripsi sisi klien di Amazon S3 dengan menggunakan perintah UNLOAD dengan opsi ENCRYPTED. UNLOAD menggunakan proses enkripsi amplop yang sama dengan yang digunakan enkripsi sisi klien Amazon S3. Anda kemudian dapat menggunakan perintah COPY dengan opsi ENCRYPTED untuk memuat file terenkripsi.

Prosesnya bekerja seperti ini:

1. Anda membuat kunci AES 256-bit yang dikodekan base64 yang akan Anda gunakan sebagai kunci enkripsi pribadi Anda, atau kunci simetris root.
2. Anda mengeluarkan perintah UNLOAD yang menyertakan kunci simetris root Anda dan opsi ENCRYPTED.
3. UNLOAD menghasilkan kunci simetris sekali pakai (disebut kunci simetris amplop) dan vektor inisialisasi (IV), yang digunakan untuk mengenkripsi data Anda.
4. UNLOAD mengenkripsi kunci simetris amplop menggunakan kunci simetris root Anda.
5. UNLOAD kemudian menyimpan file data terenkripsi di Amazon S3 dan menyimpan kunci amplop terenkripsi dan IV sebagai metadata objek dengan setiap file. Kunci amplop terenkripsi disimpan sebagai metadata objek `x-amz-meta-x-amz-key` dan IV disimpan sebagai metadata objek. `x-amz-meta-x-amz-iv`

Untuk informasi selengkapnya tentang proses enkripsi amplop, lihat enkripsi Client-side data dengan artikel AWS SDK for Java dan Amazon S3.

Untuk membongkar file data terenkripsi, tambahkan nilai kunci root ke string kredensial dan sertakan opsi ENCRYPTED. Jika Anda menggunakan opsi MANIFEST, file manifes juga dienkripsi.

```
unload ('select venuename, venuecity from venue')
to 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
manifest
encrypted;
```

Untuk membongkar file data terenkripsi yang dikompresi GZIP, sertakan opsi GZIP bersama dengan nilai kunci root dan opsi ENCRYPTED.

```
unload ('select venuename, venuecity from venue')
to 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted gzip;
```

Untuk memuat file data terenkripsi, tambahkan parameter MASTER_SYMMETRIC_KEY dengan nilai kunci root yang sama dan sertakan opsi ENCRYPTED.

```
copy venue from 's3://amzn-s3-demo-bucket/encrypted/venue_'
```

```
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted;
```

Membongkar data dalam format lebar terbatas atau tetap

Anda dapat membongkar data dalam format terbatas atau format lebar tetap. Output default adalah pipe-delimited (menggunakan karakter '|').

Contoh berikut menentukan koma sebagai pembatas:

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/venue/comma'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter ',';
```

File output yang dihasilkan terlihat seperti ini:

```
20,Air Canada Centre,Toronto,ON,0
60,Rexall Place,Edmonton,AB,0
100,U.S. Cellular Field,Chicago,IL,40615
200,Al Hirschfeld Theatre,New York City,NY,0
240,San Jose Repertory Theatre,San Jose,CA,0
300,Kennedy Center Opera House,Washington,DC,0
...
```

Untuk membongkar hasil yang sama yang disetel ke file yang dibatasi tab, keluarkan perintah berikut:

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/tickit/venue/tab'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t';
```

Atau, Anda dapat menggunakan spesifikasi FIXEDWIDTH. Spesifikasi ini terdiri dari pengenal untuk setiap kolom tabel dan lebar kolom (jumlah karakter). Perintah UNLOAD akan gagal daripada memotong data, jadi tentukan lebar yang setidaknya sepanjang entri terpanjang untuk kolom itu. Membongkar data dengan lebar tetap bekerja mirip dengan membongkar data yang dibatasi, kecuali bahwa output yang dihasilkan tidak mengandung karakter pembatas. Contoh:

```
unload ('select * from venue')
```

```
to 's3://amzn-s3-demo-bucket/tickit/venue/fw'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth '0:3,1:100,2:30,3:2,4:6';
```

Output lebar tetap terlihat seperti ini:

```
20 Air Canada Centre         Toronto      ON0
60 Rexall Place              Edmonton     AB0
100U.S. Cellular Field       Chicago      IL40615
200Al Hirschfeld Theatre     New York CityNY0
240San Jose Repertory TheatreSan Jose     CA0
300Kennedy Center Opera HouseWashington   DC0
```

Untuk detail selengkapnya tentang spesifikasi FIXEDWIDTH, lihat perintah. MEMBONGKAR

Memuat ulang data yang dibongkar

Untuk memuat ulang hasil operasi pembongkaran, Anda dapat menggunakan perintah COPY.

Contoh berikut menunjukkan kasus sederhana di mana tabel VENUE diturunkan menggunakan file manifes, terpotong, dan dimuat ulang.

```
unload  ('select * from venue order by venueid')
to 's3://amzn-s3-demo-bucket/tickit/venue/reload_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
delimiter '|';

truncate venue;

copy venue
from 's3://amzn-s3-demo-bucket/tickit/venue/reload_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
delimiter '|';
```

Setelah dimuat ulang, tabel VENUE terlihat seperti ini:

```
select * from venue order by venueid limit 5;
```

```
 venueid |         venuename          |  venuecity  | venuestate | venueseats
---------+----------------------------+-------------+------------+-----------
       1 | Toyota Park                | Bridgeview  | IL         |          0
       2 | Columbus Crew Stadium      | Columbus    | OH         |          0
       3 | RFK Stadium                | Washington  | DC         |          0
       4 | CommunityAmerica Ballpark | Kansas City | KS         |          0
       5 | Gillette Stadium           | Foxborough  | MA         |      68756
(5 rows)
```

User-defined fungsi di Amazon Redshift

Anda dapat membuat fungsi skalar yang ditentukan pengguna kustom (UDF) menggunakan klausa SQL SELECT atau program Python. Fungsi baru disimpan dalam database dan tersedia untuk setiap pengguna dengan hak istimewa yang cukup untuk dijalankan. Anda menjalankan UDF skalar kustom dengan cara yang hampir sama seperti Anda menjalankan fungsi Amazon Redshift yang ada.

Untuk UDF Python, selain menggunakan fungsionalitas Python standar, Anda dapat mengimpor modul Python kustom Anda sendiri. Untuk informasi selengkapnya, lihat Dukungan bahasa Python untuk UDF. Perhatikan bahwa Python 3 tidak tersedia untuk Python UDF. Untuk mendapatkan dukungan Python 3 untuk Amazon Redshift UDF, gunakan UDF Lambda Skalar sebagai gantinya.

Anda juga dapat membuat AWS Lambda UDF yang menggunakan fungsi kustom yang didefinisikan di Lambda sebagai bagian dari kueri SQL Anda. Dengan Lambda UDF, Anda dapat menulis UDF yang kompleks dan mengintegrasikan dengan komponen pihak ketiga. Mereka juga dapat membantu Anda mengatasi beberapa keterbatasan Python dan SQL UDF saat ini. Misalnya, mereka dapat membantu Anda mengakses sumber daya jaringan dan penyimpanan dan menulis pernyataan SQL yang lebih lengkap. Anda dapat membuat Lambda UDF di salah satu bahasa pemrograman yang didukung oleh Lambda, seperti Java, Go,,, C# PowerShell Node.js, Python, dan Ruby. Atau Anda dapat menggunakan runtime khusus.

Secara default, semua pengguna dapat menjalankan UDF. Untuk informasi selengkapnya tentang hak istimewa, lihatKeamanan dan izin UDF.

Topik

- Keamanan dan izin UDF
- Mencegah konflik penamaan UDF
- UDF SQL skalar
- UDF Python skalar
- UDF Lambda Skalar
- Contoh kasus penggunaan untuk fungsi yang ditentukan pengguna (UDF)

Keamanan dan izin UDF

Untuk membuat UDF, Anda harus memiliki izin untuk penggunaan bahasa untuk SQL atau plpythonu (Python). Secara default, USAGE ON LANGUAGE SQL diberikan kepada PUBLIK, tetapi Anda harus

secara eksplisit memberikan USAGE ON LANGUAGE PLPYTHONU kepada pengguna atau grup tertentu.

Untuk mencabut penggunaan SQL, pertama-tama cabut penggunaan dari PUBLIK. Kemudian berikan penggunaan pada SQL hanya untuk pengguna atau grup tertentu yang diizinkan untuk membuat SQL UDF. Contoh berikut mencabut penggunaan pada SQL dari PUBLIK. Kemudian memberikan penggunaan ke grup penggunaudf_devs.

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

Untuk menjalankan UDF, Anda harus memiliki izin untuk melakukannya untuk setiap fungsi. Secara default, izin untuk menjalankan UDF baru diberikan kepada PUBLIC. Untuk membatasi penggunaan, cabut izin ini dari PUBLIK untuk fungsi tersebut. Kemudian berikan hak istimewa kepada individu atau kelompok tertentu.

Contoh berikut mencabut eksekusi pada fungsi f_py_greater dari PUBLIK. Kemudian memberikan penggunaan ke grup penggunaudf_devs.

```
revoke execute on function f_py_greater(a float, b float) from PUBLIC;
grant execute on function f_py_greater(a float, b float) to group udf_devs;
```

Pengguna super memiliki semua hak istimewa secara default.

Untuk informasi selengkapnya, lihat HIBAH dan MENCABUT.

Mencegah konflik penamaan UDF

Anda dapat menghindari potensi konflik dan hasil yang tidak terduga dengan mempertimbangkan konvensi penamaan UDF Anda sebelum implementasi. Karena nama fungsi dapat kelebihan beban, mereka dapat bertabrakan dengan nama fungsi Amazon Redshift yang ada dan yang akan datang. Topik ini membahas kelebihan beban dan menyajikan strategi untuk menghindari konflik.

Membebani nama fungsi

Suatu fungsi diidentifikasi dengan nama dan tanda tangannya, yang merupakan jumlah argumen input dan tipe data dari argumen. Dua fungsi dalam skema yang sama dapat memiliki nama yang sama jika mereka memiliki tanda tangan yang berbeda. Dengan kata lain, nama fungsi dapat kelebihan beban.

Saat Anda menjalankan kueri, mesin kueri menentukan fungsi mana yang akan dipanggil berdasarkan jumlah argumen yang Anda berikan dan tipe data argumen. Anda dapat menggunakan kelebihan beban untuk mensimulasikan fungsi dengan jumlah variabel argumen, hingga batas yang diizinkan oleh CREATE FUNCTION perintah.

Menghindari konflik dengan fungsi Amazon Redshift bawaan

Sebaiknya beri nama semua UDF menggunakan aw `f_` alan. Amazon Redshift menyimpan aw `f_` alan khusus untuk UDF dan dengan mengawali nama UDF Anda dengan`f_`, Anda memastikan bahwa nama UDF Anda tidak akan bertentangan dengan nama fungsi SQL bawaan Amazon Redshift yang ada atau yang akan datang. Misalnya, dengan memberi nama UDF baru`f_sum`, Anda menghindari konflik dengan fungsi Amazon Redshift SUM. Demikian pula, jika Anda memberi nama fungsi baru`f_fibonacci`, Anda menghindari konflik jika Amazon Redshift menambahkan fungsi bernama FIBONACCI dalam rilis mendatang.

Anda dapat membuat UDF dengan nama dan tanda tangan yang sama dengan fungsi SQL bawaan Amazon Redshift yang ada tanpa nama fungsi kelebihan beban jika UDF dan fungsi bawaan ada dalam skema yang berbeda. Karena fungsi bawaan ada dalam skema katalog sistem, pg_catalog, Anda dapat membuat UDF dengan nama yang sama di skema lain, seperti skema publik atau yang ditentukan pengguna. Dalam beberapa kasus, Anda mungkin memanggil fungsi yang tidak secara eksplisit memenuhi syarat dengan nama skema. Jika demikian, Amazon Redshift mencari skema pg_catalog terlebih dahulu secara default. Dengan demikian, fungsi bawaan berjalan sebelum UDF baru dengan nama yang sama.

Anda dapat mengubah perilaku ini dengan mengatur jalur pencarian untuk menempatkan pg_catalog di akhir. Jika Anda melakukannya, UDF Anda diutamakan daripada fungsi bawaan, tetapi latihan ini dapat menyebabkan hasil yang tidak terduga. Mengadopsi strategi penamaan yang unik, seperti menggunakan awalan yang dicad `f_` angkan, adalah praktik yang lebih dapat diandalkan. Untuk informasi selengkapnya, lihat SET dan search_path.

UDF SQL skalar

Sebuah SQL UDF skalar menggabungkan klausa SQL SELECT yang berjalan ketika fungsi dipanggil dan mengembalikan nilai tunggal. Per CREATE FUNCTION intah mendefinisikan parameter berikut:

- (Opsional) Masukan argumen. Setiap argumen harus memiliki tipe data.
- Satu tipe data pengembalian.

- Satu klausa SQL SELECT. Dalam klausa SELECT, lihat argumen input menggunakan $1, $2, dan seterusnya, sesuai dengan urutan argumen dalam definisi fungsi.

Jenis data input dan return dapat berupa tipe data Amazon Redshift standar apa pun.

Jangan sertakan klausa FROM dalam klausa SELECT Anda. Sebagai gantinya, sertakan klausa FROM dalam pernyataan SQL yang memanggil SQL UDF.

Klausa SELECT tidak dapat menyertakan salah satu dari jenis klausa berikut:

- FROM
- KE DALAM
- WHERE
- GROUP BY
- ORDER BY
- LIMIT

Contoh fungsi SQL skalar

Contoh berikut menciptakan fungsi yang membandingkan dua angka dan mengembalikan nilai yang lebih besar. Untuk informasi selengkapnya, lihat CREATE FUNCTION.

```
create function f_sql_greater (float, float)
  returns float
stable
as $$
  select case when $1 > $2 then $1
    else $2
  end
$$ language sql;
```

Kueri berikut memanggil fungsi f_sql_larger baru untuk menanyakan tabel SALES dan mengembalikan COMMISSION atau 20 persen dari PRICEPAID, mana yang lebih besar.

```
select f_sql_greater(commission, pricepaid*0.20) from sales;
```

UDF Python skalar

Sebuah UDF Python skalar menggabungkan program Python yang berjalan ketika fungsi dipanggil dan mengembalikan nilai tunggal. Per CREATE FUNCTION intah mendefinisikan parameter berikut:

- (Opsional) Masukan argumen. Setiap argumen harus memiliki nama dan tipe data.
- Satu tipe data pengembalian.
- Satu program Python yang dapat dieksekusi.

Jenis data input dan return untuk Python UDF dapat berupa salah satu jenis berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- BOOLEAN
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- ELEMEN APA PUN

Alias untuk jenis ini juga valid. Untuk daftar lengkap tipe data dan aliasnya, lihatJenis Data.

Ketika Python UDF menggunakan tipe data ANYELEMENT, Amazon Redshift secara otomatis mengonversi ke tipe data standar berdasarkan argumen yang diberikan saat runtime. Untuk informasi selengkapnya, lihat Tipe data ANYELEMENT.

Saat kueri Amazon Redshift memanggil UDF skalar, langkah-langkah berikut terjadi saat runtime:

1. Fungsi mengubah argumen input ke tipe data Python.

   Untuk pemetaan tipe data Amazon Redshift ke tipe data Python, lihatJenis data Python UDF.

2. Fungsi menjalankan program Python, melewati argumen input yang dikonversi.
3. Kode Python mengembalikan nilai tunggal. Tipe data dari nilai pengembalian harus sesuai dengan tipe data RETURNS yang ditentukan oleh definisi fungsi.
4. Fungsi mengubah nilai pengembalian Python ke tipe data Amazon Redshift yang ditentukan, lalu mengembalikan nilai tersebut ke kueri.

> **Note**
>
> Python 3 tidak tersedia untuk Python UDF. Untuk mendapatkan dukungan Python 3 untuk Amazon Redshift UDF, gunakan UDF Lambda Skalar sebagai gantinya.

Contoh UDF Python Scalar

Contoh berikut menciptakan fungsi yang membandingkan dua angka dan mengembalikan nilai yang lebih besar. Perhatikan bahwa lekukan kode antara tanda dolar ganda ($$) adalah persyaratan Python. Untuk informasi selengkapnya, lihat CREATE FUNCTION.

```
create function f_py_greater (a float, b float)
  returns float
stable
as $$
  if a > b:
    return a
  return b
$$ language plpythonu;
```

Kueri berikut memanggil `f_greater` fungsi baru untuk menanyakan tabel PENJUALAN dan mengembalikan COMMISSION atau 20 persen dari PRICEPAID, mana yang lebih besar.

```
select f_py_greater (commission, pricepaid*0.20) from sales;
```

Jenis data Python UDF

Python UDF dapat menggunakan tipe data Amazon Redshift standar apa pun untuk argumen input dan nilai pengembalian fungsi. Selain tipe data standar, UDF mendukung tipe data ANYELEMENT, yang secara otomatis dikonversi Amazon Redshift ke tipe data standar berdasarkan argumen yang

diberikan saat runtime. UDF skalar dapat mengembalikan tipe data ANYELEMENT. Untuk informasi selengkapnya, lihat Tipe data ANYELEMENT.

Selama eksekusi, Amazon Redshift mengonversi argumen dari tipe data Amazon Redshift ke tipe data Python untuk diproses. Kemudian mengubah nilai pengembalian dari tipe data Python ke tipe data Amazon Redshift yang sesuai. Untuk informasi selengkapnya tentang jenis data Amazon Redshift, lihatJenis Data.

Tabel berikut memetakan tipe data Amazon Redshift ke tipe data Python.

Jenis data Amazon Redshift	Jenis data Python
smallint	int
integer	
bigint	
pendek	
long	
desimal atau numerik	desimal
double	float
real	
boolean	bool
char	string
varchar	
timestamp	datetime

Tipe data ANYELEMENT

ANYELEMENT adalah tipe data polimorfik. Ini berarti bahwa jika fungsi dideklarasikan menggunakan ANYELEMENT untuk tipe data argumen, fungsi tersebut dapat menerima tipe data Amazon Redshift standar apa pun sebagai input untuk argumen tersebut saat fungsi dipanggil.

Argumen ANYELEMENT diatur ke tipe data yang benar-benar diteruskan ke sana ketika fungsi dipanggil.

Jika fungsi menggunakan beberapa tipe data ANYELEMENT, mereka semua harus menyelesaikan ke tipe data aktual yang sama ketika fungsi dipanggil. Semua tipe data argumen ANYELEMENT diatur ke tipe data aktual dari argumen pertama yang diteruskan ke ANYELEMENT. Misalnya, fungsi yang dinyat `f_equal(anyelement, anyelement)` akan sebagai akan mengambil dua nilai input, asalkan mereka memiliki tipe data yang sama.

Jika nilai pengembalian suatu fungsi dideklarasikan sebagai ANYELEMENT, setidaknya satu argumen input harus ANYELEMENT. Tipe data aktual untuk nilai pengembalian sama dengan tipe data aktual yang disediakan untuk argumen input ANYELEMENT.

Dukungan bahasa Python untuk UDF

Anda dapat membuat UDF khusus berdasarkan bahasa pemrograman Python. P ustaka standar Python 2.7 tersedia untuk digunakan di UDF, dengan pengecualian modul berikut:

- ScrolledText
- Tix
- Tkinter
- tk
- kura-kura
- smtpd

Selain Pustaka Standar Python, modul-modul berikut adalah bagian dari implementasi Amazon Redshift:

- numpy 1.8.2
- panda 0.14.1
- Python Data Tool 2.2
- pytz 2014.7
- scipy 0.12.1
- enam 1.3.0
- wsgiref 0.1.2

Anda juga dapat mengimpor modul Python kustom Anda sendiri dan membuatnya tersedia untuk digunakan di UDF dengan menjalankan BUAT PERPUSTAKAAN perintah. Untuk informasi selengkapnya, lihat Contoh: Mengimpor modul pustaka Python khusus.

> **Important**
>
> Amazon Redshift memblokir semua akses jaringan dan akses tulis ke sistem file melalui UDF.

> **Note**
>
> Python 3 tidak tersedia untuk Python UDF. Untuk mendapatkan dukungan Python 3 untuk Amazon Redshift UDF, gunakan UDF Lambda Skalar sebagai gantinya.

Contoh: Mengimpor modul pustaka Python khusus

Anda mendefinisikan fungsi skalar menggunakan sintaks bahasa Python. Anda dapat menggunakan modul Python Standard Library dan modul Amazon Redshift yang sudah diinstal sebelumnya. Anda juga dapat membuat modul pustaka Python kustom Anda sendiri dan mengimpor pustaka ke cluster Anda, atau menggunakan pustaka yang ada dari Python atau pihak ketiga.

Anda tidak dapat membuat pustaka yang berisi modul dengan nama yang sama dengan modul Pustaka Standar Python atau modul Python yang sudah diinstal sebelumnya Amazon Redshift. Jika pustaka yang diinstal pengguna yang ada menggunakan paket Python yang sama dengan pustaka yang Anda buat, Anda harus menghapus pustaka yang ada sebelum menginstal pustaka baru.

Anda harus menjadi pengguna super atau memiliki `USAGE ON LANGUAGE plpythonu` hak istimewa untuk menginstal pustaka khusus; namun, setiap pengguna dengan hak istimewa yang cukup untuk membuat fungsi dapat menggunakan pustaka yang diinstal. Anda dapat menanyakan katalog PG_LIBRARY sistem untuk melihat informasi tentang pustaka yang diinstal pada cluster Anda.

Mengimpor modul Python khusus ke cluster Anda

Bagian ini memberikan contoh mengimpor modul Python kustom ke cluster Anda. Untuk melakukan langkah-langkah di bagian ini, Anda harus memiliki bucket Amazon S3, tempat Anda mengunggah paket perpustakaan. Anda kemudian menginstal paket di cluster Anda. Untuk informasi selengkapnya

tentang membuat bucket, buka Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Dalam contoh ini, misalkan Anda membuat UDF untuk bekerja dengan posisi dan jarak dalam data Anda. Hubungkan ke cluster Amazon Redshift Anda dari alat klien SQL, dan jalankan perintah berikut untuk membuat fungsi.

```
CREATE FUNCTION f_distance (x1 float, y1 float, x2 float, y2 float) RETURNS float
 IMMUTABLE as $$
    def distance(x1, y1, x2, y2):
        import math
        return math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2)

    return distance(x1, y1, x2, y2)
$$ LANGUAGE plpythonu;

CREATE FUNCTION f_within_range (x1 float, y1 float, x2 float, y2 float) RETURNS bool
 IMMUTABLE as $$
    def distance(x1, y1, x2, y2):
        import math
        return math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2)

    return distance(x1, y1, x2, y2) < 20
$$ LANGUAGE plpythonu;
```

Perhatikan bahwa beberapa baris kode diduplikasi dalam fungsi sebelumnya. Duplikasi ini diperlukan karena UDF tidak dapat mereferensikan isi UDF lain, dan kedua fungsi memerlukan fungsionalitas yang sama. Namun, alih-alih menduplikasi kode dalam beberapa fungsi, Anda dapat membuat pustaka khusus dan mengonfigurasi fungsi Anda untuk menggunakannya.

Untuk melakukannya, pertama buat paket perpustakaan dengan mengikuti langkah-langkah ini:

1. Buat folder bernama geometri. Folder ini adalah paket tingkat atas perpustakaan.
2. Di folder geometri, buat file bernama`__init__.py`. Perhatikan bahwa nama file berisi dua karakter garis bawah ganda. File ini menunjukkan kepada Python bahwa paket dapat diinisialisasi.
3. Juga di folder geometri, buat folder bernama trig. Folder ini adalah subpaket perpustakaan.
4. Di folder trig, buat file lain bernama `__init__.py` dan file bernama`line.py`. Dalam folder ini, `__init__.py` menunjukkan kepada Python bahwa subpaket dapat diinisialisasi dan itu `line.py` adalah file yang berisi kode perpustakaan.

Struktur folder dan file Anda harus sama seperti berikut:

```
geometry/
   __init__.py
   trig/
      __init__.py
      line.py
```

Untuk informasi lebih lanjut tentang struktur paket, buka Modul dalam tutorial Python di situs web Python.

5. Kode berikut berisi fungsi kelas dan anggota untuk perpustakaan. Salin dan tempel `keline.py`.

```
class LineSegment:
  def __init__(self, x1, y1, x2, y2):
    self.x1 = x1
    self.y1 = y1
    self.x2 = x2
    self.y2 = y2
  def angle(self):
    import math
    return math.atan2(self.y2 - self.y1, self.x2 - self.x1)
  def distance(self):
    import math
    return math.sqrt((self.y2 - self.y1) ** 2 + (self.x2 - self.x1) ** 2)
```

Setelah Anda membuat paket, lakukan hal berikut untuk menyiapkan paket dan mengunggahnya ke Amazon S3.

1. Kompres isi folder geometri menjadi file.zip bernama geometry.zip. Jangan sertakan folder geometri itu sendiri; hanya menyertakan isi folder seperti yang ditunjukkan berikut:

```
geometry.zip
   __init__.py
   trig/
      __init__.py
      line.py
```

2. Ung gah geometry.zip ke bucket Amazon S3 Anda.

> **Important**
>
> Jika bucket Amazon S3 tidak berada di Wilayah yang sama dengan cluster Amazon Redshift Anda, Anda harus menggunakan opsi REGION untuk menentukan Wilayah tempat data berada. Untuk informasi selengkapnya, lihat BUAT PERPUSTAKAAN.

3. Dari alat klien SQL Anda, jalankan perintah berikut untuk menginstal pustaka. Ganti *<bucket_name>* dengan nama bucket Anda, dan ganti *<access key id>* dan *<secret key>* dengan kunci akses dan kunci akses rahasia dari kredentif pengguna AWS Identity and Access Management (IAM) Anda.

```
CREATE LIBRARY geometry LANGUAGE plpythonu FROM 's3://<bucket_name>/geometry.zip'
 CREDENTIALS 'aws_access_key_id=<access key id>;aws_secret_access_key=<secret key>';
```

Setelah Anda menginstal pustaka di cluster Anda, Anda perlu mengonfigurasi fungsi Anda untuk menggunakan pustaka. Untuk melakukan ini, jalankan perintah berikut.

```
CREATE OR REPLACE FUNCTION f_distance (x1 float, y1 float, x2 float, y2 float) RETURNS
 float IMMUTABLE as $$
    from trig.line import LineSegment

    return LineSegment(x1, y1, x2, y2).distance()
$$ LANGUAGE plpythonu;

CREATE OR REPLACE FUNCTION f_within_range (x1 float, y1 float, x2 float, y2 float)
 RETURNS bool IMMUTABLE as $$
    from trig.line import LineSegment

    return LineSegment(x1, y1, x2, y2).distance() < 20
$$ LANGUAGE plpythonu;
```

Dalam perintah sebelumnya, `import trig/line` menghilangkan kode duplikat dari fungsi asli di bagian ini. Anda dapat menggunakan kembali fungsionalitas yang disediakan oleh pustaka ini di beberapa UDF. Perhatikan bahwa untuk mengimpor modul, Anda hanya perlu menentukan jalur ke subpaket dan nama modul (`trig/line`).

Kendala UDF Python

Dalam batasan yang tercantum dalam topik ini, Anda dapat menggunakan UDF di mana pun Anda menggunakan fungsi skalar bawaan Amazon Redshift. Untuk informasi selengkapnya, lihat Referensi fungsi SQL.

Amazon Redshift Python UDF memiliki batasan berikut:

- Python UDF tidak dapat mengakses jaringan atau membaca atau menulis ke sistem file.
- Ukuran total pustaka Python yang diinstal pengguna tidak dapat melebihi 100 MB.
- Amazon Redshift hanya dapat menjalankan satu UDF Python pada satu waktu untuk cluster yang disediakan menggunakan manajemen beban kerja otomatis (WLM) dan untuk kelompok kerja tanpa server. Jika Anda mencoba menjalankan lebih dari satu UDF secara bersamaan, Amazon Redshift mengantri UDF Python yang tersisa untuk dijalankan dalam antrian manajemen beban kerja. SQL UDF tidak memiliki batas konkurensi saat menggunakan WLM otomatis.
- Saat menggunakan WLM manual untuk cluster yang disediakan, jumlah UDF Python yang dapat berjalan secara bersamaan per cluster dibatasi hingga seperempat dari total level konkurensi cluster. Misalnya, cluster yang disediakan dengan konkurensi 15 dapat menjalankan maksimal tiga UDF Python bersamaan.
- Saat menggunakan Python UDF, Amazon Redshift tidak mendukung tipe data SUPER dan HLLSKETCH.

Kesalahan pencatatan dan peringatan di Python UDF

Anda dapat menggunakan modul logging Python untuk membuat pesan kesalahan dan peringatan yang ditentukan pengguna di UDF Anda. Setelah eksekusi kueri, Anda dapat menanyakan SVL_UDF_LOG tampilan sistem untuk mengambil pesan yang dicatat.

> **Note**
>
> Pencatatan UDF mengkonsumsi sumber daya cluster dan dapat mempengaruhi kinerja sistem. Sebaiknya terapkan logging hanya untuk pengembangan dan pemecahan masalah.

Selama eksekusi query, log handler menulis pesan ke tampilan sistem SVL_UDF_LOG, bersama dengan nama fungsi, node, dan irisan yang sesuai. Penangan log menulis satu baris ke SVL_UDF_LOG per pesan, per irisan. Pesan dipotong menjadi 4096 byte. Log UDF dibatasi hingga

500 baris per irisan. Ketika log penuh, penangan log membuang pesan lama dan menambahkan pesan peringatan ke SVL_UDF_LOG.

> **Note**
>
> Penangan log Amazon Redshift UDF lolos dari karakter newlines (\n), pipe (|), dan backslash () dengan karakter backslash (\). \

Secara default, level log UDF diatur ke PERINGATAN. Pesan dengan tingkat log WARNING, ERROR, dan CRITICAL dicatat. Pesan dengan tingkat keparahan yang lebih rendah INFO, DEBUG, dan NOTSET diabaikan. Untuk mengatur level log UDF, gunakan metode logger Python. Misalnya, berikut ini menetapkan tingkat log ke INFO.

```
logger.setLevel(logging.INFO)
```

Untuk informasi selengkapnya tentang menggunakan modul logging Python, lihat Fasilitas logging untuk Python dalam dokumentasi Python.

Contoh berikut membuat fungsi bernama f_pyerror yang mengimpor modul logging Python, membuat instance logger, dan mencatat kesalahan.

```
CREATE OR REPLACE FUNCTION f_pyerror()
RETURNS INTEGER
VOLATILE AS
$$
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
logger.info('Your info message here')
return 0
$$ language plpythonu;
```

Contoh berikut menanyakan SVL_UDF_LOG untuk melihat pesan yang dicatat dalam contoh sebelumnya.

```
select funcname, node, slice, trim(message) as message
from svl_udf_log;
```

```
  funcname  | query | node | slice |   message
------------+-------+------+-------+-----------------
  f_pyerror | 12345 |     1|     1 | Your info message here
```

UDF Lambda Skalar

Amazon Redshift dapat menggunakan fungsi khusus yang didefinisikan AWS Lambda sebagai bagian dari kueri SQL. Anda dapat menulis UDF Lambda skalar dalam bahasa pemrograman apa pun yang didukung oleh Lambda, seperti Java, Go,,, C# PowerShell Node.js, Python, dan Ruby. Atau Anda dapat menggunakan runtime khusus.

Per BUAT FUNGSI EKSTERNAL intah membuat parameter berikut:

- (Opsional) Daftar argumen dengan tipe data.
- Satu tipe data pengembalian.
- Salah satu nama fungsi dari fungsi eksternal yang dipanggil oleh Amazon Redshift.
- Satu peran IAM yang diotorisasi untuk diambil oleh cluster Amazon Redshift dan dipanggil ke Lambda.
- Satu nama fungsi Lambda yang dipanggil Lambda UDF.

Untuk informasi tentang MEMBUAT FUNGSI EKSTERNAL, lihatBUAT FUNGSI EKSTERNAL.

Jenis data input dan return untuk Lambda UDF dapat berupa salah satu jenis berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- CHAR
- VARCHAR
- BOOLEAN
- DATE
- TIMESTAMP

Alias untuk jenis ini juga valid. Untuk daftar lengkap tipe data dan aliasnya, lihatJenis Data.

Amazon Redshift memastikan bahwa fungsi eksternal dapat mengirim dan menerima argumen dan hasil batch.

UDF Lambda didefinisikan dan dikelola di Lambda, dan Anda dapat mengontrol hak akses untuk memanggil UDF ini di Amazon Redshift. Anda dapat memanggil beberapa fungsi Lambda dalam kueri yang sama atau memanggil fungsi yang sama beberapa kali.

Gunakan Lambda UDF dalam klausa apa pun dari pernyataan SQL di mana fungsi skalar didukung. Anda juga dapat menggunakan Lambda UDF dalam pernyataan SQL apa pun seperti SELECT, UPDATE, INSERT, atau DELETE.

> **Note**
>
> Menggunakan UDF Lambda dapat dikenakan biaya tambahan dari layanan Lambda. Apakah itu terjadi tergantung pada faktor-faktor seperti jumlah permintaan Lambda (pemanggilan UDF) dan total durasi eksekusi program Lambda. Namun, tidak ada biaya tambahan untuk menggunakan UDF Lambda di Amazon Redshift. Untuk informasi tentang harga AWS Lambda, lihat AWS Lambda Harga.
>
> Jumlah permintaan Lambda bervariasi tergantung pada klausa pernyataan SQL tertentu di mana Lambda UDF digunakan. Misalnya, misalkan fungsi digunakan dalam klausa WHERE seperti berikut ini.
>
> ```
> SELECT a, b FROM t1 WHERE lambda_multiply(a, b) = 64; SELECT a, b
> FROM t1 WHERE a*b = lambda_multiply(2, 32)
> ```
>
> Dalam hal ini, Amazon Redshift memanggil pernyataan SELECT pertama untuk masing-masing dan memanggil pernyataan SELECT kedua hanya sekali.
>
> Namun, menggunakan UDF di bagian proyeksi kueri mungkin hanya memanggil fungsi Lambda sekali untuk setiap baris yang memenuhi syarat atau agregat dalam kumpulan hasil.

Keamanan dan izin UDF

Untuk membuat Lambda UDF, pastikan Anda memiliki izin untuk penggunaan pada LANGUAGE EXFUNC. Anda harus secara eksplisit memberikan USAGE ON LANGUAGE EXFUNC atau mencabut USE ON LANGUAGE EXFUNC kepada pengguna, grup, atau publik tertentu.

Contoh berikut memberikan penggunaan pada EXFUNC ke PUBLIC.

```
grant usage on language exfunc to PUBLIC;
```

Contoh berikut mencabut penggunaan pada exfunc dari PUBLIC dan kemudian memberikan penggunaan ke grup pengguna lambda_udf_devs.

```
revoke usage on language exfunc from PUBLIC;
grant usage on language exfunc to group lambda_udf_devs;
```

Untuk menjalankan Lambda UDF, pastikan Anda memiliki izin untuk setiap fungsi yang dipanggil. Secara default, izin untuk menjalankan UDF Lambda baru diberikan kepada PUBLIC. Untuk membatasi penggunaan, cabut izin ini dari PUBLIK untuk fungsi tersebut. Kemudian, berikan hak istimewa kepada pengguna atau grup tertentu.

Contoh berikut mencabut eksekusi pada fungsi exfunc_sum dari PUBLIC. Kemudian, memberikan penggunaan ke grup pengguna lambda_udf_devs.

```
revoke execute on function exfunc_sum(int, int) from PUBLIC;
grant execute on function exfunc_sum(int, int) to group lambda_udf_devs;
```

Pengguna super memiliki semua hak istimewa secara default.

Untuk informasi lebih lanjut tentang pemberian dan pencabutan hak istimewa, lihat HIBAH danMENCABUT.

Mengkonfigurasi parameter otorisasi untuk Lambda UDF

Perintah CREATE EXTERNAL FUNCTION memerlukan otorisasi untuk memanggil fungsi Lambda di. AWS Lambda Untuk memulai otorisasi, tentukan peran AWS Identity and Access Management (IAM) saat Anda menjalankan perintah CREATE EXTERNAL FUNCTION. Untuk informasi selengkapnya tentang peran IAM, lihat Peran IAM dalam Panduan Pengguna IAM.

Jika ada peran IAM dengan izin untuk memanggil fungsi Lambda yang dilampirkan ke cluster Anda, maka Anda dapat mengganti peran Amazon Resource Name (ARN) Anda di parameter IAM_ROLE untuk perintah tersebut. Bagian berikut menjelaskan langkah-langkah untuk menggunakan peran IAM dalam perintah CREATE EXTERNAL FUNCTION.

Membuat peran IAM untuk Lambda

Peran IAM memerlukan izin untuk memanggil fungsi Lambda. Saat membuat peran IAM, berikan izin dengan salah satu cara berikut:

- Lampirkan `AWSLambdaRole` kebijakan pada halaman kebijakan lamp irkan izin saat membuat peran IAM. `AWSLambdaRole`Kebijakan memberikan izin untuk memanggil fungsi Lambda yang merupakan persyaratan minimal. Untuk informasi selengkapnya dan kebijakan lainnya, lihat kebijakan Identity-based IAM untuk AWS Lambda di Panduan Peng AWS Lambda embang.
- Buat kebijakan kustom Anda sendiri untuk dilampirkan ke peran IAM Anda dengan `lambda:InvokeFunction` izin dari semua sumber daya atau fungsi Lambda tertentu dengan ARN fungsi tersebut. Untuk informasi selengkapnya tentang cara membuat kebijakan, lihat Membuat kebijakan IAM di Panduan Pengguna IAM.

  Kebijakan contoh berikut memungkinkan memanggil Lambda pada fungsi Lambda tertentu.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Invoke",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-west-2:123456789012:function:my-
function"
        }
    ]
}
```

  Untuk informasi selengkapnya tentang sumber daya untuk fungsi Lambda, lihat Sumber daya dan ketentuan untuk tindakan Lambda di Referensi API IAM.

  Setelah membuat kebijakan kustom dengan izin yang diperlukan, Anda dapat melampirkan kebijakan ke peran IAM di halaman kebijakan Lampirkan izin saat membuat peran IAM.

Untuk langkah-langkah membuat peran IAM, lihat Mengotor isasi Amazon Redshift untuk mengakses AWS layanan lain atas nama Anda di Panduan Manajemen Amazon Redshift.

Jika Anda tidak ingin membuat peran IAM baru, Anda dapat menambahkan izin yang disebutkan sebelumnya ke peran IAM yang sudah ada.

Mengaitkan peran IAM dengan cluster

Lampirkan peran IAM ke cluster Anda. Anda dapat menambahkan peran ke cluster atau melihat peran yang terkait dengan cluster menggunakan Amazon Redshift Management Console, CLI, atau API. Untuk informasi selengkapnya, lihat Mengaitkan Peran IAM Dengan Cluster di Panduan Manajemen Amazon Redshift.

Termasuk peran IAM dalam perintah

Sertakan peran IAM ARN dalam perintah CREATE EXTERNAL FUNCTION. Saat Anda membuat peran IAM, IAM mengembalikan Nama Sumber Daya Amazon (ARN) untuk peran tersebut. Untuk menentukan peran IAM, berikan peran ARN dengan `IAM_ROLE` parameter. Berikut ini menunjukkan sintaks untuk `IAM_ROLE` parameter.

```
IAM_ROLE 'arn:aws:iam::aws-account-id:role/role-name'
```

Untuk memanggil fungsi Lambda yang berada di akun lain dalam Wilayah yang sama, lihat Menghubungkan peran IAM di Amazon Redshift.

Menggunakan antarmuka JSON antara Amazon Redshift dan AWS Lambda

Amazon Redshift menggunakan antarmuka umum untuk semua fungsi Lambda yang dikomunikasikan Amazon Redshift.

Tabel berikut menunjukkan daftar bidang input yang berfungsi Lambda yang ditunjuk yang dapat Anda harapkan untuk muatan JSON.

Nama bidang	Deskripsi	Rentang nilai
id permintaan	Pengidentifikasi unik universal (UUID) yang secara unik mengidentifikasi setiap permintaan pemanggilan.	UUID yang valid.
klaster	Nama Sumber Daya Amazon (ARN) lengkap dari cluster.	Cluster ARN yang valid.

Nama bidang	Deskripsi	Rentang nilai
user	Nama pengguna yang melakukan panggilan.	Nama pengguna yang valid.
basis data	Nama database tempat kueri berjalan.	Nama database yang valid.
fungsi_eksternal	Nama yang sepenuhnya memenuhi syarat dari fungsi eksternal yang membuat panggilan.	Nama fungsi yang sepenuhnya memenuhi syarat yang valid.
Query_id	ID kueri dari kueri yang membuat panggilan.	ID kueri yang valid.
nom_record	Jumlah argumen dalam payload.	Nilai 1 - 2^64.
argumen	Payload data dalam format yang ditentukan.	Data dalam format array harus berupa array JSON. Setiap elemen adalah catatan yang merupakan array jika jumlah argumen lebih besar dari 1. Dengan menggunakan array, Amazon Redshift mempertahankan urutan catatan dalam payload.

Urutan array JSON menentukan urutan pemrosesan batch. Fungsi Lambda harus memproses argumen secara berulang dan menghasilkan jumlah rekaman yang tepat. Berikut ini adalah contoh payload.

```
{
  "request_id" : "23FF1F97-F28A-44AA-AB67-266ED976BF40",
  "cluster" : "arn:aws:redshift:xxxx",
  "user" : "adminuser",
  "database" : "db1",
  "external_function": "public.foo",
  "query_id" : 5678234,
  "num_records" : 4,
  "arguments" : [
```

```
    [ 1, 2 ],
    [ 3, null],
    null,
    [ 4, 6]
  ]
}
```

Output kembali dari fungsi Lambda berisi bidang berikut.

Nama bidang	Deskripsi	Rentang nilai
keberhasilan	Indikasi keberhasilan atau kegagalan fungsi tersebut.	Nilai `"true"` atau`"false"`.
error_msg	Pesan kesalahan jika nilai keberhasilan adalah `"false"` (jika fungsi gagal); jika tidak, bidang ini diabaikan.	Pesan yang valid.
nom_record	Jumlah catatan dalam payload.	Nilai 1 - 2^64.
hasil	Hasil panggilan dalam format yang ditentukan.	N/A

Berikut ini adalah contoh output fungsi Lambda.

```
{
  "success": true,   // true indicates the call succeeded
  "error_msg" : "my function isn't working",  // shall only exist when success != true
  "num_records": 4,      // number of records in this payload
  "results" : [
    1,
    4,
    null,
    7
  ]
```

```
}
```

Saat Anda memanggil fungsi Lambda dari kueri SQL, Amazon Redshift memastikan keamanan koneksi dengan pertimbangan berikut:

- BERIKAN dan CABUT izin. Untuk informasi selengkapnya tentang keamanan dan izin UDF, lihatKeamanan dan izin UDF.
- Amazon Redshift hanya mengirimkan kumpulan data minimum ke fungsi Lambda yang ditunjuk.
- Amazon Redshift hanya memanggil fungsi Lambda yang ditunjuk dengan peran IAM yang ditunjuk.

Contoh kasus penggunaan untuk fungsi yang ditentukan pengguna (UDF)

> **Note**
>
> Mulai 1 November 2025, Amazon Redshift tidak akan lagi mendukung pembuatan UDF Python baru. UDF Python yang ada akan terus berfungsi hingga 30 Juni 2026. Mulai 1 Juli 2026, Amazon Redshift tidak akan lagi mendukung Python UDF. Kami menyarankan Anda memigrasikan UDF Python yang ada ke UDF Lambda sebelum 1 November 2025. Untuk informasi tentang membuat dan menggunakan Lambda UDF, lihat. UDF Lambda Skalar Untuk informasi tentang mengonversi UDF Python yang ada ke UDF Lambda, lihat posting blog. https://aws.amazon.com/blogs/big-data/amazon-redshift-python-user-defined-functions-will-reach-end-of-support-after-june-30-2026/

Anda dapat menggunakan fungsi yang ditentukan pengguna untuk memecahkan masalah bisnis dengan mengintegrasikan Amazon Redshift dengan komponen lain. Berikut ini adalah beberapa contoh bagaimana orang lain telah menggunakan UDF untuk kasus penggunaan mereka:

- Mengakses komponen eksternal menggunakan Amazon Redshift Lambda UDF — menjelaskan cara kerja UDF Amazon Redshift Lambda dan berjalan melalui pembuatan UDF Lambda.
- Terjemahkan dan analisis teks menggunakan fungsi SQL dengan Amazon Redshift, Amazon Translate, dan Amazon Compredekah — menyediakan UDF Amazon Redshift Lambda bawaan yang dapat Anda instal dengan beberapa klik untuk menerjemahkan, menyunting, dan menganalisis bidang teks.

- Akses Layanan Lokasi Amazon dari Amazon Redshift — menjelaskan cara menggunakan Amazon Redshift Lambda UDF untuk diintegrasikan dengan Layanan Lokasi Amazon.
- Tokenisasi Data dengan Amazon Redshift dan Protegrity — menjelaskan cara mengintegrasikan Amazon Redshift Lambda UDF dengan produk Protegrity Serverless.
- Amazon Redshift UDF — kumpulan UDF Amazon Redshift SQL, Lambda, dan Python.

Membuat prosedur tersimpan di Amazon Redshift

Topik ini menjelaskan cara membuat dan menggunakan prosedur tersimpan di Amazon Redshift. Prosedur tersimpan adalah kumpulan pernyataan SQL yang dapat digunakan beberapa program.

Anda dapat menentukan prosedur tersimpan Amazon Redshift menggunakan bahasa prosedural PostgreSQL PL/pgSQL untuk melakukan serangkaian kueri SQL dan operasi logis. Prosedur ini disimpan dalam database dan tersedia untuk setiap pengguna dengan hak database yang memadai.

Tidak seperti fungsi yang ditentukan pengguna (UDF), prosedur tersimpan dapat menggabungkan bahasa definisi data (DDL) dan bahasa manipulasi data (DML) selain kueri SELECT. Prosedur tersimpan tidak perlu mengembalikan nilai. Anda dapat menggunakan bahasa prosedural, termasuk ekspresi perulangan dan kondisional, untuk mengontrol aliran logis.

Untuk detail tentang perintah SQL untuk membuat dan mengelola prosedur tersimpan, lihat topik perintah berikut:

- BUAT PROSEDUR
- MENGUBAH PROSEDUR
- PROSEDUR DROP
- TAMPILKAN PROSEDUR
- PANGGILAN
- HIBAH
- MENCABUT
- UBAH HAK ISTIMEWA DEFAULT

Topik

- Ikhtisar prosedur tersimpan di Amazon Redshift
- PL/pgSQL referensi bahasa

Ikhtisar prosedur tersimpan di Amazon Redshift

Topik ini menjelaskan detail tentang tujuan dan penggunaan prosedur yang disimpan.

Prosedur tersimpan biasanya digunakan untuk merangkum logika untuk transformasi data, validasi data, dan logika khusus bisnis. Dengan menggabungkan beberapa langkah SQL ke dalam prosedur tersimpan, Anda dapat mengurangi perjalanan pulang pergi antara aplikasi Anda dan database.

Untuk kontrol akses halus, Anda dapat membuat prosedur tersimpan untuk melakukan fungsi tanpa memberikan akses pengguna ke tabel yang mendasarinya. Misalnya, hanya pemilik atau superuser yang dapat memotong tabel, dan pengguna memerlukan hak menulis untuk memasukkan data ke dalam tabel. Alih-alih memberikan hak istimewa pengguna pada tabel yang mendasarinya, Anda dapat membuat prosedur tersimpan yang melakukan tugas. Anda kemudian memberikan hak pengguna untuk menjalankan prosedur tersimpan.

Prosedur tersimpan dengan atribut keamanan DEFINER berjalan dengan hak istimewa pemilik prosedur tersimpan. Secara default, prosedur tersimpan memiliki keamanan INVOKER, yang berarti prosedur menggunakan hak istimewa pengguna yang memanggil prosedur.

Untuk membuat prosedur tersimpan, gunakan BUAT PROSEDUR perintah. Untuk menjalankan prosedur, gunakan PANGGILAN perintah. Contoh berikut nanti di bagian ini.

Note

Beberapa klien mungkin menampilkan kesalahan berikut saat membuat prosedur tersimpan Amazon Redshift.

```
ERROR: 42601: [Amazon](500310) unterminated dollar-quoted string at or near "$$
```

Kesalahan ini terjadi karena ketidakmampuan klien untuk mengurai pernyataan CREATE PROCEDURE dengan benar dengan tanda titik koma yang membatasi pernyataan dan dengan kutipan tanda dolar ($). Ini menghasilkan hanya sebagian dari pernyataan yang dikirim ke server Amazon Redshift. Anda sering dapat mengatasi kesalahan ini dengan menggunakan `Execute selected` opsi `Run as batch` atau klien.
Misalnya, saat menggunakan klien Aginity, gunakan `Run entire script as batch` opsi. Saat Anda menggunakan SQL Workbench/J, kami merekomendasikan versi 124. Saat Anda menggunakan SQL Workbench/J versi 125, pertimbangkan untuk menentukan pembatas alternatif sebagai solusi.
CREATE PROCEDURE berisi pernyataan SQL yang dibatasi dengan titik koma (;). Mendefinisikan pembatas alternatif seperti garis miring (/) dan menempatkannya di akhir pernyataan CREATE PROCEDURE mengirimkan pernyataan ke server Amazon Redshift untuk diproses. Berikut adalah contohnya.

```
CREATE OR REPLACE PROCEDURE test()
AS $$
BEGIN
  SELECT 1 a;
END;
$$
LANGUAGE plpgsql
;
/
```

Anda dapat menggunakan klien dengan dukungan untuk mengurai pernyataan CREATE PROCEDURE, seperti editor ku eri di konsol Amazon Redshift atau. TablePlus

Topik

- Penamaan prosedur yang disimpan
- Keamanan dan hak istimewa untuk prosedur tersimpan
- Mengembalikan hasil set dari prosedur tersimpan
- Mengelola transaksi
- Kesalahan menjebak
- Pencatatan prosedur tersimpan
- Batasan prosedur yang disimpan

Contoh berikut menunjukkan prosedur tanpa argumen output. Secara default, argumen adalah argumen input (IN).

```
CREATE OR REPLACE PROCEDURE test_sp1(f1 int, f2 varchar)
AS $$
BEGIN
  RAISE INFO 'f1 = %, f2 = %', f1, f2;
END;
$$ LANGUAGE plpgsql;

call test_sp1(5, 'abc');
INFO: f1 = 5, f2 = abc
CALL
```

> **Note**
>
> Saat Anda menulis prosedur tersimpan, kami merekomendasikan praktik terbaik untuk mengamankan nilai-nilai sensitif:
> Jangan hardcode informasi sensitif apa pun dalam logika prosedur yang disimpan. Misalnya, jangan menetapkan kata sandi pengguna dalam pernyataan CREATE USER di badan prosedur yang disimpan. Ini menimbulkan risiko keamanan, karena nilai hardcode dapat dicatat sebagai metadata skema dalam tabel katalog. Sebagai gantinya, berikan nilai sensitif, seperti kata sandi, sebagai argumen ke prosedur yang disimpan, melalui parameter.
> Untuk informasi lebih lanjut tentang prosedur yang disimpan, lihat BUAT PROSEDUR danMembuat prosedur tersimpan di Amazon Redshift. Untuk informasi selengkapnya tentang tabel katalog, lihatTabel katalog sistem.

Contoh berikut menunjukkan prosedur dengan argumen keluaran. Argumen adalah input (IN), input dan output (INOUT), dan output (OUT).

```
CREATE OR REPLACE PROCEDURE test_sp2(f1 IN int, f2 INOUT varchar(256), out_var OUT
 varchar(256))
AS $$
DECLARE
  loop_var int;
BEGIN
  IF f1 is null OR f2 is null THEN
    RAISE EXCEPTION 'input cannot be null';
  END IF;
  DROP TABLE if exists my_etl;
  CREATE TEMP TABLE my_etl(a int, b varchar);
    FOR loop_var IN 1..f1 LOOP
        insert into my_etl values (loop_var, f2);
        f2 := f2 || '+' || f2;
    END LOOP;
  SELECT INTO out_var count(*) from my_etl;
END;
$$ LANGUAGE plpgsql;


call test_sp2(2,'2019');

         f2          | column2
```

```
---------------------+---------
 2019+2019+2019+2019 | 2
(1 row)
```

Penamaan prosedur yang disimpan

Topik ini menjelaskan detail tentang nama prosedur yang disimpan.

Jika Anda menentukan prosedur dengan nama yang sama dan jenis data atau tanda tangan argumen input yang berbeda, Anda membuat prosedur baru. Akibatnya, nama prosedur kelebihan beban. Untuk informasi selengkapnya, lihat Membebani nama prosedur. Amazon Redshift tidak mengaktifkan kelebihan beban prosedur berdasarkan argumen keluaran. Anda tidak dapat memiliki dua prosedur dengan nama yang sama dan tipe data argumen input tetapi jenis argumen keluaran yang berbeda.

Pemilik atau superuser dapat mengganti badan prosedur yang disimpan dengan yang baru dengan tanda tangan yang sama. Untuk mengubah tanda tangan atau jenis pengembalian dari prosedur yang disimpan, lepaskan prosedur yang disimpan dan buat ulang. Untuk informasi selengkapnya, lihat PROSEDUR DROP dan BUAT PROSEDUR.

Anda dapat menghindari potensi konflik dan hasil yang tidak terduga dengan mempertimbangkan konvensi penamaan untuk prosedur tersimpan sebelum menerapkannya. Karena Anda dapat membebani nama prosedur, nama tersebut dapat bertabrakan dengan nama prosedur Amazon Redshift yang ada dan yang akan datang.

Membebani nama prosedur

Prosedur diidentifikasi dengan nama dan tanda tangannya, yang merupakan jumlah argumen input dan tipe data argumen. Dua prosedur dalam skema yang sama dapat memiliki nama yang sama jika mereka memiliki tanda tangan yang berbeda. Dengan kata lain, Anda dapat membebani nama prosedur.

Ketika Anda menjalankan prosedur, mesin kueri menentukan prosedur mana yang akan dipanggil berdasarkan jumlah argumen yang Anda berikan dan tipe data argumen. Anda dapat menggunakan kelebihan beban untuk mensimulasikan prosedur dengan jumlah variabel argumen, hingga batas yang diizinkan oleh perintah CREATE PROCEDURE. Untuk informasi selengkapnya, lihat BUAT PROSEDUR.

Mencegah konflik penamaan

Sebaiknya beri nama semua prosedur menggunakan awalan`sp_`. Amazon Redshift menyimpan `sp_` awalan khusus untuk prosedur yang disimpan. Dengan mengawali nama prosedur Anda dengan`sp_`, Anda memastikan bahwa nama prosedur Anda tidak akan bertentangan dengan nama prosedur Amazon Redshift yang ada atau yang akan datang.

Keamanan dan hak istimewa untuk prosedur tersimpan

Topik ini menjelaskan kredentif database yang diperlukan untuk membuat dan menjalankan prosedur tersimpan.

Secara default, semua pengguna memiliki hak istimewa untuk membuat prosedur. Untuk membuat prosedur, Anda harus memiliki hak istimewa USAGE pada bahasa PL/pgSQL, yang diberikan kepada PUBLIC secara default. Hanya pengguna super dan pemilik yang memiliki hak istimewa untuk memanggil prosedur secara default. Pengguna super dapat menjalankan REVOKE US PL/pgSQL AGE dari pengguna jika mereka ingin mencegah pengguna membuat prosedur tersimpan.

Untuk memanggil prosedur, Anda harus diberikan hak istimewa EXECUTE pada prosedur tersebut. Secara default, hak istimewa EXECUTE untuk prosedur baru diberikan kepada pemilik prosedur dan pengguna super. Untuk informasi selengkapnya, lihat HIBAH.

Pengguna yang membuat prosedur adalah pemilik secara default. Pemilik memiliki hak istimewa CREATE, DROP, dan EXECUTE pada prosedur secara default. Pengguna super memiliki semua hak istimewa.

Atribut SECURITY mengontrol hak istimewa prosedur untuk mengakses objek database. Saat Anda membuat prosedur tersimpan, Anda dapat mengatur atribut SECURITY menjadi DEFINER atau INVOKER. Atribut ini menentukan hak istimewa mana yang digunakan saat mengeksekusi pernyataan di badan prosedur yang disimpan. Jika Anda menentukan SECURITY INVOKER, prosedur menggunakan hak istimewa pengguna yang memanggil prosedur. Jika Anda menentukan SECURITY DEFINER, prosedur menggunakan hak istimewa pemilik prosedur. INVOKER adalah default.

Karena prosedur SECURITY DEFINER berjalan dengan hak istimewa pengguna yang memilikinya, Anda harus memastikan bahwa prosedur tidak dapat disalahgunakan. Untuk memastikan bahwa prosedur SECURITY DEFINER tidak dapat disalahgunakan, lakukan hal berikut:

- Berikan EXECUTE pada prosedur SECURITY DEFINER kepada pengguna tertentu, dan bukan ke PUBLIK.

- Kualifikasi semua objek database yang harus diakses prosedur dengan nama skema. Misalnya, gunakan al `myschema.mytable` ih-alih hanya`mytable`.
- Jika Anda tidak dapat memenuhi syarat nama objek dengan skema, atur `search_path` saat membuat prosedur dengan menggunakan opsi SET. Setel `search_path` untuk mengecualikan skema apa pun yang dapat ditulis oleh pengguna yang tidak tepercaya. Pendekatan ini mencegah pemanggil prosedur ini membuat objek (misalnya, tabel atau tampilan) yang menutupi objek yang dimaksudkan untuk digunakan oleh prosedur. Untuk informasi selengkapnya tentang opsi SET, lihatBUAT PROSEDUR.

Contoh berikut ditetapkan `search_path` `admin` untuk memastikan bahwa `user_creds` tabel diakses dari `admin` skema dan bukan dari publik atau skema lain di pemanggil. `search_path`

```
CREATE OR REPLACE PROCEDURE sp_get_credentials(userid int, o_creds OUT varchar)
AS $$
BEGIN
  SELECT creds INTO o_creds
  FROM user_creds
  WHERE user_id = $1;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER
-- Set a secure search_path
SET search_path = admin;
```

Mengembalikan hasil set dari prosedur tersimpan

Topik ini menjelaskan bagaimana prosedur tersimpan mengembalikan data.

Anda dapat mengembalikan set hasil menggunakan kursor atau tabel temp.

Mengembalikan kursor

Untuk mengembalikan kursor, buat prosedur dengan argumen INOUT yang ditentukan dengan tipe `refcursor` data. Saat Anda memanggil prosedur, beri nama kursor. Kemudian Anda dapat mengambil hasil dari kursor dengan nama.

Contoh berikut membuat prosedur bernama `get_result_set` dengan argumen INOUT bernama `rs_out` menggunakan tipe `refcursor` data. Prosedur membuka kursor menggunakan pernyataan SELECT.

```
CREATE OR REPLACE PROCEDURE get_result_set (param IN integer, rs_out INOUT refcursor)
AS $$
BEGIN
  OPEN rs_out FOR SELECT * FROM fact_tbl where id >= param;
END;
$$ LANGUAGE plpgsql;
```

Perintah CALL berikut membuka kursor dengan namamycursor. Gunakan kursor hanya dalam transaksi.

```
BEGIN;
CALL get_result_set(1, 'mycursor');
```

Setelah kursor dibuka, Anda dapat mengambil dari kursor, seperti yang ditunjukkan contoh berikut.

```
FETCH ALL FROM mycursor;

    id | secondary_id | name
-------+--------------+---------
     1 |            1 | Joe
     1 |            2 | Ed
     2 |            1 | Mary
     1 |            3 | Mike
(4 rows)
```

Pada akhirnya, transaksi dilakukan atau dibatalkan.

```
COMMIT;
```

Kursor yang dikembalikan oleh prosedur tersimpan tunduk pada batasan dan pertimbangan kinerja yang sama seperti yang dijelaskan dalam DECLARE CURSOR. Untuk informasi selengkapnya, lihat Kendala kursor.

Contoh berikut menunjukkan pemanggilan prosedur ter `get_result_set` simpan menggunakan tipe `refcursor` data dari JDBC. Literal `'mycursor'` (nama kursor) diteruskan ke`prepareStatement`. Kemudian hasilnya diambil dari. `ResultSet`

```
static void refcursor_example(Connection conn) throws SQLException {
    conn.setAutoCommit(false);
```

```
    PreparedStatement proc = conn.prepareStatement("CALL get_result_set(1,
 'mycursor')");
    proc.execute();
    ResultSet rs = statement.executeQuery("fetch all from mycursor");
    while (rs.next()) {
      int n = rs.getInt(1);
      System.out.println("n " + n);
    }
```

Menggunakan tabel temp

Untuk mengembalikan hasil, Anda dapat mengembalikan pegangan ke tabel temp yang berisi baris hasil. Klien dapat memberikan nama sebagai parameter untuk prosedur yang disimpan. Di dalam prosedur tersimpan, SQL dinamis dapat digunakan untuk beroperasi pada tabel temp. Bagian berikut menunjukkan satu contoh.

```
CREATE PROCEDURE get_result_set(param IN integer, tmp_name INOUT varchar(256)) as $$
DECLARE
  row record;
BEGIN
  EXECUTE 'drop table if exists ' || tmp_name;
  EXECUTE 'create temp table ' || tmp_name || ' as select * from fact_tbl where id <= '
 || param;
END;
$$ LANGUAGE plpgsql;

CALL get_result_set(2, 'myresult');
 tmp_name
-----------
 myresult
(1 row)

SELECT * from myresult;
 id | secondary_id | name
----+--------------+------
  1 |            1 | Joe
  2 |            1 | Mary
  1 |            2 | Ed
  1 |            3 | Mike
(4 rows)
```

Mengelola transaksi

Anda dapat membuat prosedur tersimpan dengan perilaku manajemen transaksi default atau perilaku nonatomik.

Mode default prosedur tersimpan manajemen transaksi

Perilaku komit otomatis mode transaksi default menyebabkan setiap perintah SQL yang berjalan secara terpisah untuk melakukan komit secara individual. Panggilan ke prosedur tersimpan diperlakukan sebagai perintah SQL tunggal. Pernyataan SQL di dalam prosedur berperilaku seolah-olah mereka berada dalam blok transaksi yang secara implisit dimulai ketika panggilan dimulai dan berakhir ketika panggilan selesai. Panggilan bersarang ke prosedur lain diperlakukan seperti pernyataan SQL lainnya dan beroperasi dalam konteks transaksi yang sama dengan pemanggil. Untuk informasi selengkapnya tentang perilaku komit otomatis, lihatTingkat isolasi di Amazon Redshift.

Namun, misalkan Anda memanggil prosedur tersimpan dari dalam blok transaksi yang ditentukan pengguna (ditentukan oleh BEGIN... COMMIT). Dalam hal ini, semua pernyataan dalam prosedur tersimpan dijalankan dalam konteks transaksi yang ditentukan pengguna. Prosedur tidak berkomitmen secara implisit saat keluar. Penelepon mengontrol prosedur commit atau rollback.

Jika ada kesalahan yang ditemui saat menjalankan prosedur tersimpan, semua perubahan yang dilakukan dalam transaksi saat ini akan dibatalkan.

Anda dapat menggunakan pernyataan kontrol transaksi berikut dalam prosedur tersimpan:

- COMMIT — melakukan semua pekerjaan yang dilakukan dalam transaksi saat ini dan secara implisit memulai transaksi baru. Untuk informasi selengkapnya, lihat COMMIT.
- ROLLBACK — mengembalikan pekerjaan yang dilakukan dalam transaksi saat ini dan secara implisit memulai transaksi baru. Untuk informasi selengkapnya, lihat ROLLBACK.

TRUNCATE adalah pernyataan lain yang dapat Anda keluarkan dari dalam prosedur tersimpan dan memengaruhi manajemen transaksi. Di Amazon Redshift, TRUNCATE mengeluarkan komit secara implisit. Perilaku ini tetap sama dalam konteks prosedur yang disimpan. Ketika pernyataan TRUNCATE dikeluarkan dari dalam prosedur tersimpan, ia melakukan transaksi saat ini dan memulai yang baru. Untuk informasi selengkapnya, lihat MEMOTONG.

Semua pernyataan yang mengikuti pernyataan COMMIT, ROLLBACK, atau TRUNCATE dijalankan dalam konteks transaksi baru. Mereka melakukannya sampai pernyataan COMMIT, ROLLBACK, atau TRUNCATE ditemui atau prosedur tersimpan keluar.

Saat Anda menggunakan pernyataan COMMIT, ROLLBACK, atau TRUNCATE dari dalam prosedur tersimpan, batasan berikut berlaku:

- Jika prosedur tersimpan dipanggil dari dalam blok transaksi, itu tidak dapat mengeluarkan pernyataan COMMIT, ROLLBACK, atau TRUNCATE. Pembatasan ini berlaku di dalam tubuh prosedur tersimpan sendiri dan dalam panggilan prosedur bersarang apa pun.
- Jika prosedur tersimpan dibuat dengan `SET config` opsi, tidak dapat mengeluarkan pernyataan COMMIT, ROLLBACK, atau TRUNCATE. Pembatasan ini berlaku di dalam tubuh prosedur tersimpan sendiri dan dalam panggilan prosedur bersarang apa pun.
- Setiap kursor yang terbuka (eksplisit atau implisit) ditutup secara otomatis ketika pernyataan COMMIT, ROLLBACK, atau TRUNCATE diproses. Untuk batasan pada kursor eksplisit dan implisit, lihatBatasan prosedur yang disimpan.

Selain itu, Anda tidak dapat menjalankan COMMIT atau ROLLBACK menggunakan SQL dinamis. Namun, Anda dapat menjalankan TRUNCATE menggunakan SQL dinamis. Untuk informasi selengkapnya, lihat SQL Dinamis.

Saat bekerja dengan prosedur tersimpan, pertimbangkan bahwa pernyataan BEGIN dan END hanya untuk pengelompokan. PL/pgSQL Mereka tidak memulai atau mengakhiri transaksi. Untuk informasi selengkapnya, lihat Blokir.

Contoh berikut menunjukkan perilaku transaksi saat memanggil prosedur tersimpan dari dalam blok transaksi eksplisit. Dua pernyataan sisipan yang dikeluarkan dari luar prosedur yang disimpan dan satu dari dalamnya semuanya adalah bagian dari transaksi yang sama (3382). Transaksi dilakukan ketika pengguna mengeluarkan komit eksplisit.

```
CREATE OR REPLACE PROCEDURE sp_insert_table_a(a int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table_a values (a);
END;
$$;

Begin;
  insert into test_table_a values (1);
```

```
  Call sp_insert_table_a(2);
  insert into test_table_a values (3);
Commit;

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  | pid |  type   |               stmt_text
--------+------+-----+---------+----------------------------------------
    103 | 3382 | 599 | UTILITY | Begin;
    103 | 3382 | 599 | QUERY   | insert into test_table_a values (1);
    103 | 3382 | 599 | UTILITY | Call sp_insert_table_a(2);
    103 | 3382 | 599 | QUERY   | INSERT INTO test_table_a values ( $1 )
    103 | 3382 | 599 | QUERY   | insert into test_table_a values (3);
    103 | 3382 | 599 | UTILITY | COMMIT
```

Sebaliknya, ambil contoh ketika pernyataan yang sama dikeluarkan dari luar blok transaksi eksplisit dan sesi memiliki komit otomatis disetel ke ON. Dalam hal ini, setiap pernyataan berjalan dalam transaksinya sendiri.

```
insert into test_table_a values (1);
Call sp_insert_table_a(2);
insert into test_table_a values (3);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  | pid |  type   |
            stmt_text
--------+------+-----+---------
+----------------------------------------------------------------------------------------------------
    103 | 3388 | 599 | QUERY   | insert into test_table_a values (1);
    103 | 3388 | 599 | UTILITY | COMMIT
    103 | 3389 | 599 | UTILITY | Call sp_insert_table_a(2);
    103 | 3389 | 599 | QUERY   | INSERT INTO test_table_a values ( $1 )
    103 | 3389 | 599 | UTILITY | COMMIT
    103 | 3390 | 599 | QUERY   | insert into test_table_a values (3);
    103 | 3390 | 599 | UTILITY | COMMIT
```

Contoh berikut mengeluarkan pernyataan TRUNCATE setelah dimasukkan ke `test_table_a` dalam. Pernyataan TRUNCATE mengeluarkan komit implisit yang melakukan transaksi saat ini (3335) dan memulai yang baru (3336). Transaksi baru dilakukan ketika prosedur keluar.

```
CREATE OR REPLACE PROCEDURE sp_truncate_proc(a int, b int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table_a values (a);
  TRUNCATE test_table_b;
  INSERT INTO test_table_b values (b);
END;
$$;

Call sp_truncate_proc(1,2);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |  pid  |  type   |
                                        stmt_text
--------+------+-------+---------
+-----------------------------------------------------------------------------------------------
    103 | 3335 | 23636 | UTILITY | Call sp_truncate_proc(1,2);
    103 | 3335 | 23636 | QUERY   | INSERT INTO test_table_a values ( $1 )
    103 | 3335 | 23636 | UTILITY | TRUNCATE test_table_b
    103 | 3335 | 23636 | UTILITY | COMMIT
    103 | 3336 | 23636 | QUERY   | INSERT INTO test_table_b values ( $1 )
    103 | 3336 | 23636 | UTILITY | COMMIT
```

Contoh berikut mengeluarkan TRUNCATE dari panggilan bersarang. TRUNCATE melakukan semua pekerjaan yang dilakukan sejauh ini dalam prosedur luar dan dalam dalam transaksi (3344). Ini memulai transaksi baru (3345). Transaksi baru dilakukan ketika prosedur luar keluar.

```
CREATE OR REPLACE PROCEDURE sp_inner(c int, d int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO inner_table values (c);
  TRUNCATE outer_table;
  INSERT INTO inner_table values (d);
END;
$$;
```

```
CREATE OR REPLACE PROCEDURE sp_outer(a int, b int, c int, d int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO outer_table values (a);
  Call sp_inner(c, d);
  INSERT INTO outer_table values (b);
END;
$$;

Call sp_outer(1, 2, 3, 4);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |  pid  |  type   |
                                          stmt_text
--------+------+-------+---------
+-----------------------------------------------------------------------------------------------------
    103 | 3344 | 23636 | UTILITY | Call sp_outer(1, 2, 3, 4);
    103 | 3344 | 23636 | QUERY   | INSERT INTO outer_table values ( $1 )
    103 | 3344 | 23636 | UTILITY | CALL sp_inner( $1 , $2 )
    103 | 3344 | 23636 | QUERY   | INSERT INTO inner_table values ( $1 )
    103 | 3344 | 23636 | UTILITY | TRUNCATE outer_table
    103 | 3344 | 23636 | UTILITY | COMMIT
    103 | 3345 | 23636 | QUERY   | INSERT INTO inner_table values ( $1 )
    103 | 3345 | 23636 | QUERY   | INSERT INTO outer_table values ( $1 )
    103 | 3345 | 23636 | UTILITY | COMMIT
```

Contoh berikut menunjukkan bahwa kursor `cur1` ditutup ketika pernyataan TRUNCATE dikomit.

```
CREATE OR REPLACE PROCEDURE sp_open_cursor_truncate()
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
BEGIN
  open cur1;
  TRUNCATE table test_table_b;
  Loop
    fetch cur1 into rec;
```

```
    raise info '%', rec.c1;
    exit when not found;
  End Loop;
END
$$;

call sp_open_cursor_truncate();
ERROR: cursor "cur1" does not exist
CONTEXT: PL/pgSQL function "sp_open_cursor_truncate" line 8 at fetch
```

Contoh berikut mengeluarkan pernyataan TRUNCATE dan tidak dapat dipanggil dari dalam blok transaksi eksplisit.

```
CREATE OR REPLACE PROCEDURE sp_truncate_atomic() LANGUAGE plpgsql
AS $$
BEGIN
  TRUNCATE test_table_b;
END;
$$;

Begin;
  Call sp_truncate_atomic();
ERROR: TRUNCATE cannot be invoked from a procedure that is executing in an atomic
 context.
HINT: Try calling the procedure as a top-level call i.e. not from within an explicit
 transaction block.
Or, if this procedure (or one of its ancestors in the call chain) was created with SET
 config options, recreate the procedure without them.
CONTEXT: SQL statement "TRUNCATE test_table_b"
PL/pgSQL function "sp_truncate_atomic" line 2 at SQL statement
```

Contoh berikut menunjukkan bahwa pengguna yang bukan superuser atau pemilik tabel dapat mengeluarkan pernyataan TRUNCATE pada tabel. Pengguna melakukan ini menggunakan prosedur `Security Definer` tersimpan. Contoh menunjukkan tindakan berikut:

- Pengguna1 membuat tabel`test_tbl`.
- Pengguna1 membuat prosedur `sp_truncate_test_tbl` tersimpan.
- Pengguna1 memberikan hak `EXECUTE` istimewa pada prosedur tersimpan kepada pengguna2.
- User2 menjalankan prosedur tersimpan untuk memotong tabel. `test_tbl` Contoh menunjukkan jumlah baris sebelum dan sesudah `TRUNCATE` perintah.

```
set session_authorization to user1;
create table test_tbl(id int, name varchar(20));
insert into test_tbl values (1,'john'), (2, 'mary');
CREATE OR REPLACE PROCEDURE sp_truncate_test_tbl() LANGUAGE plpgsql
AS $$
DECLARE
  tbl_rows int;
BEGIN
  select count(*) into tbl_rows from test_tbl;
  RAISE INFO 'RowCount before Truncate: %', tbl_rows;
  TRUNCATE test_tbl;
  select count(*) into tbl_rows from test_tbl;
  RAISE INFO 'RowCount after Truncate: %', tbl_rows;
END;
$$ SECURITY DEFINER;
grant execute on procedure sp_truncate_test_tbl() to user2;
reset session_authorization;


set session_authorization to user2;
call sp_truncate_test_tbl();
INFO:  RowCount before Truncate: 2
INFO:  RowCount after Truncate: 0
CALL
reset session_authorization;
```

Contoh berikut mengeluarkan COMMIT dua kali. COMMIT pertama melakukan semua pekerjaan yang dilakukan dalam transaksi 10363 dan secara implisit memulai transaksi 10364. Transaksi 10364 dilakukan oleh pernyataan COMMIT kedua.

```
CREATE OR REPLACE PROCEDURE sp_commit(a int, b int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table values (a);
  COMMIT;
  INSERT INTO test_table values (b);
  COMMIT;
END;
$$;

call sp_commit(1,2);
```

```
select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;
 userid |  xid  | pid  |  type   |
                               stmt_text
--------+-------+------+---------
+-----------------------------------------------------------------------------------------------
    100 | 10363 | 3089 | UTILITY | call sp_commit(1,2);
    100 | 10363 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
    100 | 10363 | 3089 | UTILITY | COMMIT
    100 | 10364 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
    100 | 10364 | 3089 | UTILITY | COMMIT
```

Contoh berikut mengeluarkan pernyataan ROLLBACK jika `sum_vals` lebih besar dari 2. Pernyataan ROLLBACK pertama mengembalikan semua pekerjaan yang dilakukan dalam transaksi 10377 dan memulai transaksi baru 10378. Transaksi 10378 dilakukan saat prosedur keluar.

```
CREATE OR REPLACE PROCEDURE sp_rollback(a int, b int) LANGUAGE plpgsql
AS $$
DECLARE
  sum_vals int;
BEGIN
  INSERT INTO test_table values (a);
  SELECT sum(c1) into sum_vals from test_table;
  IF sum_vals > 2 THEN
    ROLLBACK;
  END IF;

  INSERT INTO test_table values (b);
END;
$$;

call sp_rollback(1, 2);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

userid |  xid  | pid  |  type   |
                               stmt_text
--------+-------+------+---------
+-----------------------------------------------------------------------------------------------
    100 | 10377 | 3089 | UTILITY | call sp_rollback(1, 2);
```

```
   100 | 10377 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
   100 | 10377 | 3089 | QUERY   | SELECT sum(c1) from test_table
   100 | 10377 | 3089 | QUERY   | Undoing 1 transactions on table 133646 with current
xid 10377 : 10377
   100 | 10378 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
   100 | 10378 | 3089 | UTILITY | COMMIT
```

Manajemen transaksi prosedur tersimpan mode nonatomik

Prosedur tersimpan yang dibuat dalam mode NONATOMIC memiliki perilaku kontrol transaksi yang berbeda dari prosedur yang dibuat dalam mode default. Mirip dengan perilaku komit otomatis perintah SQL di luar prosedur tersimpan, setiap pernyataan SQL di dalam prosedur NONATOMIC berjalan dalam transaksinya sendiri dan melakukan commit secara otomatis. Jika pengguna memulai blok transaksi eksplisit dalam prosedur tersimpan NONATOMIC, maka pernyataan SQL dalam blok tidak secara otomatis melakukan komit. Blok transaksi mengontrol komit atau rollback pernyataan di dalamnya.

Dalam prosedur tersimpan NONATOMIC, Anda dapat membuka blok transaksi eksplisit di dalam prosedur menggunakan pernyataan START TRANSACTION. Namun, jika sudah ada blok transaksi terbuka, pernyataan ini tidak akan melakukan apa-apa karena Amazon Redshift tidak mendukung sub transaksi. Transaksi sebelumnya terus berlanjut.

Saat Anda bekerja dengan loop kursor FOR di dalam prosedur NONATOMIC, pastikan Anda membuka blok transaksi eksplisit sebelum mengulangi hasil kueri. Jika tidak, kursor ditutup ketika pernyataan SQL di dalam loop secara otomatis dikomit.

Beberapa pertimbangan saat menggunakan perilaku mode NONATOMIC adalah sebagai berikut:

- Setiap pernyataan SQL di dalam prosedur tersimpan secara otomatis dikomit jika tidak ada blok transaksi terbuka, dan sesi memiliki komit otomatis yang disetel ke ON.
- Anda dapat mengeluarkan COMMIT/ROLLBACK/TRUNCATE pernyataan untuk mengakhiri transaksi jika prosedur tersimpan dipanggil dari dalam blok transaksi. Ini tidak mungkin dalam mode default.
- Anda dapat mengeluarkan pernyataan START TRANSACTION untuk memulai blok transaksi di dalam prosedur yang disimpan.

Contoh berikut menunjukkan perilaku transaksi saat bekerja dengan prosedur tersimpan NONATOMIC. Sesi untuk semua contoh berikut memiliki komit otomatis disetel ke ON.

Dalam contoh berikut, prosedur tersimpan NONATOMIC memiliki dua pernyataan INSERT. Ketika prosedur dipanggil di luar blok transaksi, setiap pernyataan INSERT dalam prosedur secara otomatis melakukan commit.

```
CREATE TABLE test_table_a(v int);
CREATE TABLE test_table_b(v int);

CREATE OR REPLACE PROCEDURE sp_nonatomic_insert_table_a(a int, b int) NONATOMIC AS
$$
BEGIN
    INSERT INTO test_table_a values (a);
    INSERT INTO test_table_b values (b);
END;
$$
LANGUAGE plpgsql;

Call sp_nonatomic_insert_table_a(1,2);

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |               stmt_text
--------+------+------------+---------+----------------------------------------
      1 | 1792 | 1073807554 | UTILITY | Call sp_nonatomic_insert_table_a(1,2);
      1 | 1792 | 1073807554 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1792 | 1073807554 | UTILITY | COMMIT
      1 | 1793 | 1073807554 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1793 | 1073807554 | UTILITY | COMMIT
(5 rows)
```

Namun, ketika prosedur dipanggil dari dalam blok BEGIN.. COMMIT, semua pernyataan adalah bagian dari transaksi yang sama (xid=1799).

```
Begin;
  INSERT INTO test_table_a values (10);
  Call sp_nonatomic_insert_table_a(20,30);
  INSERT INTO test_table_b values (40);
Commit;

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;
```

```
 userid | xid  |    pid     |  type   |                stmt_text
--------+------+------------+---------+-----------------------------------------
      1 | 1799 | 1073914035 | UTILITY | Begin;
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_a values (10);
      1 | 1799 | 1073914035 | UTILITY | Call sp_nonatomic_insert_table_a(20,30);
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_b values (40);
      1 | 1799 | 1073914035 | UTILITY | COMMIT
(7 rows)
```

Dalam contoh ini, dua pernyataan INSERT berada di antara START TRANSACTION... COMMIT. Ketika prosedur dipanggil di luar blok transaksi, dua pernyataan INSERT berada dalam transaksi yang sama (xid=1866).

```
CREATE OR REPLACE PROCEDURE sp_nonatomic_txn_block(a int, b int) NONATOMIC AS
$$
BEGIN
    START TRANSACTION;
    INSERT INTO test_table_a values (a);
    INSERT INTO test_table_b values (b);
    COMMIT;
END;
$$
LANGUAGE plpgsql;

Call sp_nonatomic_txn_block(1,2);

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |                stmt_text
--------+------+------------+---------+-----------------------------------------
      1 | 1865 | 1073823998 | UTILITY | Call sp_nonatomic_txn_block(1,2);
      1 | 1866 | 1073823998 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1866 | 1073823998 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1866 | 1073823998 | UTILITY | COMMIT
(4 rows)
```

Ketika prosedur dipanggil dari dalam blok BEGIN... COMMIT, START TRANSACTION di dalam prosedur tidak melakukan apa-apa karena sudah ada transaksi terbuka. COMMIT di dalam prosedur melakukan transaksi saat ini (xid=1876) dan memulai yang baru.

```
Begin;
  INSERT INTO test_table_a values (10);
  Call sp_nonatomic_txn_block(20,30);
  INSERT INTO test_table_b values (40);
Commit;

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |               stmt_text
--------+------+------------+---------+----------------------------------------
      1 | 1876 | 1073832133 | UTILITY | Begin;
      1 | 1876 | 1073832133 | QUERY   | INSERT INTO test_table_a values (10);
      1 | 1876 | 1073832133 | UTILITY | Call sp_nonatomic_txn_block(20,30);
      1 | 1876 | 1073832133 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1876 | 1073832133 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1876 | 1073832133 | UTILITY | COMMIT
      1 | 1878 | 1073832133 | QUERY   | INSERT INTO test_table_b values (40);
      1 | 1878 | 1073832133 | UTILITY | COMMIT
(8 rows)
```

Contoh ini menunjukkan cara bekerja dengan loop kursor. Tabel test_table_a memiliki tiga nilai. Tujuannya adalah untuk mengulangi tiga nilai dan memasukkannya ke dalam tabel test_table_b. Jika prosedur tersimpan NONATOMIC dibuat dengan cara berikut, itu akan melemparkan kursor kesalahan “cur1" tidak ada setelah menjalankan pernyataan INSERT di loop pertama. Ini karena komit otomatis INSERT menutup kursor terbuka.

```
insert into test_table_a values (1), (2), (3);

CREATE OR REPLACE PROCEDURE sp_nonatomic_cursor() NONATOMIC
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
BEGIN
  open cur1;
```

```
  Loop
    fetch cur1 into rec;
    exit when not found;
    raise info '%', rec.v;
    insert into test_table_b values (rec.v);
  End Loop;
END
$$;

CALL sp_nonatomic_cursor();

INFO:  1
ERROR:  cursor "cur1" does not exist
CONTEXT:  PL/pgSQL function "sp_nonatomic_cursor" line 7 at fetch
```

Untuk membuat loop kursor berfungsi, letakkan di antara START TRANSACTION... COMMIT.

```
insert into test_table_a values (1), (2), (3);

CREATE OR REPLACE PROCEDURE sp_nonatomic_cursor() NONATOMIC
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
BEGIN
  START TRANSACTION;
  open cur1;
  Loop
    fetch cur1 into rec;
    exit when not found;
    raise info '%', rec.v;
    insert into test_table_b values (rec.v);
  End Loop;
  COMMIT;
END
$$;

CALL sp_nonatomic_cursor();

INFO:  1
INFO:  2
INFO:  3
```

```
CALL
```

Kesalahan menjebak

Topik ini menjelaskan cara Amazon Redshift menangani kesalahan.

Ketika kueri atau perintah dalam prosedur tersimpan menyebabkan kesalahan, kueri berikutnya tidak dijalankan dan transaksi diputar kembali. Tetapi Anda dapat menangani kesalahan menggunakan blok PENGECUALIAN.

> **Note**
>
> Perilaku default adalah bahwa kesalahan akan menyebabkan kueri berikutnya tidak berjalan, bahkan ketika tidak ada kondisi penghasil kesalahan tambahan dalam prosedur yang disimpan.

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  WHEN OTHERS THEN
    statements
END;
```

Ketika pengecualian terjadi, dan Anda menambahkan blok penanganan pengecualian, Anda dapat menulis pernyataan RAISE dan sebagian besar pernyataan lainnya PL/pgSQL . Misalnya, Anda dapat memunculkan pengecualian dengan pesan khusus atau menyisipkan catatan ke tabel logging.

Saat memasuki blok penanganan pengecualian, transaksi saat ini diputar kembali dan transaksi baru dibuat untuk menjalankan pernyataan di blok. Jika pernyataan dalam blok berjalan tanpa kesalahan, transaksi dilakukan dan pengecualian dilemparkan kembali. Terakhir, prosedur yang disimpan keluar.

Satu-satunya kondisi yang didukung dalam blok pengecualian adalah OTHERS, yang cocok dengan setiap jenis kesalahan kecuali pembatalan kueri. Juga, jika terjadi kesalahan dalam blok penanganan pengecualian, itu dapat ditangkap oleh blok penanganan pengecualian luar.

Ketika kesalahan terjadi di dalam prosedur NONATOMIC, kesalahan tidak dilemparkan kembali jika ditangani oleh blok pengecualian. Lihat PL/pgSQL pernyataan `RAISE` untuk melempar pengecualian yang ditangkap oleh blok penanganan pengecualian. Pernyataan ini hanya valid dalam blok penanganan pengecualian. Untuk mengetahui informasi selengkapnya, lihat MENAIKKAN.

Mengontrol apa yang terjadi setelah kesalahan dalam prosedur tersimpan, dengan handler CONTINUE

`CONTINUE`Handler adalah jenis handler pengecualian yang mengontrol aliran eksekusi dalam prosedur tersimpan NONATOMIC. Dengan menggunakannya, Anda dapat menangkap dan menangani pengecualian tanpa mengakhiri blok pernyataan yang ada. Biasanya, ketika kesalahan terjadi dalam prosedur yang disimpan, aliran terputus dan kesalahan dikembalikan ke pemanggil. Namun, dalam beberapa kasus penggunaan, kondisi kesalahan tidak cukup parah untuk menjamin gangguan aliran. Anda mungkin ingin menangani kesalahan dengan anggun, menggunakan logika penanganan kesalahan yang Anda pilih dalam transaksi terpisah, dan kemudian melanjutkan menjalankan pernyataan yang mengikuti kesalahan. Berikut ini menunjukkan sintaks.

```
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  [ CONTINUE_HANDLER | EXIT_HANDLER ] WHEN OTHERS THEN
    handler_statements
END;
```

Ada beberapa tabel sistem yang tersedia untuk membantu Anda mengumpulkan informasi tentang berbagai jenis kesalahan. Lihat informasi selengkapnya di STL_LOAD_ERROR, STL_ERROR, dan SYS_STREAM_SCAN_ERROR. Ada juga tabel sistem tambahan yang dapat Anda gunakan untuk memecahkan masalah kesalahan. Informasi lebih lanjut tentang ini dapat ditemukan diTabel sistem dan referensi tampilan.

Contoh

Contoh berikut menunjukkan cara menulis pernyataan di blok penanganan pengecualian. Prosedur tersimpan menggunakan perilaku manajemen transaksi default.

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);
```

```
CREATE OR REPLACE PROCEDURE update_employee_sp() AS
$$
BEGIN
    UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
    EXECUTE 'select invalid';
EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'An exception occurred.';
    INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp();

INFO:  An exception occurred.
ERROR:  column "invalid" does not exist
CONTEXT:  SQL statement "select invalid"
PL/pgSQL function "update_employee_sp" line 3 at execute statement
```

Dalam contoh ini, jika Anda memanggil`update_employee_sp`, pesan informasi Pengecu alian terjadi. dinaikkan dan pesan kesalahan dimasukkan ke dalam `employee_error_log` log tabel logging. Pengecualian asli dilemparkan lagi sebelum prosedur tersimpan keluar. Kueri berikut menunjukkan catatan yang dihasilkan dari menjalankan contoh.

```
SELECT * from employee;

firstname | lastname
-----------+----------
 Tomas     | Smith

SELECT * from employee_error_log;

          message
-----------------------------------------------
 Error message: column "invalid" does not exist
```

Untuk informasi selengkapnya tentang RAISE, termasuk bantuan pemformatan dan daftar level tambahan, lihatPL/pgSQL Pernyataan yang didukung.

Contoh berikut menunjukkan cara menulis pernyataan di blok penanganan pengecualian. Prosedur tersimpan menggunakan perilaku manajemen transaksi NONATOMIC. Dalam contoh ini, tidak

ada kesalahan yang dikembalikan ke pemanggil setelah panggilan prosedur selesai. Pernyataan UPDATE tidak diputar kembali karena kesalahan dalam pernyataan berikutnya. Pesan informasi dimunculkan dan pesan kesalahan dimasukkan ke dalam tabel logging.

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

-- Create the SP in NONATOMIC mode
CREATE OR REPLACE PROCEDURE update_employee_sp_2() NONATOMIC AS
$$
BEGIN
    UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
    EXECUTE 'select invalid';
EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'An exception occurred.';
    INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp_2();
INFO:  An exception occurred.
CALL

SELECT * from employee;

 firstname | lastname
-----------+----------
 Adam      | Smith
(1 row)

SELECT * from employee_error_log;

                    message
-----------------------------------------------
 Error message: column "invalid" does not exist
(1 row)
```

Contoh ini menunjukkan cara membuat prosedur dengan dua sub blok. Ketika prosedur tersimpan dipanggil, kesalahan dari sub blok pertama ditangani oleh blok penanganan pengecualian. Setelah sub blok pertama selesai, prosedur terus mengeksekusi sub blok kedua. Anda dapat melihat dari

hasil bahwa tidak ada kesalahan yang muncul saat panggilan prosedur selesai. Operasi UPDATE dan INSERT pada meja karyawan berkomitmen. Pesan kesalahan dari kedua blok pengecualian dimasukkan dalam tabel logging.

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

CREATE OR REPLACE PROCEDURE update_employee_sp_3() NONATOMIC AS
$$
BEGIN
    BEGIN
        UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
        EXECUTE 'select invalid1';
    EXCEPTION WHEN OTHERS THEN
        RAISE INFO 'An exception occurred in the first block.';
        INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
    END;
    BEGIN
        INSERT INTO employee VALUES ('Edie','Robertson');
        EXECUTE 'select invalid2';
    EXCEPTION WHEN OTHERS THEN
        RAISE INFO 'An exception occurred in the second block.';
        INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
    END;
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp_3();
INFO:  An exception occurred in the first block.
INFO:  An exception occurred in the second block.
CALL

SELECT * from employee;

 firstname | lastname
-----------+-----------
 Adam      | Smith
 Edie      | Robertson
(2 rows)

SELECT * from employee_error_log;
```

```
                    message
------------------------------------------------
 Error message: column "invalid1" does not exist
 Error message: column "invalid2" does not exist
(2 rows)
```

Contoh berikut menunjukkan cara menggunakan handler pengecualian CONTINUE. Sampel ini membuat dua tabel dan menggunakannya dalam prosedur tersimpan. Handler CONTINUE mengontrol aliran eksekusi dalam prosedur tersimpan dengan perilaku manajemen transaksi NONATOMIC.

```
CREATE TABLE tbl_1 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_1() NONATOMIC AS
$$
BEGIN
    INSERT INTO tbl_1 VALUES (1);
    -- Expect an error for the insert statement following, because of the invalid value
    INSERT INTO tbl_1 VALUES ("val");
    INSERT INTO tbl_1 VALUES (2);
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

Panggil prosedur yang disimpan:

```
CALL sp_exc_handling_1();
```

Aliran berlangsung seperti ini:

1. Kesalahan terjadi karena upaya dilakukan untuk memasukkan tipe data yang tidak kompatibel dalam kolom. Kontrol diteruskan ke blok PENGECUALIAN. Ketika blok penanganan pengecualian dimasukkan, transaksi saat ini diputar kembali dan transaksi implisit baru dibuat untuk menjalankan pernyataan di dalamnya.
2. Jika pernyataan di CONTINUE_HANDLER berjalan tanpa kesalahan, kontrol diteruskan ke pernyataan segera setelah pernyataan yang menyebabkan pengecualian. (Jika pernyataan di

CONTINUE_HANDLER memunculkan pengecualian baru, Anda dapat menanganinya dengan handler pengecualian dalam blok EXCEPTION.)

Setelah Anda memanggil prosedur tersimpan sampel, tabel berisi catatan berikut:

- Jika Anda menjalankan`SELECT * FROM tbl_1;`, itu mengembalikan dua catatan. Ini berisi nilai-nilai 1 dan2.
- Jika Anda menjalankan`SELECT * FROM tbl_error_logging;`, ia mengembalikan satu catatan dengan nilai-nilai ini: Kesalahan yang ditemui, 42703, dan kolom “val” tidak ada di tbl_1.

Contoh penanganan kesalahan tambahan berikut menggunakan handler EXIT dan handler CONTINUE. Ini membuat dua tabel: tabel data dan tabel logging. Ini juga membuat prosedur tersimpan yang menunjukkan penanganan kesalahan:

```
CREATE TABLE tbl_1 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_2() NONATOMIC AS
$$
BEGIN
    INSERT INTO tbl_1 VALUES (1);
    BEGIN
        INSERT INTO tbl_1 VALUES (100);
        -- Expect an error for the insert statement following, because of the invalid
 value
        INSERT INTO tbl_1 VALUES ("val");
        INSERT INTO tbl_1 VALUES (101);
    EXCEPTION EXIT_HANDLER WHEN OTHERS THEN
        INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
    END;
    INSERT INTO tbl_1 VALUES (2);
    -- Expect an error for the insert statement following, because of the invalid value
    INSERT INTO tbl_1 VALUES ("val");
    INSERT INTO tbl_1 VALUES (3);
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

Setelah Anda membuat prosedur tersimpan, panggil dengan yang berikut:

```
CALL sp_exc_handling_2();
```

Ketika kesalahan terjadi di blok pengecualian bagian dalam, yang dikurung oleh set dalam BEGIN dan END, itu ditangani oleh handler EXIT. Setiap kesalahan yang terjadi di blok luar ditangani oleh handler CONTINUE.

Setelah Anda memanggil prosedur tersimpan sampel, tabel berisi catatan berikut:

- Jika Anda menjalankan`SELECT * FROM tbl_1;`, ia mengembalikan empat catatan, dengan nilai 1, 2, 3, dan 100.
- Jika Anda menjalankan`SELECT * FROM tbl_error_logging;`, itu mengembalikan dua catatan. Mereka memiliki nilai-nilai ini: Kes alahan yang dit emui, 42703, dan kolom “val” tidak ada di tbl_1.

Jika tabel tbl_error_logging tidak ada, itu menimbulkan pengecualian.

Contoh berikut menunjukkan cara menggunakan handler pengecualian CONTINUE dengan loop FOR. Sampel ini membuat tiga tabel dan menggunakannya dalam loop FOR dalam prosedur tersimpan. Loop FOR adalah varian set hasil, yang berarti bahwa ia mengulangi hasil kueri:

```
CREATE TABLE tbl_1 (a int);
INSERT INTO tbl_1 VALUES (1), (2), (3);
CREATE TABLE tbl_2 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_loop() NONATOMIC AS
$$
DECLARE
 rec RECORD;
BEGIN
    FOR rec IN SELECT a FROM tbl_1
    LOOP
        IF rec.a = 2 THEN
            -- Expect an error for the insert statement following, because of the
 invalid value
            INSERT INTO tbl_2 VALUES("val");
        ELSE
            INSERT INTO tbl_2 VALUES (rec.a);
        END IF;
    END LOOP;
```

```
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

Panggil prosedur yang disimpan:

```
CALL sp_exc_handling_loop();
```

Setelah Anda memanggil prosedur tersimpan sampel, tabel berisi catatan berikut:

- Jika Anda menjalankan`SELECT * FROM tbl_2;`, itu mengembalikan dua catatan. Ini berisi nilai 1 dan 3.
- Jika Anda menjalankan`SELECT * FROM tbl_error_logging;`, ia mengembalikan satu catatan dengan nilai-nilai ini: Kesalahan yang ditemui, 42703, dan kolom “val” tidak ada di tbl_2.

Catatan penggunaan mengenai handler CONTINUE:

- Kata kunci CONTINUE_HANDLER dan EXIT_HANDLER hanya dapat digunakan dalam prosedur tersimpan NONATOMIC.
- Kata kunci CONTINUE_HANDLER dan EXIT_HANDLER bersifat opsional. EXIT_HANDLER adalah default.

Pencatatan prosedur tersimpan

Topik ini menjelaskan prosedur dan tampilan tersimpan yang digunakan Amazon Redshift untuk pencatatan prosedur tersimpan.

Rincian tentang prosedur tersimpan dicatat dalam tabel dan tampilan sistem berikut:

- SVL_STORED_PROC_CALL — detail dicatat tentang waktu mulai dan waktu akhir panggilan prosedur yang disimpan, dan apakah panggilan berakhir sebelum selesai. Untuk informasi selengkapnya, lihat SVL_STORED_PROC_CALL.
- SVL_STORED_PROC_MESSAGES — pesan dalam prosedur tersimpan yang dipancarkan oleh kueri RAISE dicatat dengan level logging yang sesuai. Untuk informasi selengkapnya, lihat SVL_STORED_PROC_MESSAGES.
- SVL_QLOG — ID kueri dari panggilan prosedur dicatat untuk setiap kueri yang dipanggil dari prosedur tersimpan. Untuk informasi selengkapnya, lihat SVL_QLOG.

- STL_UTILITYTEXT — panggilan prosedur tersimpan dicatat setelah selesai. Untuk informasi selengkapnya, lihat STL_UTILITYTEXT.
- PG_PROC_INFO — tampilan katalog sistem ini menunjukkan informasi tentang prosedur tersimpan. Untuk informasi selengkapnya, lihat PG_PROC_INFO.

Batasan prosedur yang disimpan

Topik ini menjelaskan batasan untuk prosedur tersimpan Amazon Redshift.

Pertimbangan berikut berlaku saat Anda menggunakan prosedur tersimpan Amazon Redshift.

Perbedaan antara Amazon Redshift dan PostgreSQL untuk dukungan prosedur tersimpan

Berikut ini adalah perbedaan antara dukungan prosedur tersimpan di Amazon Redshift dan PostgreSQL:

- Amazon Redshift tidak mendukung subtransaksi, dan karenanya memiliki dukungan terbatas untuk blok penanganan pengecualian.

Pertimbangan dan batasan

Berikut ini adalah pertimbangan tentang prosedur tersimpan di Amazon Redshift:

- Jumlah maksimum prosedur tersimpan untuk database adalah 10.000.
- Ukuran maksimum kode sumber untuk prosedur adalah 2 MB.
- Jumlah maksimum kursor eksplisit dan implisit yang dapat Anda buka secara bersamaan dalam sesi pengguna adalah satu. Loop FOR yang mengulangi kumpulan hasil pernyataan SQL membuka kursor implisit. Kursor bersarang tidak didukung.
- Kursor eksplisit dan implisit memiliki batasan yang sama pada ukuran set hasil seperti kursor Amazon Redshift standar. Untuk informasi selengkapnya, lihat Kendala kursor.
- Jumlah maksimum level untuk panggilan bersarang adalah 16.
- Jumlah maksimum parameter prosedur adalah 32 untuk argumen input dan 32 untuk argumen output.
- Jumlah maksimum variabel dalam prosedur tersimpan adalah 1.024.

- Setiap perintah SQL yang memerlukan konteks transaksinya sendiri tidak didukung di dalam prosedur tersimpan. Contohnya adalah:
  - MEMPERSIAPKAN
  - CREATE/DROP DATABASE
  - CREATE EXTERNAL TABLE
  - VAKUM
  - MENGATUR LOKAL
  - UBAH TABEL DITAMBAHKAN
- Pang `registerOutParameter` gilan metode melalui driver Java Database Connectivity (JDBC) tidak didukung untuk tipe `refcursor` data. Untuk contoh penggunaan tipe `refcursor` data, lihatMengembalikan hasil set dari prosedur tersimpan.

PL/pgSQL referensi bahasa

Prosedur tersimpan di Amazon Redshift didasarkan pada bahasa prosedural PostgreSQL, dengan beberapa perbedaan penting PL/pgSQL . Dalam referensi ini, Anda dapat menemukan detail PL/pgSQL sintaks seperti yang diterapkan oleh Amazon Redshift. Untuk informasi selengkapnya PL/pgSQL, lihat PL/pgSQL - Bahasa prosedural SQL dalam dokumentasi PostgreSQL.

Topik

- PL/pgSQL konvensi referensi
- Struktur PL/pgSQL
- PL/pgSQL Pernyataan yang didukung

PL/pgSQL konvensi referensi

Di bagian ini, Anda dapat menemukan konvensi yang digunakan untuk menulis sintaks untuk bahasa prosedur yang PL/pgSQL disimpan.

Karakter	Deskripsi
PENUTUP	Kata-kata dalam huruf kapital adalah kata kunci.
[ ]	Tanda kurung menunjukkan argumen opsional. Beberapa argumen dalam tanda kurung menunjukkan bahwa Anda dapat memilih sejumlah argumen.

Karakter	Deskripsi
	Selain itu, argumen dalam tanda kurung pada baris terpisah menunjukkan bahwa parser Amazon Redshift mengharapkan argumen berada dalam urutan yang tercantum dalam sintaks.
{ }	Tanda kurung menunjukkan bahwa Anda diminta untuk memilih salah satu penjelasan di dalam kurung kurawal.
|	Pipa menunjukkan bahwa Anda dapat memilih di antara argumen.
red italics	Kata-kata dalam huruf miring merah menunjukkan placeholder. Masukkan nilai yang sesuai sebagai pengganti kata dengan huruf miring merah.
. . .	Elipsis menunjukkan bahwa Anda dapat mengulangi elemen sebelumnya.
'	Kata-kata dalam tanda kutip tunggal menunjukkan bahwa Anda harus mengetikkan tanda kutip.

Struktur PL/pgSQL

PL/pgSQL adalah bahasa prosedural dengan banyak konstruksi yang sama dengan bahasa prosedural lainnya.

Topik

- Blokir
- Deklarasi variabel
- Deklarasi alias
- Built-in variabel
- Jenis rekaman

Blokir

PL/pgSQL adalah bahasa yang terstruktur blok. Tubuh lengkap prosedur didefinisikan dalam blok, yang berisi deklarasi variabel dan PL/pgSQL pernyataan. Pernyataan juga bisa berupa blok bersarang, atau subblok.

Akhiri deklarasi dan pernyataan dengan titik koma. Ikuti kata kunci END di blok atau subblok dengan titik koma. Jangan gunakan titik koma setelah kata kunci DECLARE dan BEGIN.

Anda dapat menulis semua kata kunci dan pengidentifikasi dalam huruf besar dan kecil campuran. Pengidentifikasi secara implisit dikonversi ke huruf kecil kecuali diapit dalam tanda kutip ganda.

Tanda hubung ganda (--) memulai komentar yang meluas ke akhir baris. A /* memulai komentar blok yang meluas ke kejadian berikutnya dari */. Anda tidak dapat memblokir komentar sarang. Namun, Anda dapat melampirkan komentar tanda hubung ganda dalam komentar blok, dan tanda hubung ganda dapat menyembunyikan pembatas komentar blok /* dan */.

Pernyataan apa pun di bagian pernyataan blok dapat berupa subblok. Anda dapat menggunakan subblok untuk pengelompokan logis atau untuk melokalisasi variabel ke sekelompok kecil pernyataan.

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
END [ label ];
```

Variabel yang dideklarasikan di bagian deklarasi sebelum blok diinisialisasi ke nilai defaultnya setiap kali blok dimasukkan. Dengan kata lain, mereka tidak diinisialisasi hanya sekali per panggilan fungsi.

Bagian berikut menunjukkan satu contoh.

```
CREATE PROCEDURE update_value() AS $$
DECLARE
  value integer := 20;
BEGIN
  RAISE NOTICE 'Value here is %', value;  -- Value here is 20
  value := 50;
  --
  -- Create a subblock
  --
  DECLARE
    value integer := 80;
  BEGIN
    RAISE NOTICE 'Value here is %', value;  -- Value here is 80
  END;
```

```
  RAISE NOTICE 'Value here is %', value;  -- Value here is 50
END;
$$ LANGUAGE plpgsql;
```

Gunakan label untuk mengidentifikasi blok yang akan digunakan dalam pernyataan EXIT atau untuk memenuhi syarat nama variabel yang dideklarasikan di blok.

Jangan bingung penggunaan BEGIN/END untuk mengelompokkan pernyataan PL/pgSQL dengan perintah database untuk kontrol transaksi. Begin dan END in PL/pgSQL hanya untuk pengelompokan. Mereka tidak memulai atau mengakhiri transaksi.

Deklarasi variabel

Deklarasikan semua variabel dalam blok, kecuali variabel loop, di bagian DECLARE blok. Variabel dapat menggunakan tipe data Amazon Redshift yang valid. Untuk tipe data yang didukung, lihatJenis Data.

PL/pgSQL variabel dapat berupa tipe data yang didukung Amazon Redshift, plus `RECORD` dan. `refcursor` Untuk informasi selengkapnya tentang `RECORD`, lihat Jenis rekaman. Untuk informasi selengkapnya tentang `refcursor`, lihat Cursors.

```
DECLARE
name [ CONSTANT ] type [ NOT NULL ] [ { DEFAULT | := } expression ];
```

Berikut, Anda dapat menemukan contoh deklarasi variabel.

```
customerID integer;
numberofitems numeric(6);
link varchar;
onerow RECORD;
```

Variabel loop dari loop FOR iterasi pada rentang bilangan bulat secara otomatis dinyatakan sebagai variabel integer.

Klausa DEFAULT, jika diberikan, menentukan nilai awal yang ditetapkan untuk variabel ketika blok dimasukkan. Jika klausa DEFAULT tidak diberikan, maka variabel diinisialisasi ke nilai SQL NULL. Opsi CONSTANT mencegah variabel ditetapkan, sehingga nilainya tetap konstan selama durasi blok. Jika NOT NULL ditentukan, penugasan nilai null menghasilkan kesalahan runtime. Semua variabel yang dinyatakan sebagai NOT NULL harus memiliki nilai default non-null yang ditentukan.

Nilai default dievaluasi setiap kali blok dimasukkan. Misalnya, menetapkan `now()` ke variabel tipe `timestamp` menyebabkan variabel memiliki waktu pemanggilan fungsi saat ini, bukan waktu ketika fungsi dikompilasi sebelumnya.

```
quantity INTEGER DEFAULT 32;
url VARCHAR := 'http://mysite.com';
user_id CONSTANT INTEGER := 10;
```

Tipe `refcursor` data adalah tipe data variabel kursor dalam prosedur tersimpan. `refcursor`Nilai dapat dikembalikan dari dalam prosedur yang disimpan. Untuk informasi selengkapnya, lihat Mengembalikan hasil set dari prosedur tersimpan.

Deklarasi alias

Jika tanda tangan prosedur tersimpan menghilangkan nama argumen, Anda dapat mendeklarasikan alias untuk argumen tersebut.

```
name ALIAS FOR $n;
```

Built-in variabel

Variabel bawaan berikut didukung:

- MENEMUKAN
- SQLSTATE
- SQLERRM
- DAPATKAN DIAGNOSTIK integer_var: = ROW_COUNT;

FOUND adalah variabel khusus dari jenis Boolean. FOUND memulai false dalam setiap panggilan prosedur. FOUND diatur oleh jenis pernyataan berikut:

- PILIH KE

  Set FOUND ke true jika mengembalikan baris, false jika tidak ada baris dikembalikan.
- UPDATE, INSERT, dan DELETE

  Set FOUND ke true jika setidaknya satu baris terpengaruh, false jika tidak ada baris yang terpengaruh.

- AMBIL

  Set FOUND ke true jika mengembalikan baris, false jika tidak ada baris dikembalikan.

- UNTUK pernyataan

  Set FOUND ke true jika pernyataan FOR iterasi satu kali atau lebih, dan sebaliknya false. Ini berlaku untuk ketiga varian pernyataan FOR: integer FOR loop, record-set FOR loop, dan dynamic record-set FOR loop.

  FOUND diatur ketika FOR loop keluar. Di dalam runtime loop, FOUND tidak dimodifikasi oleh pernyataan FOR. Namun, itu dapat diubah dengan menjalankan pernyataan lain di dalam badan loop.

Bagian berikut menunjukkan satu contoh.

```
CREATE TABLE employee(empname varchar);
CREATE OR REPLACE PROCEDURE show_found()
AS  $$
DECLARE
  myrec record;
BEGIN
  SELECT INTO myrec * FROM employee WHERE empname = 'John';
  IF NOT FOUND THEN
    RAISE EXCEPTION 'employee John not found';
  END IF;
END;
$$ LANGUAGE plpgsql;
```

Dalam handler pengecualian, variabel khusus SQLSTATE berisi kode kesalahan yang sesuai dengan pengecualian yang muncul. Variabel khusus SQLERRM berisi pesan kesalahan yang terkait dengan pengecualian. Variabel ini tidak terdefinisi di luar penangan pengecualian dan menampilkan kesalahan jika digunakan.

Bagian berikut menunjukkan satu contoh.

```
CREATE OR REPLACE PROCEDURE sqlstate_sqlerrm() AS
$$
BEGIN
  UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
  EXECUTE 'select invalid';
```

```
  EXCEPTION WHEN OTHERS THEN
  RAISE INFO 'error message SQLERRM %', SQLERRM;
  RAISE INFO 'error message SQLSTATE %', SQLSTATE;
END;
$$ LANGUAGE plpgsql;
```

ROW_COUNT digunakan dengan perintah GET DIAGNOSTICS. Ini menunjukkan jumlah baris yang diproses oleh perintah SQL terakhir yang dikirim ke mesin SQL.

Bagian berikut menunjukkan satu contoh.

```
CREATE OR REPLACE PROCEDURE sp_row_count() AS
$$
DECLARE
  integer_var int;
BEGIN
  INSERT INTO tbl_row_count VALUES(1);
  GET DIAGNOSTICS integer_var := ROW_COUNT;
  RAISE INFO 'rows inserted = %', integer_var;
END;
$$ LANGUAGE plpgsql;
```

Jenis rekaman

Tipe RECORD bukan tipe data yang benar, hanya placeholder. Variabel tipe rekaman mengasumsikan struktur baris sebenarnya dari baris yang ditetapkan selama perintah SELECT atau FOR. Substruktur variabel record dapat berubah setiap kali diberi nilai. Sampai variabel record pertama kali ditugaskan, ia tidak memiliki substruktur. Setiap upaya untuk mengakses bidang di dalamnya menimbulkan kesalahan runtime.

```
name RECORD;
```

Bagian berikut menunjukkan satu contoh.

```
CREATE TABLE tbl_record(a int, b int);
INSERT INTO tbl_record VALUES(1, 2);
CREATE OR REPLACE PROCEDURE record_example()
LANGUAGE plpgsql
AS $$
DECLARE
```

```
  rec RECORD;
BEGIN
  FOR rec IN SELECT a FROM tbl_record
  LOOP
    RAISE INFO 'a = %', rec.a;
  END LOOP;
END;
$$;
```

PL/pgSQL Pernyataan yang didukung

PL/pgSQL pernyataan menambah perintah SQL dengan konstruksi prosedural, termasuk perulangan dan ekspresi kondisional, untuk mengontrol aliran logis. Sebagian besar perintah SQL dapat digunakan, termasuk bahasa manipulasi data (DHTML) seperti COPY, UNLOAD, dan INSERT, dan bahasa definisi data (DDL) seperti CREATE TABLE. Untuk daftar perintah SQL yang komprehensif, lihatPerintah SQL. Selain itu, PL/pgSQL pernyataan berikut didukung oleh Amazon Redshift.

Topik

- Penugasan
- PILIH KE
- No-op
- SQL Dinamis
- Nilai yang ditampilkan
- Kondisional: JIKA
- Kondisional: KASUS
- Loop
- Cursors
- MENAIKKAN
- Kontrol transaksi

Penugasan

Pernyataan penugasan memberikan nilai ke variabel. Ekspresi harus mengembalikan satu nilai.

```
identifier := expression;
```

Menggunakan nonstandar = untuk penugasan, alih-alih:=, juga diterima.

Jika tipe data ekspresi tidak cocok dengan tipe data variabel atau variabel memiliki ukuran atau presisi, nilai hasil secara implisit dikonversi.

Berikut ini menunjukkan contoh.

```
customer_number := 20;
tip := subtotal * 0.15;
```

PILIH KE

Pernyataan SELECT INTO memberikan hasil dari beberapa kolom (tetapi hanya satu baris) ke dalam variabel rekaman atau daftar variabel skalar.

```
SELECT INTO target select_expressions FROM ...;
```

Dalam sintaks sebelumnya, *target* dapat berupa variabel rekaman atau daftar variabel sederhana dan bidang catatan yang dipisahkan koma. *select_expressions*Daftar dan sisa perintah sama seperti di SQL biasa.

Jika daftar variabel digunakan sebagai*target*, nilai yang dipilih harus sama persis dengan struktur target, atau kesalahan runtime terjadi. Ketika variabel rekaman adalah target, secara otomatis mengkonfigurasi dirinya sendiri ke jenis baris kolom hasil kueri.

Klausa INTO dapat muncul hampir di mana saja dalam pernyataan SELECT. Biasanya muncul tepat setelah klausa SELECT, atau tepat sebelum klausa FROM. Artinya, muncul tepat sebelum atau setelah *select_expressions* daftar.

Jika query mengembalikan nol baris, nilai NULL ditetapkan untuk*target*. Jika kueri mengembalikan beberapa baris, baris pertama ditetapkan *target* dan sisanya dibuang. Kecuali pernyataan tersebut berisi ORDER BY, baris pertama tidak deterministik.

Untuk menentukan apakah tugas mengembalikan setidaknya satu baris, gunakan variabel FOUND khusus.

```
SELECT INTO customer_rec * FROM cust WHERE custname = lname;
IF NOT FOUND THEN
  RAISE EXCEPTION 'employee % not found', lname;
```

```
END IF;
```

Untuk menguji apakah hasil record adalah null, Anda dapat menggunakan IS NULL bersyarat. Tidak ada cara untuk menentukan apakah ada baris tambahan yang mungkin telah dibuang. Contoh berikut menangani kasus di mana tidak ada baris yang dikembalikan.

```
CREATE OR REPLACE PROCEDURE select_into_null(return_webpage OUT varchar(256))
AS $$
DECLARE
  customer_rec RECORD;
BEGIN
  SELECT INTO customer_rec * FROM users WHERE user_id=3;
  IF customer_rec.webpage IS NULL THEN
    -- user entered no webpage, return "http://"
    return_webpage = 'http://';
  END IF;
END;
$$ LANGUAGE plpgsql;
```

No-op

Pernyataan no-op (`NULL;`) adalah pernyataan placeholder yang tidak melakukan apa-apa. Pernyataan no-op dapat menunjukkan bahwa satu cabang IF-THEN-ELSE rantai kosong.

```
NULL;
```

SQL Dinamis

Untuk menghasilkan perintah dinamis yang dapat melibatkan tabel yang berbeda atau tipe data yang berbeda setiap kali dijalankan dari prosedur yang PL/pgSQL disimpan, gunakan `EXECUTE` pernyataan tersebut.

```
EXECUTE command-string [ INTO target ];
```

Di bagian sebelumnya, *`command-string`* adalah ekspresi yang menghasilkan string (dari jenis teks) yang berisi perintah yang akan dijalankan. *`command-string`*Nilai ini dikirim ke mesin SQL. Tidak ada substitusi PL/pgSQL variabel yang dilakukan pada string perintah. Nilai-nilai variabel harus dimasukkan dalam string perintah seperti yang dibangun.

> **Note**
>
> Anda tidak dapat menggunakan pernyataan COMMIT dan ROLLBACK dari dalam SQL dinamis. Untuk informasi tentang menggunakan pernyataan COMMIT dan ROLLBACK dalam prosedur tersimpan, lihat. Mengelola transaksi

Saat bekerja dengan perintah dinamis, Anda sering harus menangani pelolosan tanda kutip tunggal. Sebaiknya lampirkan teks tetap dalam tanda kutip di badan fungsi Anda menggunakan kutipan dolar. Nilai dinamis untuk dimasukkan ke dalam kueri yang dibangun memerlukan penanganan khusus karena mungkin mengandung tanda kutip. Contoh berikut mengasumsikan kutipan dolar untuk fungsi secara keseluruhan, sehingga tanda kutip tidak perlu digandakan.

```
EXECUTE 'UPDATE tbl SET '
  || quote_ident(colname)
  || ' = '
  || quote_literal(newvalue)
  || ' WHERE key = '
  || quote_literal(keyvalue);
```

Contoh sebelumnya menunjukkan fungsi `quote_ident(text)` dan. `quote_literal(text)` Contoh ini meneruskan variabel yang berisi pengidentifikasi kolom dan tabel ke `quote_ident` fungsi. Ini juga melewati variabel yang berisi string literal dalam perintah yang dibangun ke `quote_literal` fungsi. Kedua fungsi mengambil langkah yang tepat untuk mengembalikan teks input yang diapit tanda kutip ganda atau tunggal masing-masing, dengan karakter khusus yang disematkan dengan benar lolos.

Kutipan dolar hanya berguna untuk mengutip teks tetap. Jangan menulis contoh sebelumnya dalam format berikut.

```
EXECUTE 'UPDATE tbl SET '
  || quote_ident(colname)
  || ' = $$'
  || newvalue
  || '$$ WHERE key = '
  || quote_literal(keyvalue);
```

Anda tidak melakukan ini karena contoh rusak jika konten `newvalue` kebetulan berisi $$. Masalah yang sama berlaku untuk pembatas kutipan dolar lainnya yang mungkin Anda pilih. Untuk mengutip teks yang tidak diketahui sebelumnya dengan aman, gunakan `quote_literal` fungsinya.

Nilai yang ditampilkan

Pernyataan RETURN kembali ke pemanggil dari prosedur yang disimpan.

```
RETURN;
```

Bagian berikut menunjukkan satu contoh.

```
CREATE OR REPLACE PROCEDURE return_example(a int)
AS $$
BEGIN
  FOR b in 1..10 LOOP
    IF b < a THEN
      RAISE INFO 'b = %', b;
    ELSE
      RETURN;
    END IF;
  END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Kondisional: JIKA

Pernyataan bersyarat IF dapat mengambil formulir berikut dalam PL/pgSQL bahasa yang digunakan Amazon Redshift:

- JIKA... KEMUDIAN

  ```
  IF boolean-expression THEN
    statements
  END IF;
  ```

  Bagian berikut menunjukkan satu contoh.

  ```
  IF v_user_id <> 0 THEN
    UPDATE users SET email = v_email WHERE user_id = v_user_id;
  END IF;
  ```

- JIKA... LALU... LAIN

```
IF boolean-expression THEN
  statements
ELSE
  statements
END IF;
```

Bagian berikut menunjukkan satu contoh.

```
IF parentid IS NULL OR parentid = ''
THEN
  return_name = fullname;
  RETURN;
ELSE
  return_name = hp_true_filename(parentid) || '/' || fullname;
  RETURN;
END IF;
```

- JIKA... LALU... ELSIF... LALU... LAIN

Kata kunci ELSIF juga bisa dieja ELSEIF.

```
IF boolean-expression THEN
  statements
[ ELSIF boolean-expression THEN
  statements
[ ELSIF boolean-expression THEN
  statements
    ...] ]
[ ELSE
  statements ]
END IF;
```

Bagian berikut menunjukkan satu contoh.

```
IF number = 0 THEN
  result := 'zero';
ELSIF number > 0 THEN
  result := 'positive';
ELSIF number < 0 THEN
  result := 'negative';
```

```
ELSE
  -- the only other possibility is that number is null
  result := 'NULL';
END IF;
```

Kondisional: KASUS

Pernyataan bersyarat CASE dapat mengambil formulir berikut dalam PL/pgSQL bahasa yang digunakan Amazon Redshift:

- KASUS Sederhana

```
CASE search-expression
WHEN expression [, expression [ ... ]] THEN
  statements
[ WHEN expression [, expression [ ... ]] THEN
  statements
  ... ]
[ ELSE
  statements ]
END CASE;
```

Sebuah pernyataan CASE sederhana memberikan eksekusi bersyarat berdasarkan kesetaraan operan.

*search-expression*Nilai dievaluasi satu kali dan berturut-turut dibandingkan dengan masing-masing *expression* dalam klausa WHEN. Jika kecocokan ditemukan, maka *statements* run yang sesuai, dan kemudian kontrol lolos ke pernyataan berikutnya setelah END CASE. Ekspresi WHEN berikutnya tidak dievaluasi. Jika tidak ada kecocokan yang ditemukan, ELSE *statements* berjalan. Namun, jika ELSE tidak ada, maka pengecualian CASE_NOT_FOUND dimunculkan.

Bagian berikut menunjukkan satu contoh.

```
CASE x
WHEN 1, 2 THEN
  msg := 'one or two';
ELSE
  msg := 'other value than one or two';
END CASE;
```

- Dicari CASE

```
CASE
WHEN boolean-expression THEN
  statements
[ WHEN boolean-expression THEN
  statements
  ... ]
[ ELSE
  statements ]
END CASE;
```

Bentuk CASE yang dicari menyediakan eksekusi bersyarat berdasarkan kebenaran ekspresi Boolean.

Setiap klausa WHEN *boolean-expression* dievaluasi secara bergantian, sampai ditemukan yang menghasilkan benar. Kemudian pernyataan yang sesuai berjalan, dan kemudian kontrol lolos ke pernyataan berikutnya setelah END CASE. Selanjutnya KAPAN *expressions* tidak dievaluasi. Jika tidak ada hasil yang benar ditemukan, ELSE *statements* dijalankan. Namun, jika ELSE tidak ada, maka pengecualian CASE_NOT_FOUND dimunculkan.

Bagian berikut menunjukkan satu contoh.

```
CASE
WHEN x BETWEEN 0 AND 10 THEN
  msg := 'value is between zero and ten';
WHEN x BETWEEN 11 AND 20 THEN
  msg := 'value is between eleven and twenty';
END CASE;
```

Loop

Pernyataan loop dapat mengambil formulir berikut dalam PL/pgSQL bahasa yang digunakan Amazon Redshift:

- Loop sederhana

```
[<<label>>]
LOOP
  statements
```

```
END LOOP [ label ];
```

Loop sederhana mendefinisikan loop tanpa syarat yang diulang tanpa batas hingga dihentikan oleh pernyataan EXIT atau RETURN. Label opsional dapat digunakan oleh pernyataan EXIT dan CONTINUE dalam loop bersarang untuk menentukan loop mana yang dirujuk oleh pernyataan EXIT dan CONTINUE.

Bagian berikut menunjukkan satu contoh.

```
CREATE OR REPLACE PROCEDURE simple_loop()
LANGUAGE plpgsql
AS $$
BEGIN
  <<simple_while>>
  LOOP
    RAISE INFO 'I am raised once';
    EXIT simple_while;
    RAISE INFO 'I am not raised';
  END LOOP;
  RAISE INFO 'I am raised once as well';
END;
$$;
```

- Keluar dari loop

```
EXIT [ label ] [ WHEN expression ];
```

Jika *label* tidak ada, loop terdalam dihentikan dan pernyataan berikut END LOOP berjalan berikutnya. Jika *label* ada, itu harus berupa label arus atau tingkat luar loop atau blok bersarang. Kemudian, loop atau blok bernama dihentikan dan kontrol berlanjut dengan pernyataan setelah loop atau blok END yang sesuai.

Jika WHEN ditentukan, loop exit hanya terjadi jika *expression* benar. Jika tidak, kontrol lolos ke pernyataan setelah EXIT.

Anda dapat menggunakan EXIT dengan semua jenis loop; itu tidak terbatas untuk digunakan dengan loop tanpa syarat.

Ketika digunakan dengan blok BEGIN, EXIT meneruskan kontrol ke pernyataan berikutnya setelah akhir blok. Label harus digunakan untuk tujuan ini. EXIT yang tidak berlabel tidak pernah dianggap cocok dengan blok BEGIN.

Bagian berikut menunjukkan satu contoh.

```
CREATE OR REPLACE PROCEDURE simple_loop_when(x int)
LANGUAGE plpgsql
AS $$
DECLARE i INTEGER := 0;
BEGIN
  <<simple_loop_when>>
  LOOP
    RAISE INFO 'i %', i;
    i := i + 1;
    EXIT simple_loop_when WHEN (i >= x);
  END LOOP;
END;
$$;
```

- Lanjutkan loop

```
CONTINUE [ label ] [ WHEN expression ];
```

Jika tidak *label* diberikan, eksekusi melompat ke iterasi berikutnya dari loop terdalam. Artinya, semua pernyataan yang tersisa di badan loop dilewati. Kontrol kemudian kembali ke ekspresi kontrol loop (jika ada) untuk menentukan apakah iterasi loop lain diperlukan. Jika *label* ada, itu menentukan label loop yang eksekusinya dilanjutkan.

Jika WHEN ditentukan, iterasi loop berikutnya dimulai hanya jika *expression* benar. Jika tidak, kontrol lolos ke pernyataan setelah LANJUTKAN.

Anda dapat menggunakan CONTINUE dengan semua jenis loop; itu tidak terbatas untuk digunakan dengan loop tanpa syarat.

```
CONTINUE mylabel;
```

- LINGKARAN SEMENTARA

```
[<<label>>]
```

```
WHILE expression LOOP
  statements
END LOOP [ label ];
```

Pernyataan WHILE mengulangi urutan pernyataan selama *boolean-expression* evaluasi menjadi benar. Ekspresi diperiksa tepat sebelum setiap entri ke badan loop.

Bagian berikut menunjukkan satu contoh.

```
WHILE amount_owed > 0 AND gift_certificate_balance > 0 LOOP
  -- some computations here
END LOOP;

WHILE NOT done LOOP
  -- some computations here
END LOOP;
```

- UNTUK loop (varian integer)

```
[<<label>>]
FOR name IN [ REVERSE ] expression .. expression LOOP
  statements
END LOOP [ label ];
```

Loop FOR (varian integer) menciptakan loop yang iterasi pada rentang nilai integer. Nama variabel secara otomatis didefinisikan sebagai tipe integer dan hanya ada di dalam loop. Setiap definisi yang ada dari nama variabel diabaikan dalam loop. Dua ekspresi yang memberikan batas bawah dan atas rentang dievaluasi satu kali saat memasuki loop. Jika Anda menentukan REVERSE, maka nilai langkah dikurangi, bukan ditambahkan, setelah setiap iterasi.

Jika batas bawah lebih besar dari batas atas (atau kurang dari, dalam kasus REVERSE), badan loop tidak berjalan. Tidak ada kesalahan yang muncul.

Jika label dilampirkan ke loop FOR, maka Anda dapat mereferensikan variabel loop integer dengan nama yang memenuhi syarat, menggunakan label itu.

Bagian berikut menunjukkan satu contoh.

```
FOR i IN 1..10 LOOP
  -- i will take on the values 1,2,3,4,5,6,7,8,9,10 within the loop
END LOOP;
```

```
FOR i IN REVERSE 10..1 LOOP
  -- i will take on the values 10,9,8,7,6,5,4,3,2,1 within the loop
END LOOP;
```

- UNTUK loop (varian set hasil)

```
[<<label>>]
FOR target IN query LOOP
  statements
END LOOP [ label ];
```

*target*Ini adalah variabel rekaman atau daftar variabel skalar yang dipisahkan koma. Target secara berturut-turut ditetapkan setiap baris yang dihasilkan dari kueri, dan badan loop dijalankan untuk setiap baris.

Loop FOR (varian set hasil) memungkinkan prosedur tersimpan untuk mengulangi hasil kueri dan memanipulasi data yang sesuai.

Bagian berikut menunjukkan satu contoh.

```
CREATE PROCEDURE cs_refresh_reports() AS $$
DECLARE
  reports RECORD;
BEGIN
  FOR reports IN SELECT * FROM cs_reports ORDER BY sort_key LOOP
    -- Now "reports" has one record from cs_reports
    EXECUTE 'INSERT INTO ' || quote_ident(reports.report_name) || ' ' ||
 reports.report_query;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE plpgsql;
```

- UNTUK loop dengan SQL dinamis

```
[<<label>>]
FOR record_or_row IN EXECUTE text_expression LOOP
  statements
END LOOP;
```

Sebuah FOR loop dengan SQL dinamis memungkinkan prosedur tersimpan untuk iterasi melalui hasil query dinamis dan memanipulasi data yang sesuai.

Bagian berikut menunjukkan satu contoh.

```
CREATE OR REPLACE PROCEDURE for_loop_dynamic_sql(x int)
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  query text;
BEGIN
  query := 'SELECT * FROM tbl_dynamic_sql LIMIT ' || x;
  FOR rec IN EXECUTE query
  LOOP
    RAISE INFO 'a %', rec.a;
  END LOOP;
END;
$$;
```

Cursors

Daripada menjalankan seluruh kueri sekaligus, Anda dapat mengatur kursor. Kursor merangkum kueri dan membaca hasil kueri beberapa baris sekaligus. Salah satu alasan untuk melakukan ini adalah untuk menghindari overrun memori ketika hasilnya berisi sejumlah besar baris. Alasan lain adalah mengembalikan referensi ke kursor yang telah dibuat oleh prosedur tersimpan, yang memungkinkan pemanggil membaca baris. Pendekatan ini menyediakan cara yang efisien untuk mengembalikan set baris besar dari prosedur yang disimpan.

Untuk menggunakan kursor dalam prosedur tersimpan NONATOMIC, tempatkan kursor loop antara MULAI TRANSAKSI... COMMIT.

Untuk mengatur kursor, pertama Anda mendeklarasikan variabel kursor. Semua akses ke kursor PL/pgSQL masuk melalui variabel kursor, yang selalu dari tipe data khusus. `refcursor` Tipe `refcursor` data hanya memegang referensi ke kursor.

Anda dapat membuat variabel kursor dengan mendeklarasikannya sebagai variabel tipe. `refcursor` Atau, Anda dapat menggunakan sintaks deklarasi kursor berikut.

```
name CURSOR [ ( arguments ) ] FOR query ;
```

Dalam sebelumnya, *arguments* (jika ditentukan) adalah daftar *name datatype* pasangan yang dipisahkan koma yang masing-masing menentukan nama yang akan diganti dengan nilai parameter di. *query* Nilai aktual untuk menggantikan nama-nama ini ditentukan kemudian, ketika kursor dibuka.

Berikut ini menunjukkan contoh.

```
DECLARE
  curs1 refcursor;
  curs2 CURSOR FOR SELECT * FROM tenk1;
  curs3 CURSOR (key integer) IS SELECT * FROM tenk1 WHERE unique1 = key;
```

Ketiga variabel ini memiliki tipe data`refcursor`, tetapi yang pertama dapat digunakan dengan kueri apa pun. Sebaliknya, yang kedua memiliki kueri yang ditentukan sepenuhnya yang sudah terikat padanya, dan yang terakhir memiliki kueri berparameter yang terikat padanya. `key`Nilai digantikan oleh nilai parameter integer ketika kursor dibuka. Variabel `curs1` dikatakan tidak terikat karena tidak terikat pada kueri tertentu.

Sebelum Anda dapat menggunakan kursor untuk mengambil baris, itu harus dibuka. PL/pgSQL memiliki tiga bentuk pernyataan OPEN, di mana dua menggunakan variabel kursor tidak terikat dan yang ketiga menggunakan variabel kursor terikat:

- Buka untuk pilih: Variabel kursor dibuka dan diberikan kueri yang ditentukan untuk dijalankan. Kursor belum bisa dibuka. Juga, itu pasti telah dideklarasikan sebagai kursor tidak terikat (yaitu, sebagai `refcursor` variabel sederhana). Kueri SELECT diperlakukan dengan cara yang sama seperti pernyataan SELECT lainnya di PL/pgSQL.

  ```
  OPEN cursor_name FOR SELECT ...;
  ```

  Bagian berikut menunjukkan satu contoh.

  ```
  OPEN curs1 FOR SELECT * FROM foo WHERE key = mykey;
  ```

- Buka untuk dieksekusi: Variabel kursor dibuka dan diberikan kueri yang ditentukan untuk dijalankan. Kursor belum bisa dibuka. Juga, itu pasti telah dideklarasikan sebagai kursor tidak terikat (yaitu, sebagai `refcursor` variabel sederhana). Query ditentukan sebagai ekspresi string

dengan cara yang sama seperti pada perintah EXECUTE. Pendekatan ini memberikan fleksibilitas sehingga kueri dapat bervariasi dari satu run ke yang berikutnya.

```
OPEN cursor_name FOR EXECUTE query_string;
```

Bagian berikut menunjukkan satu contoh.

```
OPEN curs1 FOR EXECUTE 'SELECT * FROM ' || quote_ident($1);
```

- Buka kursor terikat: Bentuk OPEN ini digunakan untuk membuka variabel kursor yang kuerinya terikat padanya ketika dideklarasikan. Kursor belum bisa dibuka. Daftar ekspresi nilai argumen aktual harus muncul jika dan hanya jika kursor dideklarasikan untuk mengambil argumen. Nilai-nilai ini diganti dalam kueri.

```
OPEN bound_cursor_name [ ( argument_values ) ];
```

Bagian berikut menunjukkan satu contoh.

```
OPEN curs2;
OPEN curs3(42);
```

Setelah kursor dibuka, Anda dapat bekerja dengannya dengan menggunakan pernyataan yang dijelaskan berikut. Pernyataan ini tidak harus terjadi dalam prosedur tersimpan yang sama yang membuka kursor. Anda dapat mengembalikan `refcursor` nilai dari prosedur yang disimpan dan membiarkan pemanggil beroperasi pada kursor. Semua portal ditutup secara implisit pada akhir transaksi. Dengan demikian, Anda dapat menggunakan `refcursor` nilai untuk mereferensikan kursor terbuka hanya sampai akhir transaksi.

- FETCH mengambil baris berikutnya dari kursor ke target. Target ini dapat berupa variabel baris, variabel rekaman, atau daftar variabel sederhana yang dipisahkan koma, seperti halnya SELECT INTO. Seperti halnya SELECT INTO, Anda dapat memeriksa variabel khusus FOUND untuk melihat apakah baris diperoleh.

```
FETCH cursor INTO target;
```

Bagian berikut menunjukkan satu contoh.

```
FETCH curs1 INTO rowvar;
```

- CLOSE menutup portal yang mendasari kursor terbuka. Anda dapat menggunakan pernyataan ini untuk melepaskan sumber daya lebih awal dari akhir transaksi. Anda juga dapat menggunakan pernyataan ini untuk membebaskan variabel kursor untuk dibuka lagi.

  ```
  CLOSE cursor;
  ```

  Bagian berikut menunjukkan satu contoh.

  ```
  CLOSE curs1;
  ```

MENAIKKAN

Gunakan `RAISE level` pernyataan untuk melaporkan pesan dan memunculkan kesalahan.

```
RAISE level 'format' [, variable [, ...]];
```

Level yang mungkin adalah NOTICE, INFO, LOG, WARNING, dan EXCEPTION. EXCEPTION menimbulkan kesalahan, yang biasanya membatalkan transaksi saat ini. Level lain hanya menghasilkan pesan dari tingkat prioritas yang berbeda.

Di dalam format string,% digantikan oleh representasi string argumen opsional berikutnya. Tulis%% untuk memancarkan% literal. Saat ini, argumen opsional harus variabel sederhana, bukan ekspresi, dan formatnya harus berupa string literal sederhana.

Dalam contoh berikut, nilai `v_job_id` menggantikan% dalam string.

```
RAISE NOTICE 'Calling cs_create_job(%)', v_job_id;
```

Gunakan `RAISE` pernyataan untuk melempar kembali pengecualian yang ditangkap oleh blok penanganan pengecualian. Pernyataan ini hanya valid di blok penanganan pengecualian dari prosedur tersimpan mode NONATOMIC.

```
RAISE;
```

Kontrol transaksi

Anda dapat bekerja dengan pernyataan kontrol transaksi dalam PL/pgSQL bahasa yang digunakan Amazon Redshift. Untuk informasi tentang menggunakan pernyataan COMMIT, ROLLBACK, dan TRUNCATE dalam prosedur tersimpan, lihat. Mengelola transaksi

Dalam prosedur tersimpan mode NONATOMIC, gunakan `START TRANSACTION` untuk memulai blok transaksi.

```
START TRANSACTION;
```

> **Note**
>
> PL/pgSQL Pernyataan MULAI TRANSAKSI berbeda dari perintah SQL MULAI TRANSAKSI dengan cara berikut:
>
> - Dalam prosedur tersimpan, MULAI TRANSAKSI tidak identik dengan BEGIN.
> - PL/pgSQL Pernyataan tersebut tidak mendukung tingkat isolasi opsional dan kata kunci izin akses.

Tampilan yang terwujud di Amazon Redshift

Bagian ini menjelaskan cara membuat dan menggunakan tampilan terwujud di Amazon Redshift. Tampilan terwujud adalah objek database yang menyimpan hasil kueri, yang dapat digunakan untuk meningkatkan kinerja dan efisiensi.

Dalam lingkungan gudang data, aplikasi sering harus melakukan kueri kompleks pada tabel besar. Contohnya adalah pernyataan SELECT yang melakukan gabungan multi-tabel dan agregasi pada tabel yang berisi miliaran baris. Memproses kueri ini bisa mahal, dalam hal sumber daya sistem dan waktu yang dibutuhkan untuk menghitung hasilnya.

Tampilan terwujud di Amazon Redshift menyediakan cara untuk mengatasi masalah ini. T ampilan terwujud berisi kumpulan hasil yang telah dihitung sebelumnya, berdasarkan kueri SQL pada satu atau lebih tabel dasar. Anda dapat mengeluarkan pernyataan SELECT untuk menanyakan tampilan yang terwujud, dengan cara yang sama seperti Anda dapat menanyakan tabel atau tampilan lain dalam database. Amazon Redshift mengembalikan hasil yang telah dihitung sebelumnya dari tampilan yang terwujud, tanpa harus mengakses tabel dasar sama sekali. Dari sudut pandang pengguna, hasil kueri dikembalikan jauh lebih cepat dibandingkan saat mengambil data yang sama dari tabel dasar.

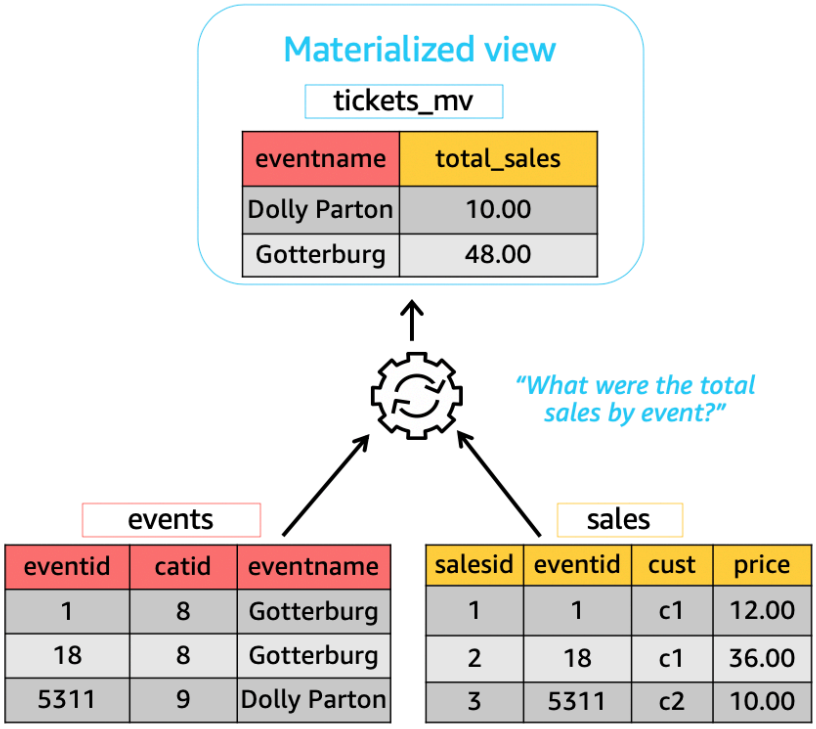
Tampilan terwujud sangat berguna untuk mempercepat kueri yang dapat diprediksi dan diulang. Alih-alih melakukan kueri intensif sumber daya terhadap tabel besar (seperti agregat atau beberapa gabungan), aplikasi dapat menanyakan tampilan yang terwujud dan mengambil kumpulan hasil yang telah dihitung sebelumnya. Misalnya, pertimbangkan skenario di mana serangkaian kueri digunakan untuk mengisi dasbor, seperti Amazon Quick. Kasus penggunaan ini sangat ideal untuk tampilan yang terwujud, karena kueri dapat diprediksi dan diulang berulang-ulang.

Anda dapat menentukan tampilan yang terwujud dalam hal pandangan terwujud lainnya. Gunakan pandangan yang terwujud pada tampilan yang terwujud untuk memperluas kemampuan tampilan yang terwujud. Dalam pendekatan ini, tampilan terwujud yang ada memainkan peran yang sama dengan tabel dasar untuk kueri untuk mengambil data.

Pendekatan ini sangat berguna untuk menggunakan kembali gabungan yang telah dihitung sebelumnya untuk agregat atau opsi GROUP BY yang berbeda. Misalnya, ambil tampilan terwujud yang menggabungkan informasi pelanggan (berisi jutaan baris) dengan informasi detail pesanan item (berisi miliaran baris). Ini adalah kueri mahal untuk dihitung sesuai permintaan berulang kali. Anda dapat menggunakan opsi GROUP BY yang berbeda untuk tampilan terwujud yang dibuat di

atas tampilan terwujud ini dan bergabung dengan tabel lain. Melakukan hal ini menghemat waktu komputasi jika tidak digunakan untuk menjalankan penggabungan dasar yang mahal setiap saat. STV_MV_DEPSTabel menunjukkan dependensi dari tampilan terwujud pada tampilan terwujud lainnya.

Saat Anda membuat tampilan terwujud, Amazon Redshift menjalankan pernyataan SQL yang ditentukan pengguna untuk mengumpulkan data dari tabel dasar atau tabel dan menyimpan kumpulan hasil. Ilustrasi berikut memberikan gambaran umum dari tampilan terwujud `tickets_mv` yang didefinisikan kueri SQL dengan menggunakan dua tabel dasar, `events` dan`sales`.



Anda kemudian dapat menggunakan tampilan terwujud ini dalam kueri untuk mempercepatnya. Selain itu, Amazon Redshift dapat secara otomatis menulis ulang kueri ini untuk menggunakan tampilan yang terwujud, bahkan ketika kueri tidak secara eksplisit mereferensikan tampilan yang terwujud. Penulisan ulang kueri secara otomatis sangat kuat dalam meningkatkan kinerja ketika Anda tidak dapat mengubah kueri Anda untuk menggunakan tampilan yang terwujud.

Untuk memperbarui data dalam tampilan terwujud, Anda dapat menggunakan pernyataan REFRESH MATERIALIZED VIEW kapan saja untuk menyegarkan tampilan terwujud secara manual. Amazon Redshift mengidentifikasi perubahan yang telah terjadi di tabel dasar atau tabel, dan kemudian menerapkan perubahan tersebut ke tampilan yang terwujud. Karena penulisan ulang kueri secara otomatis memerlukan tampilan yang terwujud untuk diperbarui, sebagai pemilik tampilan yang terwujud, pastikan untuk menyegarkan tampilan yang terwujud setiap kali tabel dasar berubah.

Amazon Redshift menyediakan beberapa cara untuk memperbarui tampilan yang terwujud untuk penulisan ulang otomatis. Anda dapat mengonfigurasi tampilan yang terwujud dengan opsi penyegaran otomatis untuk menyegarkan tampilan yang terwujud saat tabel dasar tampilan terwujud diperbarui. Operasi autorefresh ini berjalan pada saat sumber daya cluster tersedia untuk meminimalkan gangguan pada beban kerja lainnya. Karena penjadwalan autorefresh bergantung pada beban kerja, Anda dapat memiliki kontrol lebih besar atas kapan Amazon Redshift menyegarkan tampilan terwujud Anda. Anda dapat menjadwalkan pekerjaan penyegaran tampilan terwujud dengan menggunakan API penjadwal Amazon Redshift dan integrasi konsol. Untuk informasi selengkapnya tentang penjadwalan kueri, lihat Men jadwalkan kueri di konsol Amazon Redshift.

Melakukan hal ini sangat berguna ketika ada persyaratan perjanjian tingkat layanan (SLA) untuk data terkini dari tampilan yang terwujud. Anda juga dapat menyegarkan tampilan terwujud secara manual yang dapat Anda refresh secara otomatis. Untuk informasi tentang cara membuat tampilan yang terwujud, lihatBUAT TAMPILAN YANG TERWUJUD.

Anda dapat mengeluarkan pernyataan SELECT untuk menanyakan tampilan yang terwujud. Untuk informasi tentang cara menanyakan tampilan yang terwujud, lihatKueri tampilan yang terwujud. Kumpulan hasil akhirnya menjadi basi ketika data dimasukkan, diperbarui, dan dihapus dalam tabel dasar. Anda dapat menyegarkan tampilan terwujud kapan saja untuk memperbaruinya dengan perubahan terbaru dari tabel dasar. Untuk informasi tentang cara menyegarkan tampilan yang terwujud, lihatMENYEGARKAN TAMPILAN YANG TERWUJUD.

Untuk detail tentang perintah SQL yang digunakan untuk membuat dan mengelola tampilan yang terwujud, lihat topik perintah berikut:

- BUAT TAMPILAN YANG TERWUJUD
- MENGUBAH TAMPILAN YANG TERWUJUD
- MENYEGARKAN TAMPILAN YANG TERWUJUD
- JATUHKAN TAMPILAN TERWUJUD

Untuk informasi tentang tabel sistem dan tampilan untuk memantau tampilan yang terwujud, lihat topik berikut:

- STV_MV_INFO
- STL_MV_STATE
- SVL_MV_REFRESH_STATUS

- STV_MV_DEPS

Topik

- Kueri tampilan yang terwujud
- Penulisan ulang kueri otomatis untuk menggunakan tampilan yang terwujud
- Tampilan terwujud pada tabel danau data eksternal di Amazon Redshift Spectrum
- Menyegarkan tampilan yang terwujud
- Tampilan terwujud otomatis
- Menggunakan fungsi yang ditentukan pengguna (UDF) dalam tampilan terwujud
- Streaming konsumsi ke tampilan yang terwujud

Kueri tampilan yang terwujud

Anda dapat menggunakan tampilan terwujud dalam kueri SQL apa pun dengan mereferensikan nama tampilan terwujud sebagai sumber data, seperti tabel atau tampilan standar.

Ketika kueri mengakses tampilan terwujud, ia hanya melihat data yang disimpan dalam tampilan terwujud pada penyegaran terbarunya. Dengan demikian, kueri mungkin tidak melihat semua perubahan terbaru dari tabel dasar yang sesuai dari tampilan yang terwujud.

Jika pengguna lain ingin menanyakan tampilan yang terwujud, pemilik tampilan terwujud memberikan izin SELECT kepada pengguna tersebut. Pengguna lain tidak perlu memiliki izin SELECT pada tabel dasar yang mendasarinya. Pemilik tampilan yang terwujud juga dapat mencabut izin SELECT dari pengguna lain untuk mencegah mereka menanyakan tampilan yang terwujud. Perhatikan bahwa pengguna lain masih memerlukan izin USAGE pada skema yang berisi tabel dasar dari tampilan yang terwujud.

Jika pemilik tampilan terwujud tidak lagi memiliki izin SELECT lokal pada tabel dasar yang mendasarinya:

- Pemilik tidak dapat lagi menanyakan tampilan yang terwujud.
- Pengguna lain yang memiliki izin SELECT pada tampilan terwujud tidak dapat lagi menanyakan tampilan yang terwujud.

Ini terbatas pada izin lokal. Perubahan izin yang dikelola oleh Lake Formation tidak diverifikasi saat menanyakan tampilan yang terwujud. Ini berarti bahwa jika tabel dasar yang mendasarinya

dikelola oleh Formasi Danau dan izin pilih pada tabel dicabut di Formasi Danau, Anda masih dapat menanyakan tampilan yang terwujud.

Contoh berikut menanyakan tampilan yang `tickets_mv` terwujud. Untuk informasi selengkapnya tentang perintah SQL yang digunakan untuk membuat tampilan yang terwujud, lihatBUAT TAMPILAN YANG TERWUJUD.

```
SELECT sold
FROM tickets_mv
WHERE catgroup = 'Concerts';
```

Karena hasil kueri sudah dihitung sebelumnya, tidak perlu mengakses tabel yang mendasarinya (`category`,`event`, dan`sales`). Amazon Redshift dapat mengembalikan hasil langsung dari`tickets_mv`.

Penulisan ulang kueri otomatis untuk menggunakan tampilan yang terwujud

Anda dapat menggunakan penulisan ulang kueri otomatis dari tampilan yang terwujud di Amazon Redshift agar kueri Amazon Redshift menulis ulang untuk menggunakan tampilan yang terwujud. Melakukan hal ini mempercepat beban kerja kueri bahkan untuk kueri yang tidak secara eksplisit mereferensikan tampilan yang terwujud. Saat Amazon Redshift menulis ulang kueri, itu hanya menggunakan tampilan terwujud yang diperbarui.

Catatan penggunaan

Untuk memeriksa apakah penulisan ulang kueri otomatis digunakan untuk kueri, Anda dapat memeriksa rencana kueri atau STL_EXPLAIN. Berikut ini menunjukkan pernyataan SELECT dan output EXPLAIN dari rencana kueri asli.

```
SELECT catgroup, SUM(qtysold) AS sold
FROM category c, event e, sales s
WHERE c.catid = e.catid AND e.eventid = s.eventid
GROUP BY 1;

EXPLAIN
 XN HashAggregate  (cost=920021.24..920021.24 rows=1 width=35)
   ->  XN Hash Join DS_BCAST_INNER  (cost=440004.53..920021.22 rows=4 width=35)
         Hash Cond: ("outer".eventid = "inner".eventid)
```

```
         ->  XN Seq Scan on sales s  (cost=0.00..7.40 rows=740 width=6)
         ->  XN Hash  (cost=440004.52..440004.52 rows=1 width=37)
               ->  XN Hash Join DS_BCAST_INNER  (cost=0.01..440004.52 rows=1 width=37)
                     Hash Cond: ("outer".catid = "inner".catid)
                     ->  XN Seq Scan on event e  (cost=0.00..2.00 rows=200 width=6)
                     ->  XN Hash  (cost=0.01..0.01 rows=1 width=35)
                           ->  XN Seq Scan on category c  (cost=0.00..0.01 rows=1
 width=35)
```

Berikut ini menunjukkan output EXPLAIN setelah penulisan ulang otomatis yang berhasil. Output ini mencakup pemindaian pada tampilan terwujud dalam rencana kueri yang menggantikan bagian dari rencana kueri asli.

```
* EXPLAIN
     XN HashAggregate  (cost=11.85..12.35 rows=200 width=41)
       ->  XN Seq Scan on mv_tbl__tickets_mv__0 derived_table1  (cost=0.00..7.90
 rows=790 width=41)
```

Hanya tampilan terwujud terbaru (segar) yang dipertimbangkan untuk penulisan ulang kueri secara otomatis, terlepas dari strategi penyegaran, seperti otomatis, terjadwal, atau manual. Oleh karena itu, kueri asli mengembalikan hasil terbaru. Ketika tampilan terwujud secara eksplisit direferensikan dalam kueri, Amazon Redshift mengakses data yang saat ini disimpan dalam tampilan terwujud. Data ini mungkin tidak mencerminkan perubahan terbaru dari tabel dasar tampilan yang terwujud.

Anda dapat menggunakan penulisan ulang kueri otomatis dari tampilan terwujud yang dibuat pada cluster versi 1.0.20949 atau yang lebih baru.

Anda dapat menghentikan penulisan ulang kueri otomatis di tingkat sesi dengan menggunakan SET mv_enable_aqmv_for_session ke FALSE.

Cara kerja penulisan ulang kueri otomatis dari tampilan yang terwujud

Berdasarkan pengoptimalan internal, Amazon Redshift dapat memutuskan untuk memanggil penulisan ulang kueri otomatis dari tampilan yang terwujud secara transparan untuk memberikan eksekusi kueri yang paling optimal dengan waktu kueri serendah mungkin.

Sebagai contoh, misalkan Pengguna A membuat tampilan terwujud M1 pada tabel T1 dengan kueri. `SELECT * FROM T1` Pengguna A memiliki hak SELECT penuh pada T1. Pengguna A memberikan akses ke M1 kepada semua pengguna, termasuk Pengguna B. Namun, ketika Pengguna B mencoba menanyakan T1 secara langsung, mereka ditolak aksesnya. Ini karena Pengguna B tidak memiliki hak istimewa SELECT pada T1.

Beberapa saat kemudian, Pengguna B kembali mencoba menanyakan T1, tetapi kali ini mendapatkan hasil kembali dari T1. Ini karena penulisan ulang kueri otomatis menggunakan tampilan terwujud menulis ulang kueri Pengguna B pada tabel T1 (`SELECT <cols> FROM T1`) ke kueri pada tampilan terwujud M1 (). `SELECT <cols> FROM M1`

Batasan

Berikut ini adalah batasan untuk menggunakan penulisan ulang kueri otomatis dari tampilan yang terwujud:

- Penulisan ulang kueri otomatis berfungsi dengan tampilan terwujud yang tidak mereferensikan atau menyertakan salah satu dari berikut ini:
  - Subkueri
  - Gabungan luar kiri, kanan, atau penuh
  - Tetapkan operasi
  - Fungsi agregat apa pun, kecuali SUM, COUNT, MIN, MAX, dan AVG. (Ini adalah satu-satunya fungsi agregat yang bekerja dengan penulisan ulang kueri otomatis.)
  - Setiap fungsi agregat dengan DISTINCT
  - Fungsi jendela apa pun
  - PILIH klausa DISTINCT atau MEMILIKI
  - Tabel eksternal atau bersama
  - Pandangan terwujud lainnya
- Penulisan ulang kueri otomatis menulis ulang kueri SELECT yang merujuk ke tabel Amazon Redshift yang ditentukan pengguna. Amazon Redshift tidak menulis ulang kueri berikut:
  - BUAT TABEL AS pernyataan
  - PERNYATAAN SELECT INTO
  - Kueri pada katalog atau tabel sistem
  - Kueri dengan gabungan luar atau klausa SELECT DISTINCT
- Jika kueri tidak ditulis ulang secara otomatis, periksa apakah Anda memiliki izin SELECT pada tampilan materialisasi yang ditentukan dan mv_enable_aqmv_untuk_sesi opsi disetel ke TRUE.

  Anda juga dapat memeriksa apakah tampilan terwujud memenuhi syarat untuk penulisan ulang kueri secara otomatis dengan memeriksa STV_MV_INFO. Untuk informasi selengkapnya, lihat STV_MV_INFO.

Tampilan terwujud pada tabel danau data eksternal di Amazon Redshift Spectrum

Tampilan terwujud dapat memberikan pemeliharaan tambahan pada tabel danau data eksternal. Dengan pemeliharaan inkremental, Amazon Redshift memperbarui data dalam tampilan terwujud hanya dengan perubahan pada data di tabel dasar sejak penyegaran terakhir. Pemeliharaan inkremental lebih hemat biaya daripada menghitung ulang tampilan yang terwujud sepenuhnya setelah setiap perubahan data pada tabel dasar.

Bila Anda menggunakan tampilan terwujud pada setidaknya satu tabel eksternal, pembuatan tampilan terwujud bersifat inkremental pada:

- Tabel danau data standar, dipartisi dan tidak dipartisi, dengan file data dalam format yang didukung (Parquet, Avro, CSV, dan sebagainya).
- Tabel Apache Iceberg, dipartisi dan tidak dipartisi, dengan copy-on-write dan merge-on-read.
- Tabel Amazon Redshift Spectrum bergabung dengan tabel Amazon Redshift apa pun dalam database yang sama.

Refresh tampilan yang terwujud bersifat inkremental pada:

- Tabel danau data standar setelah S3 DELETE atau PUT overwrite (penghapusan file data), jika tampilan yang terwujud tidak melakukan agregasi.
- Tabel Apache Iceberg setelah INSERT, DELETE, UPDATE, atau pemadatan tabel.

Untuk informasi selengkapnya tentang Amazon Redshift Spectrum, lihatSpektrum Pergeseran Merah Amazon.

Batasan

Batasan umum pada tampilan yang terwujud masih berlaku untuk tampilan terwujud pada tabel danau data. Untuk informasi selengkapnya, lihat Menyegarkan tampilan yang terwujud. Selain itu, pertimbangkan batasan berikut saat Anda menggunakan tampilan terwujud pada tabel danau data eksternal.

- Pembuatan tampilan yang terwujud bersifat non-inkremental pada:
  - Meja Hudi atau Danau Delta.

  - Akses data bersarang spektrum.
  - Referensi ke kolom VARBYTE.
- Penyegaran tampilan terwujud kembali ke perhitungan ulang penuh pada:
  - Tabel Apache Iceberg ketika snapshot yang diperlukan kedaluwarsa, jika tampilan yang terwujud melakukan agregasi.
  - Tabel danau data standar setelah penghapusan atau pembaruan file data di Amazon S3, jika tampilan terwujud melakukan agregasi.
  - Tabel danau data standar diperbarui lebih dari sekali dalam blok transaksi.
  - Tabel danau data standar diatur oleh manifes. Untuk informasi selengkapnya tentang manifes, lihat Menggunakan manifes untuk menentukan file data.
  - Amazon Redshift kembali ke komputasi ulang penuh jika ini diharapkan lebih berkinerja, khususnya untuk tampilan terwujud yang berisi gabungan dan lebih dari satu tabel dasar telah diperbarui sejak penyegaran terakhir.
- Pada tabel Apache Iceberg, penyegaran tampilan terwujud hanya dapat menangani hingga 4 juta posisi yang dihapus dalam satu file data. Setelah batas ini tercapai, tabel dasar Apache Iceberg harus dipadatkan untuk terus menyegarkan tampilan yang terwujud.
- Pada tabel Apache Iceberg, penskalaan konkurensi tidak didukung untuk pembuatan dan penyegaran tampilan yang terwujud.
- Fitur otonom tidak didukung. Ini termasuk tampilan terwujud otomatis dan penulisan ulang kueri otomatis.
- Saat tampilan materialisasi inkremental disegarkan, izin IAM hanya berlaku untuk bagian yang diakses dari tabel dasar Amazon Redshift.
- Perubahan izin yang dikelola oleh Lake Formation tidak diverifikasi saat menanyakan tampilan yang terwujud. Ini berarti bahwa jika tampilan terwujud didefinisikan pada tabel danau data dan hak istimewa pilih dihapus dari tabel dengan Formasi Danau, maka Anda masih dapat menanyakan tampilan yang terwujud.

Menyegarkan tampilan yang terwujud

Topik ini menjelaskan cara menyegarkan data dalam tampilan material dari tabel yang mendasarinya.

Saat Anda membuat tampilan terwujud, isinya mencerminkan keadaan relasi database yang mendasarinya (tabel atau tampilan terwujud lainnya) pada saat itu. Data dalam tampilan terwujud tetap tidak berubah, bahkan ketika aplikasi mengubah data dalam tabel yang mendasarinya. Untuk

memperbarui data dalam tampilan terwujud, Anda dapat menggunakan `REFRESH MATERIALIZED VIEW` pernyataan kapan saja untuk menyegarkan tampilan terwujud secara manual. Saat Anda menggunakan pernyataan ini, Amazon Redshift mengidentifikasi perubahan yang telah terjadi dalam relasi dasar dan menerapkan perubahan tersebut ke tampilan yang terwujud.

Amazon Redshift memiliki dua strategi untuk menyegarkan tampilan yang terwujud:

- Dalam banyak kasus, Amazon Redshift dapat melakukan penyegaran tambahan. Dalam penye garan tambahan, Amazon Redshift dengan cepat mengidentifikasi perubahan pada data dalam hubungan dasar sejak penyegaran terakhir dan memperbarui data dalam tampilan terwujud. Refresh inkremental didukung pada konstruksi SQL berikut yang digunakan dalam kueri saat mendefinisikan tampilan terwujud:
  - Konstruksi yang berisi klausa SELECT, FROM, [INNER] JOIN, WHERE, GROUP BY, atau HAVING.
  - Konstruksi yang berisi agregasi, seperti SUM, MIN, MAX, AVG, dan COUNT.
  - Sebagian besar fungsi SQL bawaan, khususnya yang tidak dapat diubah, mengingat ini memiliki argumen input yang sama dan selalu menghasilkan output yang sama.

  Penyegaran inkremental juga didukung untuk tampilan terwujud yang didasarkan pada tabel datashare atau tampilan terwujud.
- Jika penyegaran tambahan tidak memungkinkan, maka Amazon Redshift melakukan penyegaran penuh. Penye garan penuh menjalankan kembali pernyataan SQL yang mendasarinya, menggantikan semua data dalam tampilan yang terwujud.
- Amazon Redshift secara otomatis memilih metode penyegaran untuk tampilan yang terwujud tergantung pada kueri SELECT yang digunakan untuk menentukan tampilan yang terwujud.

Tampilan terwujud bersarang

Tampilan terwujud dapat didefinisikan di atas tampilan terwujud lainnya. Untuk menyegarkan tampilan terwujud seperti itu, Anda harus secara eksplisit menggunakan `CASCADE` kata kunci pada penyegaran tampilan terwujud paling atas. Misalnya, asumsikan struktur tampilan terwujud bersarang berikut:

```
CREATE TABLE t(a INT);
CREATE MATERIALIZED VIEW u AS SELECT * FROM t;
CREATE MATERIALIZED VIEW v AS SELECT * FROM u;
CREATE MATERIALIZED VIEW w AS SELECT * FROM v;
```

```
-- w -> v -> u -> t

INSERT INTO t VALUES (1);
```

Untuk memperbaharui sepenuhnya, Anda memiliki dua pilihan:

- (Direkomendasikan) Refresh w menggunakan `REFRESH MATERIALIZED VIEW w CASCADE` perintah. Perintah ini mengeksekusi refresh semua tampilan yang terwujud dalam satu transaksi.
- Segarkan u, v, dan w sebagai perintah terpisah, dalam urutan ketergantungan (pertama u, lalu v, lalu w).

Jika `CASCADE` kata kunci tidak digunakan secara eksplisit, tampilan yang terwujud akan disegarkan dalam `RESTRICT` mode, hanya menyegarkan tampilan terwujud saat ini. Contoh berikut menunjukkan pesan informasi saat Anda menjalankan tampilan terwujud yang bergantung `REFRESH MATERIALIZED VIEW` pada tampilan terwujud yang sudah ketinggalan zaman.

```
REFRESH MATERIALIZED VIEW w;
INFO:  Materialized view w is already up to date.  However, it depends on another
 materialized view that is not up to date.

REFRESH MATERIALIZED VIEW w CASCADE;
INFO:  Materialized view w was incrementally updated successfully.
```

```
REFRESH MATERIALIZED VIEW v;
INFO: Materialized view v is already up to date. However, it depends on another
 materialized view that is not up to date.

REFRESH MATERIALIZED VIEW v CASCADE;
INFO: Materialized view v was incrementally updated successfully.
```

Dalam contoh di atas dengan opsi penyegaran kaskade, tampilan terwujud u disegarkan terlebih dahulu, tampilan terwujud v disegarkan berikutnya, dan tampilan terwujud w tidak disegarkan.

Contoh berikut menunjukkan bagaimana Anda dapat membuat rencana penyegaran penuh untuk tampilan terwujud secara terprogram. Untuk menyegarkan tampilan terwujud v, pertama-tama segarkan tampilan terwujud u. Untuk menyegarkan tampilan terwujud w, pertama-tama segarkan tampilan terwujud u dan kemudian tampilan terwujud v.

```
WITH RECURSIVE recursive_deps (mv_tgt, lvl, mv_dep) AS
( SELECT trim(name) as mv_tgt, 0 as lvl, trim(ref_name) as mv_dep
  FROM stv_mv_deps
  UNION ALL
  SELECT R.mv_tgt, R.lvl+1 as lvl, trim(S.ref_name) as mv_dep
  FROM stv_mv_deps S, recursive_deps R
  WHERE R.mv_dep = S.name
)

SELECT mv_tgt, mv_dep from recursive_deps
ORDER BY mv_tgt, lvl DESC;

 mv_tgt | mv_dep
--------+--------
 v      | u
 w      | u
 w      | v
(3 rows)
```

Batasan

Amazon Redshift tidak mendukung penyegaran bertingkat untuk tampilan yang terwujud berdasarkan sumber selain:

- Tabel lokal
- Tabel bersama
- MV Lokal
- Streaming MV

Penyegaran bertingkat tidak meluas ke tampilan terwujud bersama.

Amazon Redshift tidak mendukung penyegaran tambahan untuk tampilan terwujud yang ditentukan dengan kueri menggunakan elemen SQL berikut:

- OUTER JOIN (KANAN, KIRI, atau PENUH).
- Setel operasi UNION, INTERSECT, EXCEPT, dan MINUS.
- Fungsi agregat MEDIAN, PERCENTILE_CONT, LISTAGG, STDDEV_SAMP, STDDEV_POP, APPROXIMATE COUNT, APPROXIMATE PERCENTILE, dan fungsi agregat bitwise.

> **Note**
>
> Fungsi agregat COUNT, SUM, dan AVG didukung.

- Fungsi agregat DISTINCT, seperti DISTINCT COUNT, DISTINCT SUM, dan sebagainya.
- Fungsi Jendela.
- Kueri yang menggunakan tabel sementara untuk pengoptimalan kueri, seperti mengoptimalkan subexpressi umum.
- Subkueri.
- Tabel eksternal merujuk format berikut dalam kueri yang mendefinisikan tampilan terwujud.
  - Danau Delta
  - Hudi

  Penyegaran tambahan didukung untuk tampilan terwujud yang ditentukan menggunakan format selain yang tercantum di atas. Untuk informasi selengkapnya, lihat Tampilan terwujud pada tabel danau data eksternal di Amazon Redshift Spectrum.

Menyegarkan tampilan yang terwujud secara otomatis

Amazon Redshift dapat secara otomatis menyegarkan tampilan yang terwujud dengan data terkini dari tabel dasarnya saat tampilan terwujud dibuat dengan atau diubah untuk memiliki opsi penyegaran otomatis. Amazon Redshift memperbarui tampilan yang terwujud sesegera mungkin setelah tabel dasar berubah.

Untuk menyelesaikan penyegaran tampilan terwujud yang paling penting dengan dampak minimal terhadap beban kerja aktif di cluster Anda, Amazon Redshift mempertimbangkan beberapa faktor. Faktor-faktor ini termasuk beban sistem saat ini, sumber daya yang diperlukan untuk penyegaran, sumber daya cluster yang tersedia, dan seberapa sering tampilan terwujud digunakan.

Amazon Redshift memprioritaskan beban kerja Anda daripada autorefresh dan mungkin menghentikan penyegaran otomatis untuk mempertahankan kinerja beban kerja pengguna. Pendekatan ini mungkin menunda penyegaran beberapa pandangan yang terwujud. Dalam beberapa kasus, Anda mungkin memerlukan perilaku penyegaran yang lebih deterministik untuk pandangan yang terwujud. Jika demikian, pertimbangkan untuk menggunakan penyegaran manual seperti yang dijelaskan dalam MENYEGARKAN TAMPILAN YANG TERWUJUD atau penyegaran terjadwal menggunakan operasi API penjadwal Amazon Redshift atau konsol.

Anda dapat mengatur autorefresh untuk tampilan terwujud menggunakan CREATE MATERIALIZED VIEW. Anda juga dapat menggunakan klausa AUTO REFRESH untuk menyegarkan tampilan yang terwujud secara otomatis. Untuk informasi selengkapnya tentang membuat tampilan yang terwujud, lihatBUAT TAMPILAN YANG TERWUJUD. Anda dapat mengaktifkan autorefresh untuk tampilan terwujud saat ini dengan menggunakan. MENGUBAH TAMPILAN YANG TERWUJUD

Pertimbangkan hal berikut saat Anda menyegarkan tampilan yang terwujud:

- Anda masih dapat menyegarkan tampilan terwujud secara eksplisit menggunakan perintah REFRESH MATERIALIZED VIEW meskipun Anda belum mengaktifkan autorefresh untuk tampilan terwujud.
- Penyegaran otomatis didukung pada tampilan terwujud yang ditentukan pada tabel berbagi data atau tabel Iceberg tetapi tidak pada kombinasi keduanya.
- Untuk status penyegaran, Anda dapat memeriksa SVL_MV_REFRESH_STATUS, yang merekam kueri yang dimulai pengguna atau diperbarui secara otomatis.
- Untuk menjalankan recompute REFRESH penuh dari tampilan yang terwujud, pastikan Anda memiliki izin CREATE pada skema. Untuk informasi selengkapnya, lihat HIBAH.

Mulai 27 Februari 2026, kueri Auto REFRESH untuk tampilan terwujud Amazon Redshift dijalankan sebagai kueri pengguna daripada proses otonom latar belakang. Akibatnya, kueri Auto REFRESH sekarang berjalan dengan prioritas yang sama dengan kueri pengguna lainnya.

Perubahan ini meningkatkan kesegaran tampilan yang terwujud dengan Auto REFRESH diaktifkan, membantu mereka tetap up to date dengan perubahan terbaru pada tabel dasar mereka dibandingkan dengan perilaku sebelumnya.

Catatan: Fitur perubahan perilaku MV Auto REFRESH hanya diaktifkan untuk cluster Amazon Redshift Provisioned pada Track CURRENT dari rilis patch P198 dan yang lebih baru. Saat ini dinonaktifkan di Serverless.

Tampilan terwujud otomatis

Topik ini menjelaskan cara Amazon Redshift menggunakan tampilan terwujud otomatis untuk meningkatkan kinerja. Amazon Redshift membuat tampilan yang terwujud secara otomatis berdasarkan aktivitas dan kinerja database. Amazon Redshift menggunakan tampilan terwujud otomatis secara default.

Tampilan terwujud adalah alat yang ampuh untuk meningkatkan kinerja kueri di Amazon Redshift. Mereka melakukan ini dengan menyimpan set hasil yang telah dihitung sebelumnya. Kueri serupa tidak harus menjalankan kembali logika yang sama setiap kali, karena mereka dapat mengambil catatan dari kumpulan hasil yang ada. Pengembang dan analis membuat tampilan yang terwujud setelah menganalisis beban kerja mereka untuk menentukan kueri mana yang akan menguntungkan, dan apakah biaya pemeliharaan setiap tampilan yang terwujud bermanfaat. Ketika beban kerja tumbuh atau berubah, pandangan yang terwujud ini harus ditinjau untuk memastikan mereka terus memberikan manfaat kinerja yang nyata.

Fitur Automated Materialized Views (AutoMV) di Redshift meningkatkan kinerja kueri dengan secara otomatis membuat dan mengelola tampilan terwujud berdasarkan pemantauan beban kerja dan algoritma pembelajaran mesin. Berikut ini termasuk fitur utama AutoMV:

- Pemantauan berkelanjutan — Redshift terus memantau beban kerja menggunakan teknik pembelajaran mesin untuk mengidentifikasi peluang peningkatan kinerja melalui pembuatan tampilan yang terwujud.
- Pembuatan dan penghapusan otomatis — Ketika sistem mendeteksi bahwa tampilan yang terwujud akan bermanfaat, sistem secara otomatis membuat dan memeliharanya. Sebaliknya, jika AutoMV yang dibuat sebelumnya tidak lagi memberikan manfaat kinerja, sistem akan secara otomatis menghapusnya.
- Tidak ada persyaratan aktivitas pengguna — Fitur AutoMV hanya beroperasi selama periode aktivitas pengguna rendah atau beban kerja yang berjalan di cluster. Ini memastikan bahwa operasi AutoMV tidak mengganggu atau memengaruhi beban kerja pelanggan.
- Lonjakan penggunaan CPU — Selama waktu tidak ada aktivitas beban kerja, pembuatan atau penyegaran tampilan yang terwujud oleh AutoMV dapat menyebabkan lonjakan penggunaan CPU. Ini adalah perilaku normal karena sistem menggunakan sumber daya yang tersedia untuk membuat dan menyegarkan tampilan yang terwujud.
- Prioritas beban kerja pengguna — Jika Anda memulai beban kerja saat operasi AutoMV sedang berlangsung, tugas AutoMV akan berhenti untuk melepaskan sumber daya untuk beban kerja pengguna. Ini memastikan bahwa beban kerja Anda diprioritaskan daripada operasi AutoMV.

Meskipun fitur AutoMV dapat menyebabkan lonjakan penggunaan CPU selama periode tidak ada aktivitas pengguna, fitur ini beroperasi secara transparan dan tanpa memengaruhi beban kerja Anda. Sistem mengelola tampilan yang terwujud untuk meningkatkan kinerja kueri dan secara bersamaan memprioritaskan beban kerja pengguna di atas operasi AutoMV.

Perilaku dan kemampuan AutoMV sama dengan tampilan terwujud yang dibuat pengguna. Mereka disegarkan secara otomatis dan bertahap, menggunakan kriteria dan batasan yang sama. Sama seperti tampilan terwujud yang dibuat oleh pengguna, Penulisan ulang kueri otomatis untuk menggunakan tampilan yang terwujud mengidentifikasi kueri yang dapat memperoleh manfaat dari AutoMV yang dibuat sistem. Secara otomatis menulis ulang kueri tersebut untuk menggunakan AutoMV, meningkatkan kinerja kueri. Pengembang tidak perlu merevisi kueri untuk memanfaatkan AutoMV.

> **Note**
>
> Tampilan terwujud otomatis disegarkan sebentar-sebentar. Kueri yang ditulis ulang untuk menggunakan AutoMV selalu mengembalikan hasil terbaru. Ketika Redshift mendeteksi bahwa data tidak diperbarui, kueri tidak ditulis ulang untuk dibaca dari tampilan terwujud otomatis. Sebagai gantinya, kueri memilih data terbaru dari tabel dasar.

Setiap beban kerja dengan kueri yang digunakan berulang kali dapat memperoleh manfaat dari AutoMV. Kasus penggunaan umum meliputi:

- Dasbor - Dasbor banyak digunakan untuk memberikan tampilan cepat indikator bisnis utama (KPI), peristiwa, tren, dan metrik lainnya. Mereka sering memiliki tata letak umum dengan bagan dan tabel, tetapi menunjukkan tampilan yang berbeda untuk pemfilteran, atau untuk operasi pemilihan dimensi, seperti menelusuri. Dasbor sering memiliki kumpulan kueri umum yang digunakan berulang kali dengan parameter yang berbeda. Kueri dasbor dapat memperoleh manfaat besar dari tampilan terwujud otomatis.
- Laporan - Kueri pelaporan dapat dijadwalkan pada berbagai frekuensi, berdasarkan persyaratan bisnis dan jenis laporan. Selain itu, mereka dapat otomatis atau sesuai permintaan. Karakteristik umum dari kueri pelaporan adalah bahwa mereka dapat berjalan lama dan padat sumber daya. Dengan AutoMV, kueri ini tidak perlu dihitung ulang setiap kali dijalankan, yang mengurangi runtime untuk setiap kueri dan pemanfaatan sumber daya di Redshift.

Untuk mematikan tampilan terwujud otomatis, Anda memperbarui grup `auto_mv` parameter `kefalse`. Untuk informasi selengkapnya, lihat grup parameter Amazon Redshift di Panduan Manajemen Cluster Amazon Redshift.

Lingkup SQL dan pertimbangan untuk tampilan terwujud otomatis

- Tampilan terwujud otomatis dapat dimulai dan dibuat oleh kueri atau subkueri, asalkan berisi `GROUP BY` klausa atau salah satu fungsi agregat berikut: SUM, COUNT, MIN, MAX atau AVG. Tetapi itu tidak dapat mengandung salah satu dari berikut ini:
  - Gabungan luar kiri, kanan, atau penuh
  - Fungsi agregat selain SUM, COUNT, MIN, MAX, dan AVG. (Fungsi khusus ini bekerja dengan penulisan ulang kueri otomatis.)
  - Setiap fungsi agregat yang mencakup DISTINCT
  - Fungsi jendela apa pun
  - PILIH klausa DISTINCT atau MEMILIKI
  - Pandangan terwujud lainnya

  Tidak dijamin bahwa kueri yang memenuhi kriteria akan memulai pembuatan tampilan terwujud otomatis. Sistem menentukan dari kandidat mana untuk membuat tampilan, berdasarkan manfaat yang diharapkan untuk beban kerja dan biaya sumber daya untuk dipelihara, yang mencakup biaya untuk sistem untuk menyegarkan. Setiap tampilan terwujud yang dihasilkan dapat digunakan dengan penulisan ulang kueri otomatis.
- Meskipun AutoMV mungkin diprakarsai oleh subkueri atau kaki individu dari operator set, tampilan terwujud yang dihasilkan tidak akan berisi subkueri atau operator set.
- Untuk menentukan apakah AutoMV digunakan untuk kueri, lihat rencana EXPLAIN dan cari `%_auto_mv_%` di output. Untuk informasi lebih lanjut, lihat JELASKAN.
- Tampilan terwujud otomatis tidak didukung pada tabel eksternal, seperti berbagi data dan tabel federasi.

Batasan tampilan terwujud otomatis

Berikut ini adalah batasan untuk bekerja dengan tampilan terwujud otomatis:

- Jumlah maksimum AutoMV - Batas tampilan terwujud otomatis adalah 200 per database di cluster.
- Ruang dan kapasitas penyimpanan - Karakteristik penting AutoMV adalah bahwa hal itu dilakukan menggunakan siklus latar belakang cadangan untuk membantu mencapai bahwa beban kerja pengguna tidak terpengaruh. Jika cluster sibuk atau kehabisan ruang penyimpanan, AutoMV menghentikan aktivitasnya. Secara khusus, pada 80% dari total kapasitas cluster, tidak ada tampilan terwujud otomatis baru yang dibuat. Pada 90% dari total

kapasitas, mereka dapat dijatuhkan untuk memfasilitasi beban kerja pengguna berlanjut tanpa penurunan kinerja. Untuk informasi selengkapnya tentang menentukan kapasitas cluster, lihatSTV_NODE_STORAGE_KAPASITAS.

Penagihan untuk tampilan terwujud otomatis

Kemampuan pengoptimalan otomatis Amazon Redshift menciptakan dan menyegarkan tampilan terwujud otomatis. Tidak ada biaya untuk sumber daya komputasi untuk proses ini. Penyimpanan tampilan terwujud otomatis dibebankan pada tarif reguler untuk penyimpanan. Untuk informasi selengkapnya, lihat harga https://aws.amazon.com/redshift/pricing/ Amazon Redshift.

Sumber daya tambahan

Posting blog berikut memberikan penjelasan lebih lanjut mengenai tampilan terwujud otomatis. Ini merinci bagaimana mereka dibuat, dipelihara, dan dijatuhkan. Ini juga menjelaskan algoritma mendasar yang mendorong keputusan ini: Optimalkan kinerja kueri Amazon Redshift Anda dengan tampilan terwujud otomatis.

Video ini dimulai dengan penjelasan tentang tampilan yang terwujud dan menunjukkan bagaimana mereka meningkatkan kinerja dan menghemat sumber daya. Kemudian memberikan penjelasan mendalam tentang tampilan terwujud otomatis dengan animasi aliran proses dan demonstrasi langsung.

Menggunakan fungsi yang ditentukan pengguna (UDF) dalam tampilan terwujud

Anda dapat menggunakan UDF skalar dalam tampilan terwujud Amazon Redshift. Tentukan ini baik dalam python atau SQL dan referensikan mereka dalam definisi tampilan yang terwujud.

Mereferensikan UDF dalam tampilan terwujud

Prosedur berikut menunjukkan cara menggunakan UDF yang melakukan perbandingan aritmatika sederhana, dalam definisi tampilan terwujud.

1. Buat tabel untuk digunakan dalam definisi tampilan terwujud.

```
CREATE TABLE base_table (a int, b int);
```

2. Buat fungsi skalar yang ditentukan pengguna di python yang mengembalikan nilai boolean yang menunjukkan apakah bilangan bulat lebih besar dari bilangan bulat perbandingan.

```
CREATE OR REPLACE FUNCTION udf_python_bool(x1 int, x2 int) RETURNS bool IMMUTABLE
AS $$
  return x1 > x2
$$ LANGUAGE plpythonu;
```

   Secara opsional, buat UDF yang mirip secara fungsional dengan SQL, yang dapat Anda gunakan untuk membandingkan hasil dengan yang pertama.

```
CREATE OR REPLACE FUNCTION udf_sql_bool(int, int) RETURNS bool IMMUTABLE
AS $$
  select $1 > $2;
$$ LANGUAGE SQL;
```

3. Buat tampilan terwujud yang memilih dari tabel yang Anda buat dan mereferensikan UDF.

```
CREATE MATERIALIZED VIEW mv_python_udf AS SELECT udf_python_bool(a, b) AS a FROM
 base_table;
```

   Secara opsional, Anda dapat membuat tampilan terwujud yang mereferensikan SQL UDF.

```
CREATE MATERIALIZED VIEW mv_sql_udf AS SELECT udf_sql_bool(a, b) AS a FROM
 base_table;
```

4. Tambahkan data ke tabel dan segarkan tampilan yang terwujud.

```
INSERT INTO base_table VALUES (1,2), (1,3), (4,2);
```

```
REFRESH MATERIALIZED VIEW mv_python_udf;
```

   Secara opsional, Anda dapat menyegarkan tampilan terwujud yang mereferensikan SQL UDF.

```
REFRESH MATERIALIZED VIEW mv_sql_udf;
```

5. Kueri data dari tampilan terwujud Anda.

```
SELECT * FROM mv_python_udf ORDER BY a;
```

Hasil kueri adalah sebagai berikut:

```
a
-----
false
false
true
```

Ini mengembalikan `true` untuk set nilai terakhir karena nilai untuk kolom a (4) lebih besar dari nilai untuk kolom b (2).

6. Secara opsional, Anda dapat menanyakan tampilan terwujud yang mereferensikan SQL UDF. Hasil untuk fungsi SQL cocok dengan hasil dari versi Python.

```
SELECT * FROM mv_sql_udf ORDER BY a;
```

Hasil kueri adalah sebagai berikut:

```
a
-----
false
false
true
```

Ini mengembalikan `true` set nilai terakhir untuk dibandingkan.

7. Gunakan pernyataan DROP dengan CASCADE untuk melepaskan fungsi yang ditentukan pengguna dan tampilan terwujud yang mereferensikannya.

```
DROP FUNCTION udf_python_bool(int, int) CASCADE;
```

```
DROP FUNCTION udf_sql_bool(int, int) CASCADE;
```

Streaming konsumsi ke tampilan yang terwujud

Topik ini menjelaskan cara menggunakan tampilan terwujud untuk akses cepat ke data streaming.

Penyerapan streaming menyediakan penyerapan data berkecepatan tinggi dengan latensi rendah dari Amazon Kinesis Data Streams atau Amazon Managed Streaming untuk Apache Kafka ke database Amazon Redshift yang disediakan atau Amazon Redshift Serverless. Data mendarat di tampilan Redshift yang terwujud yang dikonfigurasi untuk tujuan tersebut. Ini menghasilkan akses cepat ke data eksternal. Konsumsi streaming mengurangi waktu akses data dan mengurangi biaya penyimpanan. Anda dapat mengonfigurasinya untuk cluster Amazon Redshift atau untuk grup kerja Amazon Redshift Serverless Anda, menggunakan kumpulan kecil perintah SQL. Setelah diatur, setiap penyegaran tampilan terwujud dapat menyerap ratusan megabyte data per detik.

Bagaimana data mengalir dari layanan streaming ke Redshift

Ini membantu untuk memahami bagaimana penyerapan streaming bekerja dan objek database yang digunakan dalam proses tersebut. Data mengalir langsung dari penyedia aliran data ke cluster yang disediakan Amazon Redshift atau ke grup kerja Amazon Redshift Serverless. Tidak ada area pendaratan sementara, seperti bucket Amazon S3. Cluster atau workgroup yang disediakan adalah konsumen aliran. Dalam database Redshift, data yang dibaca dari aliran mendarat dalam tampilan yang terwujud. Data diproses saat tiba. Misalnya, nilai JSON dapat dikonsumsi dan dipetakan ke kolom data tampilan yang terwujud, menggunakan SQL. Saat tampilan terwujud disegarkan, Redshift mengkonsumsi data dari pecahan data Kinesis yang dialokasikan atau partisi Kafka hingga tampilan diperbarui dengan aliran.

Kasus penggunaan untuk konsumsi streaming Amazon Redshift melibatkan data yang dihasilkan terus menerus dan harus diproses dalam waktu singkat, atau latensi, sejak asalnya. Ini biasa disebut anal itik hampir real-time. Sumber dapat mencakup perangkat TI, perangkat telemetri sistem, dan data aliran klik dari situs web atau aplikasi yang sibuk.

Praktik terbaik penguraian data untuk meningkatkan kinerja

Saat Anda mengonfigurasi konsumsi streaming, ada opsi dalam cara Anda dapat mengurai data yang masuk. Praktik dapat mencakup melakukan logika bisnis atau pemformatan saat data tiba. Kami merekomendasikan praktik terbaik berikut untuk menghindari kesalahan atau kehilangan data. Ini berasal dari pengujian internal dan membantu pelanggan mengatasi masalah konfigurasi dan penguraian.

- Mengekstrak nilai dari data streaming — Jika Anda menggunakan fungsi JSON_EXTRACT_PATH_TEXT dalam definisi tampilan terwujud Anda untuk mengurai atau menghancurkan JSON yang dialirkan, hal itu dapat berdampak signifikan pada kinerja dan latensi. Untuk menjelaskan, untuk setiap kolom yang diekstraksi menggunakan

JSON_EXTRACT_PATH_TEXT, JSON yang masuk diurai ulang. Setelah ini, konversi tipe data, pemfilteran, dan perhitungan logika bisnis terjadi. Ini berarti, misalnya, bahwa jika Anda mengekstrak 10 kolom dari data JSON, setiap catatan JSON diurai 10 kali, yang mencakup logika tambahan. Ini menghasilkan latensi konsumsi yang lebih tinggi. Pendekatan alternatif yang kami sarankan adalah menggunakan fungsi JSON_PARSE untuk mengonversi catatan JSON ke tipe data SUPER Redshift. Setelah data streaming mendarat di tampilan terwujud, gunakan PartiQL untuk mengekstrak string individu dari representasi SUPER dari data JSON. Untuk informasi selengkapnya, lihat Mengku eri data semi-terstruktur.

Selain itu, perhatikan bahwa JSON_EXTRACT_PATH_TEXT memiliki ukuran data maksimum 16MB. Jadi, jika ada catatan JSON yang lebih besar dari 16MB, memprosesnya dengan JSON_EXTRACT_PATH_TEXT menghasilkan kesalahan.

- Memetakan Amazon Kinesis Data Streams aliran atau topik Amazon MSK ke beberapa tampilan yang terwujud — Kami tidak menyarankan untuk membuat beberapa tampilan terwujud untuk menyerap data dari satu aliran atau topik. Ini karena setiap tampilan yang terwujud menciptakan konsumen untuk setiap pecahan dalam aliran atau partisi Aliran Data Kinesis dalam topik Kafka. Hal ini dapat mengakibatkan pembatasan atau melebihi throughput aliran atau topik. Ini juga dapat menghasilkan biaya yang lebih tinggi, karena Anda menelan data yang sama beberapa kali. Saat mengonfigurasi penyerapan streaming, sebaiknya buat satu tampilan terwujud untuk setiap aliran atau topik.

  Jika kasus penggunaan mengharuskan Anda menyerap data dari satu aliran KDS atau topik MSK ke dalam beberapa tampilan yang terwujud, lihat blog AWS Big Data, khususnya Praktik terbaik untuk menerapkan analitik hampir real-time menggunakan Amazon Redshift Streaming Ingestion dengan Amazon M SK, sebelum Anda melakukannya.

Perilaku penyerapan streaming dan tipe data

Tabel berikut menjelaskan detail perilaku teknis dan batas ukuran untuk berbagai jenis data. Sebaiknya Anda terbiasa dengan ini sebelum mengonfigurasi tampilan terwujud untuk konsumsi streaming.

Fitur atau perilaku	Deskripsi
Batas panjang topik Kafka	Tidak mungkin menggunakan topik Kafka dengan nama lebih dari 128 karakter (tidak termasuk tanda kutip). Untuk informasi selengkapnya, lihat Nama dan pengidentifikasi.

Fitur atau perilaku	Deskripsi
Penyegaran inkremental dan JOIN pada tampilan terwujud	Pandangan yang terwujud harus dapat dipertahankan secara bertahap. Komputasi ulang penuh tidak dimungkinkan untuk Kinesis atau Amazon MSK karena mereka tidak menyimpan riwayat aliran atau topik selama 24 jam atau 7 hari, secara default. Anda dapat mengatur periode penyimpanan data yang lebih lama di Kinesis atau Amazon MSK. Namun, ini dapat menghasilkan lebih banyak perawatan dan biaya. Selain itu, JOIN saat ini tidak didukung pada tampilan terwujud yang dibuat di aliran Kinesis, atau pada topik Amazon MSK. Setelah membuat tampilan terwujud pada aliran atau topik Anda, Anda dapat membuat tampilan terwujud lain untuk menggabungkan tampilan terwujud streaming Anda ke tampilan, tabel, atau tampilan terwujud lainnya. Untuk informasi selengkapnya, lihat REFRESH MATERIALIZED VIEW.
Penguraian rekaman	Konsumsi streaming Amazon Redshift tidak mendukung penguraian catatan yang telah dikumpulkan oleh Kinesis Producer Library (KPL Key Concepts - Aggregation). Catatan agregat dicerna, tetapi disimpan sebagai data buffer protokol biner. (Lihat Protokol buffer untuk informasi lebih lanjut.) Tergantung pada bagaimana Anda mendorong data ke Kinesis, Anda mungkin perlu mematikan fitur ini.
Nilai duplikat dalam header Kafka	Klien konsumen streaming Amazon Redshift untuk topik Kafka yang bersumber dari Amazon MSK, Confluent, atau Apache Kafka tidak mendukung nilai duplikat di header topik Kafka.
Dekompresi	`VARBYTE`tidak mendukung dekompresi. Karena itu, catatan yang berisi data terkompresi tidak dapat ditanyakan di Redshift. Dekompres data Anda sebelum menambahkannya ke aliran Kinesis atau topik Amazon MSK.

Fitur atau perilaku	Deskripsi
Ukuran rekaman maksimum	Ukuran maksimum rekaman apa pun yang dapat diambil Amazon Redshift dari layanan streaming adalah 16.777.216 byte (16 MiB), ukuran maksimum yang didukung oleh tipe data VARBYTE di Amazon Redshift. Streaming tampilan terwujud yang dibuat pada topik Amazon MSK atau aliran data Kinesis mengatur ukuran kolom data VARBYTE menjadi 16 MiB dan dapat menggunakan catatan hingga ukuran tersebut. Namun, ketika mengkonsumsi dari aliran data Kinesis, ukuran rekaman maksimum yang didukung Kinesis saat ini adalah 10.485.760 byte (10 MiB). Dukungan untuk batas ukuran rekaman Kinesis 10 MiB memerlukan patch 203 atau yang lebih baru. Tampilan terwujud yang dibuat pada aliran data Kinesis pada patch sebelumnya memiliki kolom data VARBYTE yang disetel ke 1.048.576 byte (1 MiB) dan terus mencerna catatan hanya hingga 1 MiB. Untuk menelan catatan yang lebih besar, buat kembali tampilan yang terwujud pada patch 203 atau yang lebih baru. Untuk informasi selengkapnya tentang batas ukuran Kinesis, lihat Ku ota dan batas dan Men angani catatan besar di Panduan Pengembang Amazon Kinesis Data Streams.
Catatan kesalahan	Dalam setiap kasus di mana catatan tidak dapat dicerna ke Redshift karena ukuran data melebihi maksimum, catatan itu dilewati. Penyegaran tampilan terwujud masih berhasil, dalam hal ini, dan segmen dari setiap catatan kesalahan ditulis ke tabel SYS_STREAM_SCAN_ERROR sistem. Kesalahan yang dihasilkan dari logika bisnis, seperti kesalahan dalam perhitung an atau kesalahan yang dihasilkan dari konversi tipe, tidak dilewati. Uji logika dengan hati-hati sebelum Anda menambahk annya ke definisi tampilan yang terwujud.

Fitur atau perilaku	Deskripsi
Konektivitas Multi-VPC pribadi Amazon MSK	Konektivitas pribadi Amazon MSK Multi-VPC saat ini tidak didukung untuk konsumsi streaming Redshift. Atau, Anda dapat menggunakan peering VPC untuk menghubungkan VPC atau AWS Transit Gateway untuk menghubungkan VPC dan jaringan lokal melalui hub pusat. Salah satu dari ini dapat mengaktifkan Redshift untuk berkomunikasi dengan cluster Amazon MSK atau dengan Amazon MSK Serverless di VPC lain.
Penggunaan dan aktivasi penyegaran otomatis	Kueri penyegaran otomatis untuk tampilan atau tampilan yang terwujud diperlakukan sebagai beban kerja pengguna lainnya. Penyegaran otomatis memuat data dari aliran saat tiba. Penyegaran otomatis dapat diaktifkan secara eksplisit untuk tampilan terwujud yang dibuat untuk konsumsi streaming. Untuk melakukan ini, tentukan `AUTO REFRESH` dalam definisi tampilan terwujud. Refresh manual adalah default. Untuk menentukan penyegaran otomatis untuk tampilan terwujud yang ada untuk penyerapan streaming, Anda dapat menjalan `ALTER MATERIALIZED VIEW` kannya untuk menyalakannya. Untuk informasi selengkapnya, lihat MEMBUAT TAMPILAN TERMATERIALISASI atau MENGUBAH TAMPILAN TERMATERIALISASI.
Penyerapan streaming dan Amazon Redshift Tanpa Server	Petunjuk penyiapan dan konfigurasi yang berlaku untuk penyerapan streaming Amazon Redshift pada cluster yang disediakan juga berlaku untuk penyerapan streaming di Amazon Redshift Serverless. Penting untuk menentukan tingkat RPU yang diperlukan untuk mendukung penyerapan streaming dengan penyegaran otomatis dan beban kerja lainnya. Untuk informasi selengkapnya, lihat Pen agihan untuk Amazon Redshift Server less.

Fitur atau perilaku	Deskripsi
Node Amazon Redshift di zona ketersediaan yang berbeda dari cluster Amazon MSK	Saat Anda mengonfigurasi penyerapan streaming, Amazon Redshift mencoba menyambung ke cluster Amazon MSK di zona ketersediaan yang sama, jika kesadaran rak diaktifkan untuk Amazon MSK. Jika semua node berada di zona ketersedi aan yang berbeda dari cluster Amazon Redshift, Anda dapat dikenakan biaya transfer data zona lintas-ketersediaan. Untuk menghindari hal ini, simpan setidaknya satu node cluster broker Amazon MSK di AZ yang sama dengan cluster atau grup kerja yang disediakan Redshift Anda.
Segarkan lokasi awal	Setelah membuat tampilan yang terwujud, penyegaran awalnya dimulai `TRIM_HORIZON` dari aliran Kinesis, atau dari offset 0 dari topik Amazon MSK.
Format data	Format data yang didukung terbatas pada format yang dapat dikonversi`VARBYTE`. Untuk informasi selengkapnya, lihat Jenis VARBYTE dan Operator VARBYTE.
Menambahkan catatan ke tabel	Anda dapat menjalankan `ALTER TABLE APPEND` untuk menambahkan baris ke tabel target dari tampilan terwujud sumber yang ada. Ini hanya berfungsi jika tampilan terwujud dikonfigurasi untuk konsumsi streaming. Untuk informasi selengkapnya, lihat ALTER TABLE APPEND. `ALTER TABLE APPEND`Operasi menyimpan kunci eksklusif saat dijalankan di Amazon Redshift streaming tampilan terwujud yang terhubung ke salah satu dari berikut ini: • Aliran data Amazon Kinesis • Topik Streaming Terkelola Amazon untuk Apache Kafka • Aliran eksternal yang didukung, seperti topik Confluent Cloud Kafka

Fitur atau perilaku	Deskripsi
Menjalankan TRUNCATE atau DELETE	Anda dapat menghapus catatan dari tampilan terwujud yang digunakan untuk penyerapan streaming, menggunakan cara berikut: • `TRUNCATE`— Ini menghapus semua baris dari tampilan terwujud yang dikonfigurasi untuk penyerapan streaming. Itu tidak melakukan pemindaian tabel. Untuk informasi selengkapnya, lihat MEM OTONG. • `DELETE`— Ini menghapus semua baris dari tampilan terwujud yang dikonfigurasi untuk penyerapan streaming. Untuk informasi selengkapnya, lihat HAPUS. `TRUNCATE`dan `DELETE` operasi memiliki kunci eksklusif saat dijalankan di Amazon Redshift streaming tampilan terwujud yang terhubung ke salah satu dari berikut ini: • Aliran data Amazon Kinesis • Topik Streaming Terkelola Amazon untuk Apache Kafka • Aliran eksternal yang didukung, seperti topik Confluent Cloud Kafka

Fitur atau perilaku	Deskripsi
Non-lowercase pengident ifikasi	Saat membuat tampilan terwujud streaming di Amazon Managed Streaming untuk topik Apache Kafka atau Aliran Data Kinesis yang berisi pengidentifikasi non-huruf kecil, penyegaran otomatis mungkin gagal. Untuk mengatasi masalah ini, lakukan salah satu hal berikut: • Gunakan penyegaran manual dengan `enable_ca se_sensitive_identifier` parameter disetel ke `true` • Aktifkan penyegaran otomatis dengan meny `enable_ca se_sensitive_identifier` et `true` el ke tingkat database atau cluster **Note** Pengaturan `enable_case_sensitive_ident ifier` di tingkat pengguna tidak cukup untuk penyegaran otomatis, tetapi akan berfungsi untuk penyegaran manual. Untuk informasi selengkapnya tentang pengidentifikasi peka huruf besar/kecil, lihatenable_case_sensitive_identifier.
Idempotensi	Amazon Redshift menjamin bahwa setiap rekaman diproses tepat sekali saat menelan data dari sumber streaming. Jaminan ini berlaku untuk dua jenis sumber: Amazon Kinesis (mengguna kan pengidentifikasi aliran, pecahan, dan nomor urut) dan Apache Kafka (menggunakan pengidentifikasi topik, partisi, dan offset), termasuk Amazon Managed Streaming untuk Apache Kafka (Amazon MSK) dan Confluent Cloud.

Memulai penyerapan streaming dari Amazon Kinesis Data Streams

Topik ini menjelaskan cara menggunakan data streaming dari Kinesis Data Streams menggunakan tampilan yang terwujud.

Menyiapkan penyerapan streaming Amazon Redshift melibatkan pembuatan skema eksternal yang dipetakan ke sumber data streaming dan membuat tampilan terwujud yang mereferensikan skema eksternal. Penyerapan streaming Amazon Redshift mendukung Kinesis Data Streams sebagai sumbernya. Dengan demikian, Anda harus memiliki sumber Kinesis Data Streams yang tersedia sebelum mengonfigurasi penyerapan streaming. Jika Anda tidak memiliki sumber, ikuti petunjuk dalam dokumentasi Kinesis di Mem ulai dengan Amazon Kinesis Data Streams atau buat satu di konsol menggunakan petunjuk di Membuat Stream melalui Konsol AWS Manajemen.

Penyerapan streaming Amazon Redshift menggunakan tampilan terwujud, yang diperbarui langsung dari aliran saat `REFRESH` dijalankan. Tampilan terwujud memetakan ke sumber data aliran. Anda dapat melakukan pemfilteran dan agregasi pada data aliran sebagai bagian dari definisi tampilan terwujud. Tampilan terwujud penyerapan streaming Anda (tampilan ter wujud dasar) hanya dapat mereferensikan satu aliran, tetapi Anda dapat membuat tampilan terwujud tambahan yang bergabung dengan tampilan terwujud dasar dan dengan tampilan atau tabel terwujud lainnya.

> **Note**
>
> Penyerapan streaming dan Amazon Redshift Serverless - Langkah-langkah konfigurasi dalam topik ini berlaku untuk cluster Amazon Redshift yang disediakan dan ke Amazon Redshift Serverless. Untuk informasi selengkapnya, lihat Perilaku penyerapan streaming dan tipe data.

Dengan asumsi Anda memiliki aliran Kinesis Data Streams yang tersedia, langkah pertama adalah menentukan skema di Amazon Redshift dengan `CREATE EXTERNAL SCHEMA` dan mereferensikan sumber daya Kinesis Data Streams. Setelah itu, untuk mengakses data dalam aliran, tentukan `STREAM` dalam tampilan yang terwujud. Anda dapat menyimpan catatan aliran dalam `SUPER` format semi-terstruktur, atau menentukan skema yang menghasilkan data yang dikonversi ke tipe data Redshift. Saat Anda menanyakan tampilan yang terwujud, catatan yang dikembalikan adalah tampilan point-in-time dari aliran.

1. Buat peran IAM dengan kebijakan kepercayaan yang memungkinkan cluster Amazon Redshift atau grup kerja Amazon Redshift Serverless Anda mengambil peran tersebut. Untuk informasi

tentang cara mengonfigurasi kebijakan kepercayaan untuk peran IAM, lihat Mengotor isasi Amazon Redshift untuk mengakses AWS layanan lain atas nama Anda. Setelah dibuat, peran harus memiliki kebijakan IAM berikut, yang memberikan izin untuk komunikasi dengan aliran data Amazon Kinesis.

Kebijakan IAM untuk aliran yang tidak dienkripsi dari Kinesis Data Streams

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards",
                "kinesis:DescribeStream"
            ],
            "Resource": "arn:aws:kinesis:*:111122223333:stream/*"
        },
        {
            "Sid": "ListStream",
            "Effect": "Allow",
            "Action": "kinesis:ListStreams",
            "Resource": "*"
        }
    ]
}
```

Kebijakan IAM untuk aliran terenkripsi dari Kinesis Data Streams

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "ReadStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards",
                "kinesis:DescribeStream"
            ],
            "Resource": "arn:aws:kinesis:*:111122223333:stream/*"
        },
        {
            "Sid": "DecryptStream",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "Sid": "ListStream",
            "Effect": "Allow",
            "Action": "kinesis:ListStreams",
            "Resource": "*"
        }
    ]
}
```

2. Periksa VPC Anda dan verifikasi bahwa cluster Amazon Redshift atau Amazon Redshift Serverless Anda memiliki rute untuk mencapai titik akhir Kinesis Data Streams melalui internet menggunakan gateway NAT atau gateway internet. Jika Anda ingin lalu lintas antara Redshift dan Kinesis Data Streams tetap berada di dalam AWS jaringan, pertimbangkan untuk menggunakan Titik Akhir VPC Antarmuka Kinesis. Untuk informasi selengkapnya, lihat Menggunakan Amazon Kinesis Data Streams Kinesis Data Streams dengan Titik Akhir Antarmuka VPC.

3. Di Amazon Redshift, buat skema eksternal untuk memetakan data dari Kinesis ke skema.

```
CREATE EXTERNAL SCHEMA kds
FROM KINESIS
```

```
IAM_ROLE { default | 'iam-role-arn' };
```

Penyerapan streaming untuk Kinesis Data Streams tidak memerlukan jenis otentikasi. Ini menggunakan peran IAM yang ditentukan dalam `CREATE EXTERNAL SCHEMA` pernyataan untuk membuat permintaan Kinesis Data Streams.

Opsional: Gunakan kata kunci REGION untuk menentukan Wilayah tempat aliran Amazon Kinesis Data Streams atau Amazon MSK berada.

```
CREATE EXTERNAL SCHEMA kds
FROM KINESIS
REGION 'us-west-2'
IAM_ROLE { default | 'iam-role-arn' };
```

Dalam sampel ini, Wilayah menentukan lokasi aliran sumber. IAM_ROLE adalah contoh.

4. Buat tampilan terwujud untuk menggunakan data aliran. Dengan pernyataan seperti berikut, jika catatan tidak dapat diurai, itu menyebabkan kesalahan. Gunakan perintah seperti ini jika Anda tidak ingin catatan kesalahan dilewati.

```
CREATE MATERIALIZED VIEW my_view AUTO REFRESH YES AS
SELECT *
FROM kds.my_stream_name;
```

Nama aliran Kinesis peka huruf besar dan dapat berisi huruf besar dan kecil. Untuk mengambil dari aliran dengan nama huruf besar, Anda dapat mengatur konfigurasi `enable_case_sensitive_identifier` ke `true` tingkat database. Untuk informasi selengkapnya, lihat Nama dan pengidentifikasi dan enable _case_sensitive_identifier.

Untuk mengaktifkan penyegaran otomatis, gunakan`AUTO REFRESH YES`. Perilaku defaultnya adalah penyegaran manual. Perhatikan saat Anda menggunakan CAN_JSON_PARSE, ada kemungkinan bahwa catatan yang tidak dapat diurai dilewati.

Kolom metadata meliputi:

Kolom metadata	Jenis data	Deskripsi
perkiraan timestamp kedatangan_kedatangan_	stempel waktu tanpa zona waktu	Perkiraan waktu rekaman dimasukkan ke dalam aliran Kinesis
partition_key	bangsawan (256)	Kunci yang digunakan oleh Kinesis untuk menetapkan catatan ke pecahan
shard_id	arang (20)	Pengidentifikasi unik pecahan di dalam aliran dari mana catatan diambil
urut_nomor_	bangkai (128)	Pengidentifikasi unik rekaman dari pecahan Kinesis
Refresh_time	stempel waktu tanpa zona waktu	Waktu penyegaran dimulai
kinesis_data	warbyte	Rekaman dari aliran Kinesis

Penting untuk dicatat jika Anda memiliki logika bisnis dalam definisi tampilan terwujud bahwa kesalahan logika bisnis dapat menyebabkan konsumsi streaming diblokir dalam beberapa kasus. Ini mungkin menyebabkan Anda harus melepaskan dan membuat kembali tampilan yang terwujud. Untuk menghindari hal ini, kami menyarankan agar Anda menjaga logika Anda sesederhana mungkin dan melakukan sebagian besar pemeriksaan logika bisnis Anda pada data setelah dicerna.

5. Segarkan tampilan, yang memanggil Redshift untuk membaca dari aliran dan memuat data ke tampilan yang terwujud.

```
REFRESH MATERIALIZED VIEW my_view;
```

6. Data kueri dalam tampilan terwujud.

```
select * from my_view;
```

Memulai dengan konsumsi streaming dari sumber Apache Kafka

Topik ini menjelaskan cara menggunakan data streaming dari Amazon MSK, Apache Kafka, atau Confluent Cloud menggunakan tampilan yang terwujud.

Tujuan dari penyerapan streaming Amazon Redshift adalah untuk menyederhanakan proses untuk langsung menelan data aliran dari layanan streaming ke Amazon Redshift atau Amazon Redshift Serverless. Ini bekerja dengan Amazon MSK Provisioned dan Amazon MSK Serverless, dengan sumber terbuka Apache Kafka, dan dengan Confluent Cloud. Penyerapan streaming Amazon Redshift menghilangkan kebutuhan untuk menampilkan topik Apache Kafka di Amazon S3 sebelum menelan data aliran ke Redshift.

Pada tingkat teknis, penyerapan streaming menyediakan penyerapan data aliran atau topik dengan latensi rendah dan berkecepatan tinggi ke dalam tampilan terwujud Amazon Redshift. Setelah pengaturan, menggunakan penyegaran tampilan terwujud, Anda dapat mengambil volume data yang besar.

Anda harus memiliki sumber Apache Kafka yang tersedia sebelum mengonfigurasi konsumsi streaming Amazon Redshift. Jika Anda tidak memiliki sumber, buat satu menggunakan petunjuk berikut:

- Amazon MSK — Mem ulai Menggunakan Amazon MSK
- Apache Kafka — Memulai Cepat Apache Kafka
- Confluent Cloud — Mulai C epat untuk Confluent Cloud

Menyiapkan konsumsi streaming dari Kafka

Gunakan prosedur berikut untuk mengatur penyerapan streaming ke Amazon Redshift dari Amazon MSK, atau sumber Apache Kafka yang tidak AWS dikelola (Apache Kafka dan Confluent Cloud).

Topik

- Siapkan otentikasi
- Menyiapkan VPC Anda
- Buat Tampilan Terwujud

Siapkan otentikasi

Bagian ini menjelaskan pengaturan otentikasi untuk memungkinkan aplikasi Amazon Redshift Anda mengakses sumber Amazon MSK.

Setelah Anda membuat peran aplikasi, lampirkan salah satu kebijakan berikut untuk mengizinkan akses ke cluster Amazon MSK, Apache Kafka, atau Confluent Cloud. Untuk otentikasi MTL, Anda dapat menyimpan sertifikat yang digunakan Amazon Redshift di ACM atau Secrets Manager, jadi Anda harus memilih kebijakan yang cocok dengan tempat penyimpanan sertifikat.

Perhatikan bahwa sertifikat yang ditandatangani sendiri tidak didukung untuk otentikasi atau data dalam perjalanan saat Anda menggunakan penyerapan streaming langsung ke Amazon Redshift dengan salah satu sumber streaming Apache Kafka yang didukung. Ini termasuk Amazon MSK, Apache Kafka, dan Confluent Cloud. Pertimbangkan untuk menggunakan sertifikat yang dibuat dengan AWS Certificate Manager atau otoritas sertifikat tepercaya publik lainnya.

Otentikasi Amazon Redshift IAM dengan MSK hanya didukung pada Kafka versi 2.7.1 atau lebih tinggi.

OTENTIKASI IAM (hanya Amazon MSK):

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MSKIAMpolicy",
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeTopic",
                "kafka-cluster:Connect"
            ],
            "Resource": [
                "arn:aws:kafka:*:111122223333:cluster/MyTestCluster/*",
                "arn:aws:kafka:*:111122223333:topic/MyTestCluster/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
                "arn:aws:kafka:*:111122223333:group/MyTestCluster/*"
            ]
        }
    ]
}
```

OTENTIKASI MTLS: menggunakan sertifikat yang disimpan di AWS Certificate Manager

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MSKmTLSACMpolicy",
            "Effect": "Allow",
            "Action": [
                "acm:ExportCertificate"
            ],
            "Resource": [
                "arn:aws:acm:us-east-1:444455556666:certificate/certificate_ID"
            ]
        }
    ]
}
```

OTENTIKASI MTLS: menggunakan sertifikat yang disimpan di AWS Secrets Manager

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "MSKmTLSSecretsManagerpolicy",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:444455556666:secret:secret_ID"
            ]
        }
    ]
}
```

Amazon MSK

Jika Anda menggunakan AUTHENTICATION NONE untuk menyambung ke sumber Amazon MSK, tidak diperlukan peran IAM. Namun, jika Anda menggunakan AUTHENTICATION IAM atau MTLS untuk mengotentikasi dengan cluster Amazon MSK Anda, cluster Amazon Redshift atau namespace Tanpa Server Amazon Redshift Anda harus memiliki peran IAM terlampir dengan izin yang sesuai. Buat peran IAM dengan kebijakan kepercayaan yang memungkinkan cluster Amazon Redshift atau ruang nama Amazon Redshift Serverless Anda mengambil peran tersebut. Setelah Anda membuat peran, tambahkan salah satu izin berikut untuk mendukung IAM atau MTLS. Untuk otentikasi MTL, sertifikat yang digunakan Amazon Redshift dapat disimpan di AWS Certificate Manager atau AWS Secrets Manager, jadi Anda harus memilih kebijakan yang cocok dengan tempat penyimpanan sertifikat. Lampirkan peran ke cluster yang disediakan Amazon Redshift atau namespace Redshift Serverless Anda. Untuk informasi tentang cara mengonfigurasi kebijakan kepercayaan untuk peran IAM, lihat Mengotor isasi Amazon Redshift untuk mengakses AWS layanan lain atas nama Anda.

Tabel berikut menunjukkan opsi konfigurasi gratis untuk disetel untuk konsumsi streaming dari Amazon MSK:

Konfigurasi Amazon Redshift	Konfigurasi Amazon MSK	Port untuk dibuka antara Redshift dan Amazon MSK
OTENTIKASI TIDAK ADA	Transportasi TLS dinonaktifkan	9092
OTENTIKASI TIDAK ADA	Transportasi TLS diaktifkan	9094

Konfigurasi Amazon Redshift	Konfigurasi Amazon MSK	Port untuk dibuka antara Redshift dan Amazon MSK
OTENTIKASI SUDAH	IAM	9098/9198
OTENTIKASI MTLS	Transportasi TLS diaktifkan	9094

Otentikasi Amazon Redshift diatur dalam pernyataan CREATE EXTERNAL SCHEMA.

> **Note**
>
> Dalam kasus di mana cluster Amazon MSK mengaktifkan otentikasi Mutual Transport Layer Security (MTL), mengonfigurasi Amazon Redshift untuk menggunakan AUTHENTICATION NONE mengarahkannya ke menggunakan port 9094 untuk akses yang tidak diautentikasi. Namun, ini akan gagal karena port sedang digunakan oleh otentikasi MTL. Karena itu, kami menyarankan Anda beralih ke OTENTIKASI mtls saat Anda menggunakan MTL.

Apache Kafka or Confluent Cloud

Untuk Apache Kafka dan Confluent Cloud, Amazon Redshift mendukung protokol koneksi berikut:

- Anda dapat menggunakan MTL atau plaintext dengan transportasi TLS untuk otentikasi saat menghubungkan ke Apache Kafka.
- Anda hanya dapat menggunakan MTL untuk otentikasi saat menghubungkan ke Confluent Cloud.

Amazon Redshift mendukung protokol enkripsi berikut untuk menghubungkan ke Apache Kafka atau Confluent Cloud:

Metode otentikasi yang didukung untuk Apache Kafka dan Confluent Cloud

Amazon Redshift	Protokol Keamanan Kafka	Dukungan Apache Kafka	Dukungan Confluent Cloud
OTENTIKASI TIDAK ADA	TEKS BIASA	Tidak	Tidak
OTENTIKASI TIDAK ADA	SSL	Ya	Tidak
OTENTIKASI SUDAH	SASL_SSL	Tidak	Tidak
OTENTIKASI MTLS	SSL	Ya (dengan sertifikat)	Ya (dengan sertifikat)

Perhatikan bahwa Amazon Redshift tidak mendukung SASL/SCRAM atau SASL/PLAINTEXT.

Menyiapkan VPC Anda

Setelah Anda membuat sumber daya otentikasi, periksa VPC Anda dan verifikasi bahwa cluster Amazon Redshift atau grup kerja Amazon Redshift Serverless Anda memiliki rute untuk mengakses sumber Apache Kafka Anda.

> **Note**
>
> Untuk Amazon MSK, aturan grup keamanan masuk untuk cluster Amazon MSK Anda harus mengizinkan grup keamanan cluster Amazon Redshift atau grup kerja Redshift Serverless Anda. Port yang Anda tentukan bergantung pada metode otentikasi yang dikonfigurasi pada cluster Amazon MSK Anda. Untuk informasi selengkapnya, lihat Informasi port dan Akses dari dalam AWS tetapi di luar VPC.

Selanjutnya, aktifkan perutean VPC yang disempurnakan pada cluster Amazon Redshift atau grup kerja Amazon Redshift Serverless Anda. Untuk informasi selengkapnya, lihat Meng aktifkan routing VPC yang disempurnakan.

Buat Tampilan Terwujud

Di bagian ini, Anda mengatur tampilan terwujud yang digunakan Amazon Redshift untuk mengakses data streaming Apache Kafka Anda.

Dengan asumsi Anda memiliki cluster Apache Kafka yang tersedia, langkah pertama adalah mendefinisikan skema di Redshift dengan `CREATE EXTERNAL SCHEMA` dan mereferensikan cluster sebagai sumber data. Setelah itu, untuk mengakses data dalam topik, tentukan `STREAM` dalam tampilan yang terwujud. Anda dapat menyimpan catatan dari topik menggunakan tipe data Amazon Redshift VARBYTE default, atau menentukan skema yang mengonversi data ke format semi-terstruktur. `SUPER` Saat Anda menanyakan tampilan yang terwujud, catatan yang dikembalikan adalah tampilan titik waktu dari topik.

1. Di Amazon Redshift, buat skema eksternal untuk dipetakan ke cluster Apacke Kafka. Sintaksnya adalah sebagai berikut:

   ```
   CREATE EXTERNAL SCHEMA MySchema
   FROM KAFKA
   [ IAM_ROLE [ default | 'iam-role-arn' ] ]
   AUTHENTICATION [ none | iam | mtls ]
   {AUTHENTICATION_ARN 'acm-certificate-arn' |  SECRET_ARN 'asm-secret-arn'};
   ```

   Dalam `FROM` klausa, `KAFKA` menunjukkan bahwa skema memetakan data dari sumber Apache Kafka.

   `AUTHENTICATION`menunjukkan jenis otentikasi untuk konsumsi streaming. Ada tiga jenis yang tersedia:

   - none — Menentukan bahwa tidak ada otentikasi yang diperlukan. Ini sesuai dengan akses yang tidak diautentikasi di MSK. Ini sesuai dengan otentikasi SSL di Apache Kafka. Metode otentikasi ini tidak didukung untuk Confluent Cloud.
   - iam — Menentukan otentikasi IAM. Anda hanya dapat menggunakan otentikasi IAM dengan Amazon MSK. Ketika Anda memilih ini, pastikan bahwa peran IAM memiliki izin untuk otentikasi IAM. Untuk informasi selengkapnya tentang menyiapkan kebijakan IAM yang diperlukan, lihatMenyiapkan konsumsi streaming dari Kafka.
   - mtls — Menentukan bahwa keamanan lapisan transportasi bersama menyediakan komunikasi yang aman dengan memfasilitasi otentikasi antara klien dan server. Dalam hal ini, kliennya adalah Redshift dan servernya adalah Apache Kafka. Untuk informasi selengkapnya tentang

mengonfigurasi konsumsi streaming dengan MTL, lihatOtentikasi dengan MTL untuk penyerapan streaming Redshift dari sumber Apache Kafka.

Perhatikan bahwa otentikasi Amazon MSK dengan nama pengguna dan kata sandi tidak didukung untuk konsumsi streaming.

`AUTHENTICATION_ARN`Parameter menentukan ARN sertifikat ACM Mutual Transport Layer Security (MTL) yang Anda gunakan untuk membuat koneksi terenkripsi.

`SECRET_ARN`Parameter menentukan arn AWS Secrets Manager rahasia yang berisi sertifikat yang akan digunakan oleh Amazon Redshift untuk MTL.

Contoh berikut menunjukkan cara mengatur URI broker untuk cluster Amazon MSK saat Anda membuat skema eksternal:

Menggunakan otentikasi IAM:

```
CREATE EXTERNAL SCHEMA my_schema
FROM KAFKA
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION IAM
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098'
```

Tidak menggunakan otentikasi:

```
CREATE EXTERNAL SCHEMA my_schema
FROM KAFKA
AUTHENTICATION none
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9092,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9092'
```

Menggunakan MTL:

```
CREATE EXTERNAL SCHEMA my_schema
FROM KAFKA
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION MTLS
```

```
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-west-1.amazonaws.com:9094,b-
 2.myTestCluster.123z8u.c2.kafka.us-west-1.amazonaws.com:9094'
{AUTHENTICATION_ARN 'acm-certificate-arn' |  SECRET_ARN 'asm-secret-arn'}
```

Untuk informasi selengkapnya tentang membuat skema eksternal, lihat MEMBUAT SKEMA EKSTERNAL.

2. Buat tampilan yang terwujud untuk mengkonsumsi data dari topik. Gunakan perintah SQL seperti contoh berikut.

```
CREATE MATERIALIZED VIEW MyView AUTO REFRESH YES AS
SELECT *
FROM MySchema."mytopic";
```

Nama topik Kafka peka huruf besar dan dapat berisi huruf besar dan kecil. Untuk mengambil dari topik dengan nama huruf besar, Anda dapat mengatur konfigurasi `enable_case_sensitive_identifier` ke `true` tingkat sesi atau database. Untuk informasi selengkapnya, lihat Nama dan pengidentifikasi dan enable _case_sensitive_identifier.

Untuk mengaktifkan penyegaran otomatis, gunakan`AUTO REFRESH YES`. Perilaku defaultnya adalah penyegaran manual.

3. Kolom metadata meliputi:

Kolom metadata	Jenis data	Deskripsi
kafka_partisi	bigint	ID partisi rekaman dari topik Kafka
kafka_offset	bigint	Offset rekaman dalam topik Kafka untuk partisi tertentu
kafka_tipe_timestamp_	harimau (1)	Jenis stempel waktu yang digunakan dalam catatan Kafka: • C - Rekam waktu pembuatan (CREATE_TIME) di sisi klien

Kolom metadata	Jenis data	Deskripsi
		• L - Rekam waktu penambahan (LOG_APPEND_TIME) di sisi server Kafka • U - Waktu pembuatan rekaman tidak tersedia (NO_TIMESTAMP_TYPE)
cap waktu kafka	stempel waktu tanpa zona waktu	Nilai stempel waktu untuk catatan
kafka_key	warbyte	Kunci rekor Kafka
nilai_kafka	warbyte	Rekor yang diterima dari Kafka
kafka_header	super	Header rekaman diterima dari Kafka
Refresh_time	stempel waktu tanpa zona waktu	Waktu penyegaran dimulai

Penting untuk dicatat jika Anda memiliki logika bisnis dalam definisi tampilan terwujud yang mengakibatkan kesalahan logika bisnis, hal ini dapat mengakibatkan kegagalan penyerapan dalam penyerapan streaming dalam beberapa kasus. Ini mungkin menyebabkan Anda harus melepaskan dan membuat kembali tampilan yang terwujud. Untuk menghindari hal ini, kami sarankan Anda menjaga logika bisnis Anda tetap sederhana dan menjalankan logika tambahan pada data setelah Anda menelannya.

4. Segarkan tampilan, yang memanggil Amazon Redshift untuk membaca dari topik dan memuat data ke tampilan yang terwujud.

```
REFRESH MATERIALIZED VIEW MyView;
```

5. Data kueri dalam tampilan terwujud.

```
select * from MyView;
```

Tampilan terwujud diperbarui langsung dari topik saat REFRESH dijalankan. Anda membuat tampilan terwujud yang dipetakan ke sumber data topik Kafka. Anda dapat melakukan pemfilteran dan agregasi pada data sebagai bagian dari definisi tampilan yang terwujud. Tampilan terwujud penyerapan streaming Anda (tampilan terwujud dasar) hanya dapat mereferensikan satu topik Kafka, tetapi Anda dapat membuat tampilan terwujud tambahan yang bergabung dengan tampilan materialisasi dasar dan dengan tampilan atau tabel terwujud lainnya.

Untuk informasi selengkapnya tentang batasan untuk konsumsi streaming, lihatPerilaku penyerapan streaming dan tipe data.

Otentikasi dengan MTL untuk penyerapan streaming Redshift dari sumber Apache Kafka

Mutual transport-layer security (MTL) menyediakan sarana bagi server untuk mengotentikasi klien yang dikirim informasinya, dan bagi klien untuk mengotentikasi server. Manfaat menggunakan MTL adalah untuk menyediakan otentikasi tepercaya untuk berbagai kasus penggunaan di beberapa aplikasi industri-vertikal. Ini termasuk kasus penggunaan di industri keuangan, ritel, pemerintah dan perawatan kesehatan. Dalam kasus penyerapan streaming ke Redshift, otentikasi terjadi antara server, yang dapat berupa Amazon MSK, Apache Kafka, atau Confluent Cloud, dan cluster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Serverless.

Topik ini memberikan prosedur dan SQL-command contoh yang menunjukkan cara untuk membuat skema eksternal yang menggunakan MTL untuk mengotentikasi antara klien Redshift dan server Apache Kafka apa pun. Langkah-langkah dalam topik ini melengkapi rangkaian lengkap langkah untuk mengatur konsumsi streaming dari sumber Apache Kafka. Untuk informasi selengkapnya, lihat Memulai dengan konsumsi streaming dari sumber Apache Kafka.

Prasyarat untuk menggunakan MTL untuk konsumsi streaming

Bagian ini memberikan langkah-langkah prasyarat untuk menggunakan MTL untuk konsumsi streaming dengan salah satu AWS Certificate Manager atau AWS Secrets Manager.

Sebagai langkah awal, Anda harus memiliki atau membuat otoritas sertifikat pribadi (PCA), yang dapat Anda gunakan untuk mengeluarkan sertifikat yang, di antara fungsi-fungsi lainnya,

memungkinkan komunikasi yang aman melalui saluran komunikasi yang aman. AWS Private Certificate Authority (Private CA) adalah layanan yang tersedia yang melakukan fungsi ini. Untuk informasi selengkapnya, lihat Membuat CA pribadi di Panduan AWS Private Certificate Authority Pengguna. Setelah membuat CA Pribadi, ekspor sertifikat CA root dan simpan ke file dengan ekstensi.pem.

Untuk membuat cluster yang menggunakan sertifikat CA, lakukan hal berikut:

Amazon MSK

1. Buat cluster Amazon MSK yang mendukung otentikasi klien mtls. Untuk informasi selengkapnya tentang mengonfigurasi cluster Amazon MSK, lihat Autenti kasi klien TLS bersama untuk Amazon MSK di Panduan Pengembang Amazon Managed Streaming untuk Apache Kafka.
2. Edit pengaturan keamanan untuk cluster Amazon MSK, aktifkan otentikasi klien TLS menggunakan AWS Certificate Manager (ACM) dan pilih AWS Private CA (PCA) yang Anda buat sebelumnya. Untuk informasi selengkapnya, lihat Memperbarui setelan keamanan cluster di Panduan Pengembang Amazon Managed Streaming untuk Apache Kafka.

Confluent Cloud

1. Buat cluster Confluent Cloud khusus, sebaiknya Wilayah AWS sama dengan cluster Amazon Redshift Anda. Untuk informasi tentang membuat cluster Confluent Cloud, lihat Membuat cluster Kafka di Confluent Cloud.
2. Unggah file pem sertifikat CA AWS Private CA root yang diekspor yang Anda buat sebelumnya. Untuk informasi selengkapnya, lihat M engelola otoritas sertifikat untuk otentikasi MTL untuk Confluent Cloud. Confluent Cloud menggunakan sertifikat ini untuk memverifikasi sertifikat klien Amazon Redshift.

Menggunakan MTL untuk konsumsi streaming dengan AWS Certificate Manager

Prosedur berikut menunjukkan cara mengonfigurasi MTL untuk penyerapan streaming Redshift dengan menggunakan AWS Certificate Manager (ACM) untuk penyimpanan dan pengelolaan sertifikat:

1. Minta sertifikat pribadi melalui ACM. Ketika Anda melakukan ini, pilih PCA yang Anda buat di bagian Prasyarat sebagai otoritas sertifikat. ACM menyimpan sertifikat yang ditandatangani dan kunci pribadi terlampir untuk komunikasi yang aman. Untuk informasi selengkapnya tentang

mengelola sertifikat dengan ACM, lihat Menerbitkan dan mengelola sertifikat di Panduan AWS Certificate Manager Pengguna.

2. Untuk peran IAM yang Anda gunakan untuk mengelola cluster Redshift atau grup kerja Amazon Redshift Serverless, lampirkan izin untuk mengekspor sertifikat, yaitu acm:. ExportCertificate Untuk informasi selengkapnya tentang menyiapkan sumber daya IAM yang diperlukan untuk penyerapan streaming, lihatMenyiapkan konsumsi streaming dari Kafka. Tentukan peran IAM yang sama pada langkah berikutnya untuk membuat skema eksternal.

   > **Note**
   >
   > Permintaan untuk AWS Certificate Manager memerlukan gateway Internet (IGW) atau gateway NAT (NGW) di VPC Anda. Jika VPC Anda tidak memiliki IGW atau NGW, lakukan hal berikut:
   >
   > - Gunakan Secrets Manager alih-alih ACM untuk menyimpan sertifikat Anda.
   > - Lampirkan titik akhir VPC Secrets Manager ke VPC Anda.
   >
   > Untuk informasi tentang menggunakan Secrets Manager dengan MTL untuk konsumsi streaming, lihat Menggunakan MTL untuk konsumsi streaming dengan AWS Secrets Manager berikut ini.

3. Dapatkan URI broker bootstrap untuk cluster Amazon MSK, Apache Kafka, atau Confluent Cloud. Untuk informasi tentang mendapatkan URI broker bootstrap untuk Amazon MSK, lihat Mendapatkan broker bootstrap untuk cluster Amazon MSK di Panduan Pengembang Amazon Managed Streaming untuk Apache Kafka.
4. Jalankan perintah SQL seperti contoh berikut untuk membuat skema eksternal yang memetakan cluster ke skema eksternal Redshift, menggunakan`mtls`.

   Amazon MSK

   ```
   CREATE EXTERNAL SCHEMA my_schema
   FROM KAFKA
   IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
   AUTHENTICATION mtls
   URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
   west-1.amazonaws.com:9094,b-2.myTestCluster.123z8u.c2.kafka.us-
   west-1.amazonaws.com:9094'
   AUTHENTICATION_ARN 'arn:aws:acm:Region:444455556666:certificate/certificate_ID';
   ```

Apache Kafka or Confluent Cloud

```
CREATE EXTERNAL SCHEMA my_schema
FROM KAFKA
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION mtls
URI 'lkc-2v531.domz6wj0p.us-west-1.aws.confluent.cloud:9092'
AUTHENTICATION_ARN 'arn:aws:acm:region:444455556666:certificate/certificate_ID';
```

Parameter penting:

- IAM_ROLE — Peran IAM yang terkait dengan cluster, untuk penyerapan streaming.
- URI — URI broker bootstrap untuk cluster. Perhatikan bahwa untuk Amazon MSK, port 9094 ditentukan untuk berkomunikasi dengan broker untuk enkripsi TLS.
- AUTHENTICATION_ARN — ARN dari sertifikat ACM. ARN tersedia di konsol ACM ketika Anda memilih sertifikat yang dikeluarkan.

Setelah melakukan langkah-langkah konfigurasi ini, Anda dapat membuat tampilan materialisasi Redshift yang mereferensikan skema yang ditentukan dalam sampel dan kemudian menggunakan REFRESH MATERIALIZED VIEW untuk mengalirkan data. Untuk informasi selengkapnya, lihat Memulai dengan konsumsi streaming dari sumber Apache Kafka.

Menggunakan MTL untuk konsumsi streaming dengan AWS Secrets Manager

Anda dapat mengonfigurasi MTL untuk penyerapan streaming Redshift dengan menggunakan AWS Secrets Manager untuk manajemen sertifikat jika Anda tidak ingin mereferensikan sertifikat di. AWS Certificate Manager Langkah-langkah berikut menjelaskan cara mengkonfigurasi MTL menggunakan Secrets Manager.

1. Buat permintaan penandatanganan sertifikat dan kunci pribadi dengan alat pilihan Anda. Kemudian Anda dapat menggunakan permintaan penandatanganan untuk menghasilkan sertifikat yang ditandatangani, menggunakan CA AWS Pribadi (PCA) yang sama yang Anda gunakan untuk menghasilkan sertifikat untuk cluster. Untuk informasi selengkapnya tentang menerbitkan sertifikat, lihat IssueCertificate di Refer AWS Private Certificate Authority ensi API.
2. Ekstrak sertifikat menggunakan AWS Private Certificate Authority. Untuk informasi selengkapnya, lihat Mengambil sertifikat pribadi di Panduan AWS Private Certificate Authority Pengguna.

3. Simpan sertifikat dan kunci pribadi yang dihasilkan pada langkah sebelumnya AWS Secrets Manager. Pilih `Other type of secret` dan gunakan format plaintext. Pasangan kunci-nilai harus dalam format`{"certificate":"<cert value>","privateKey":"<pkey value>"}`, seperti pada contoh berikut. Untuk informasi selengkapnya tentang membuat dan mengelola rahasia di AWS Secrets Manager, lihat Membuat dan mengelola rahasia dengan AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna.

```
{"certificate":"-----BEGIN CERTIFICATE-----
klhdslkfjahksgdfkgioeuyihbflahabhbdslv6akybeoiwv1hoaiusdhbahsbdi
H4hAX8/eE96qCcjkpfT84EdvHzp6fC+/WwM0oXlwUEWlvfMCXNaG5D8SqRq3qA==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
klhdslkfjahksgdfkgioeuyihbflahabhbdslv6akybeoiwv1hoaiusdhbahsbdi
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
-----END CERTIFICATE-----",
"privateKey":"-----BEGIN PRIVATE KEY-----
klhdslkfjahksgdfkgioeuyihbflahabhbdslv6akybeoiwv1hoaiusdhbahsbdi
7OD4m1dBEs3Fj++hDMH9rYRp99RqtCOf0EWOUe139KOilOsW+cyhAoc9Ci2+Jo/k
17u2N1iGILMQEZuCRtnJOkFYkw==
-----END PRIVATE KEY-----"}
```

4. Lampirkan kebijakan izin untuk mengambil rahasia ke peran IAM yang Anda gunakan untuk mengelola cluster Amazon Redshift atau grup kerja Amazon Redshift Serverless. Izin ini adalah`secretsmanager:GetSecretValue`. Untuk informasi selengkapnya, lihat Siapkan otentikasi. Untuk informasi selengkapnya tentang mengelola kebijakan IAM, lihat Meng edit kebijakan IAM. Tentukan peran IAM yang sama pada langkah berikutnya untuk membuat skema eksternal.

5. Di Redshift, jalankan perintah SQL untuk membuat skema eksternal. Anda menggunakan jenis OTENTIKASI`mtls`. Anda juga menentukan URI cluster dan ARN rahasia di AWS Secrets Manager.

```
CREATE EXTERNAL SCHEMA my_schema
FROM KAFKA
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION mtls
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9094,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9094'
SECRET_ARN 'arn:aws:secretsmanager:us-east-1:012345678910:secret:myMTLSSecret';
```

Parameter penting:

- IAM_ROLE — Peran IAM yang terkait dengan cluster, untuk penyerapan streaming.
- URI — URI broker bootstrap untuk cluster. Perhatikan bahwa untuk Amazon MSK, port 9094 ditentukan untuk berkomunikasi dengan broker untuk enkripsi TLS.
- SECRET_ARN — ARN rahasia dari Secrets Manager, berisi sertifikat untuk digunakan untuk MTL.

Mengaktifkan otentikasi MTL untuk skema eksternal yang ada

Jika Anda memiliki skema eksternal yang ada yang Anda gunakan untuk penyerapan streaming dan Anda ingin menerapkan TLS bersama untuk otentikasi, Anda dapat menjalankan perintah seperti berikut, yang menentukan otentikasi MTL dan sertifikat ACM ARN di ACM.

```
ALTER EXTERNAL SCHEMA schema_name
AUTHENTICATION mtls
AUTHENTICATION_ARN 'arn:aws:acm:Region:444455556666:certificate/certificate_ID';
```

Atau Anda dapat menentukan otentikasi MTL, dengan mengacu pada ARN rahasia di AWS Secrets Manager.

```
ALTER EXTERNAL SCHEMA schema_name
AUTHENTICATION mtls
SECRET_ARN 'arn:aws:secretsmanager:us-east-1:012345678910:secret:myMTLSSecret';
```

Untuk informasi tentang perintah ALTER EXTERNAL SCHEMA, lihatMENGUBAH SKEMA EKSTERNAL.

Menelan data streaming menggunakan Kinesis

Prosedur ini menunjukkan cara menelan data dari aliran Kinesis bernama ev_station_data, yang berisi data konsumsi dari stasiun pengisian EV yang berbeda, dalam format JSON. Skema didefinisikan dengan baik. Contoh menunjukkan cara menyimpan data sebagai JSON mentah dan juga cara mengonversi data JSON ke tipe data Amazon Redshift saat dicerna.

Pengaturan produser

1. Menggunakan Amazon Kinesis Data Streams, ikuti langkah-langkah untuk membuat aliran bernama`ev_station_data`. Pilih On-demand untuk mode Kapasitas. Untuk informasi selengkapnya, lihat Membuat Stream melalui Konsol AWS Manajemen.

2. Amazon Kinesis Data Generator dapat membantu Anda menghasilkan data uji untuk digunakan dengan streaming Anda. Ikuti langkah-langkah yang dijelaskan dalam alat untuk memulai, dan gunakan template data berikut untuk menghasilkan data Anda:

```
{

   "_id" : "{{random.uuid}}",
   "clusterID": "{{random.number(
        {   "min":1,
            "max":50
        }
    )}}",
    "connectionTime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "kWhDelivered": "{{commerce.price}}",
    "stationID": "{{random.number(
        {   "min":1,
            "max":467
        }
    )}}",
      "spaceID": "{{random.word}}-{{random.number(
        {   "min":1,
            "max":20
        }
    )}}",

   "timezone": "America/Los_Angeles",
   "userID": "{{random.number(
        {   "min":1000,
            "max":500000
        }
    )}}"
}
```

Setiap objek JSON dalam data aliran memiliki properti berikut:

```
{
    "_id": "12084f2f-fc41-41fb-a218-8cc1ac6146eb",
    "clusterID": "49",
    "connectionTime": "2022-01-31 13:17:15",
    "kWhDelivered": "74.00",
    "stationID": "421",
    "spaceID": "technologies-2",
```

```
    "timezone": "America/Los_Angeles",
    "userID": "482329"
}
```

Pengaturan Amazon Redshift

Langkah-langkah ini menunjukkan cara mengonfigurasi tampilan terwujud untuk menelan data.

1. Buat skema eksternal untuk memetakan data dari Kinesis ke objek Redshift.

   ```
   CREATE EXTERNAL SCHEMA evdata FROM KINESIS
   IAM_ROLE 'arn:aws:iam::0123456789:role/redshift-streaming-role';
   ```

   Untuk informasi tentang cara mengkonfigurasi peran IAM, lihatMemulai penyerapan streaming dari Amazon Kinesis Data Streams.

2. Buat tampilan terwujud untuk menggunakan data aliran. Contoh berikut menunjukkan cara mendefinisikan tampilan terwujud untuk menyerap data berformat JSON dari aliran Kinesis.

   Pertama, simpan catatan aliran dalam format SUPER semi-terstruktur. Dalam contoh ini, sumber JSON disimpan di Redshift tanpa mengonversi ke tipe Redshift.

   ```
   CREATE MATERIALIZED VIEW ev_station_data AS
       SELECT approximate_arrival_timestamp,
       partition_key,
       shard_id,
       sequence_number,
       case when can_json_parse(kinesis_data) then json_parse(kinesis_data) else null
    end as payload,
       case when not can_json_parse(kinesis_data) then kinesis_data else null end as
    failed_payload
       FROM evdata."ev_station_data" ;
   ```

Kueri aliran

1. Aktifkan atribut SUPER yang peka huruf besar/kecil menggunakan perintah di bawah ini. Amazon Redshift tidak peka huruf besar/kecil secara default, jadi untuk mengakses atribut SUPER yang peka huruf besar/kecil, Anda harus mengaktifkan fungsi ini.

```
SET enable_case_sensitive_super_attribute to TRUE;
```

2. Segarkan tampilan yang terwujud dengan perintah berikut untuk menarik data dari aliran.

```
REFRESH MATERIALIZED VIEW ev_station_data;
```

3. Kueri tampilan terwujud yang diperbarui untuk mendapatkan statistik penggunaan.

```
SELECT e.payload.connectionTime::date as connectiontime
,SUM(e.payload.kWhDelivered::decimal(10,2)) AS Energy_Consumed
,count(distinct e.payload.userID) AS #Users
from ev_station_data as e
group by connectiontime
order by 1 desc;
```

4. Lihat hasil.

```
connectiontime        energy_consumed    #users
2022-02-08                 4139          10
2022-02-09                 5571          10
2022-02-10                 8697          20
2022-02-11                 4408          10
2022-02-12                 4257          10
2022-02-23                 6861          10
```

AWS Glue Data Catalog pandangan

Topik ini menjelaskan cara membuat tampilan di AWS Glue Data Catalog. Anda dapat menggunakan tampilan di Katalog Data untuk mengakses data di sumber data yang berbeda menggunakan skema yang sama.

Dengan membuat tampilan di Katalog Data, Anda dapat membuat skema tampilan umum tunggal dan objek metadata untuk digunakan di seluruh mesin seperti Amazon Athena dan Amazon EMR Spark. Dengan demikian, Anda dapat menggunakan tampilan yang sama di seluruh danau data dan gudang data agar sesuai dengan kasus penggunaan Anda. Tampilan di Katalog Data khusus karena dikategorikan sebagai tampilan penentu, di mana izin akses ditentukan oleh pengguna yang membuat tampilan alih-alih pengguna menanyakan tampilan. Berikut ini adalah beberapa kasus penggunaan dan manfaat membuat tampilan di Katalog Data:

- Buat tampilan yang membatasi akses data berdasarkan izin yang dibutuhkan pengguna. Misalnya, Anda dapat menggunakan tampilan di Katalog Data untuk mencegah karyawan yang tidak bekerja di departemen SDM melihat informasi identitas pribadi (PII).
- Pastikan pengguna tidak dapat mengakses catatan yang tidak lengkap. Dengan menerapkan filter tertentu ke tampilan Anda di Katalog Data, Anda memastikan bahwa catatan data di dalam tampilan di Katalog Data selalu lengkap.
- Tampilan Katalog Data memiliki manfaat keamanan yang disertakan untuk memastikan bahwa definisi kueri yang digunakan untuk membuat tampilan harus lengkap untuk membuat tampilan. Manfaat keamanan ini berarti bahwa tampilan di Katalog Data tidak rentan terhadap perintah SQL dari pemain jahat.
- Tampilan di Katalog Data mendukung keuntungan yang sama seperti tampilan normal, seperti mengizinkan pengguna mengakses tampilan tanpa membuat tabel dasar tersedia bagi pengguna.

Untuk membuat tampilan di Katalog Data, Anda harus memiliki tabel eksternal Spectrum, objek yang terkandung dalam Formation-managed datashare Lake, atau tabel Apache Iceberg.

Definisi tampilan Katalog Data disimpan di AWS Glue Data Catalog. Gunakan AWS Lake Formation untuk memberikan akses melalui hibah sumber daya, hibah kolom, atau kontrol akses berbasis tag. Untuk informasi selengkapnya tentang pemberian dan pencabutan akses di Lake Formation, lihat Memberikan dan mencabut izin pada sumber daya Katalog Data.

Saat Anda menggunakan Amazon Redshift untuk menjalankan kueri yang mereferensikan AWS Glue Data Catalog tampilan tersebut, Amazon Redshift secara otomatis menutupi bidang di tabel sistem

tertentu dan menampilkan kolom saat mencatat metadata tentang kueri tersebut. Untuk informasi selengkapnya, lihat Log in aman di Panduan Manajemen Amazon Redshift.

Prasyarat

Sebelum Anda dapat membuat tampilan di Katalog Data, pastikan bahwa Anda telah menyelesaikan prasyarat berikut:

- Pastikan peran IAM Anda memiliki kebijakan kepercayaan berikut.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": [
                    "glue.amazonaws.com",
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- Anda juga memerlukan kebijakan peran pass berikut.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com",
                        "lakeformation.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

- Terakhir, Anda juga memerlukan izin berikut.
  - Glue:GetDatabase
  - Glue:GetDatabases
  - Glue:CreateTable
  - Glue:GetTable
  - Glue:UpdateTable
  - Glue:DeleteTable
  - Glue:GetTables
  - Glue:SearchTables
  - Glue:BatchGetPartition
  - Glue:GetPartitions
  - Glue:GetPartition
  - Glue:GetTableVersion
  - Glue:GetTableVersions

End-to-end contoh

Mulailah dengan membuat skema eksternal berdasarkan database Katalog Data Anda.

```
CREATE EXTERNAL SCHEMA IF NOT EXISTS external_schema FROM DATA CATALOG DATABASE
 'external_data_catalog_db'
IAM_ROLE 'arn:aws:iam::123456789012:role/sample-role';
```

Anda sekarang dapat membuat tampilan Katalog Data.

```
CREATE EXTERNAL PROTECTED VIEW external_schema.remote_view
AS SELECT * FROM external_schema.remote_table;
```

Anda kemudian dapat mulai menanyakan pandangan Anda.

```
SELECT * FROM external_schema.remote_view;
```

Untuk informasi selengkapnya tentang perintah SQL terkait dengan tampilan di Katalog Data, lihat MEMBUAT TAMPILAN EKSTERNAL, MENGUBAH TAMPILAN EKSTER NAL, dan MELEPASKAN TAM PILAN EKSTERNAL.

Pertimbangan dan batasan

Berikut ini adalah pertimbangan dan batasan yang berlaku untuk tampilan yang dibuat dalam Katalog Data.

- AWS Glue Data Catalog tampilan hanya didukung pada cluster yang disediakan RG dan RA3 atau kelompok kerja Redshift Serverless.
- Anda tidak dapat membuat tampilan Katalog Data yang didasarkan pada tampilan lain.
- Anda hanya dapat memiliki 10 tabel dasar dalam tampilan Katalog Data.
- Pendefinisikan tampilan harus memiliki `SELECT GRANTABLE` izin penuh pada tabel dasar.
- Tampilan hanya dapat berisi objek Formasi Danau dan bawaan. Objek berikut tidak diizinkan di dalam tampilan.
  - Tabel sistem
  - User-defined fungsi (UDF)
  - Tabel pergeseran merah, tampilan, tampilan terwujud, dan tampilan pengikatan terlambat yang tidak ada dalam berbagi data yang dikelola Lake Formation.
- Tampilan tidak dapat berisi tabel Redshift Spectrum bersarang.
- AWS Glue representasi objek dasar tampilan harus berada di Wilayah yang sama Akun AWS dengan tampilan.

Mengkueri data spasial di Amazon Redshift

Data spasial menggambarkan posisi dan bentuk geometri dalam ruang yang ditentukan (sistem referensi spasial). Amazon Redshift mendukung data spasial dengan tipe GEOGRAPHY data GEOMETRY dan, yang berisi data spasial dan secara opsional pengidentifikasi sistem referensi spasial data (SRID).

Data spasial berisi data geometris yang dapat Anda gunakan untuk mewakili fitur geografis. Contoh jenis data ini termasuk laporan cuaca, arah peta, tweet dengan posisi geografis, lokasi toko, dan rute maskapai penerbangan. Data spasial memainkan peran penting dalam analitik bisnis, pelaporan, dan peramalan.

Anda dapat menanyakan data spasial dengan fungsi Amazon Redshift SQL. Data spasial berisi nilai geometris untuk suatu objek.

Oper GEOMETRY asi tipe data bekerja pada bidang Cartesian. Meskipun pengenal sistem referensi spasial (SRID) disimpan di dalam objek, SRID ini hanyalah pengidentifikasi sistem koordinat dan tidak memainkan peran dalam algoritma yang digunakan untuk memproses objek. GEOMETRY Sebaliknya, operasi pada tipe GEOGRAPHY data memperlakukan koordinat di dalam objek sebagai koordinat bola pada spheroid. Spheroid ini didefinisikan oleh SRID, yang mereferensikan sistem referensi spasial geografis. Secara default, tipe GEOGRAPHY data dibuat dengan referensi spasial (SRID) 4326, merujuk Sistem Geodetik Dunia (WGS) 84. Untuk informasi lebih lanjut tentang SRID, lihat Sistem referensi spasial di Wikipedia.

Anda dapat menggunakan fungsi ST_transform untuk mengubah koordinat dari berbagai sistem referensi spasial. Setelah transformasi koordinat selesai, Anda juga dapat menggunakan cast sederhana di antara keduanya, selama input GEOMETRY dikodekan dengan SRID geografis. Pemeran ini hanya menyalin koordinat tanpa transformasi lebih lanjut. Contoh:

```
SELECT ST_AsEWKT(ST_GeomFromEWKT('SRID=4326;POINT(10 20)')::geography);
```

```
st_asewkt
------------------------
 SRID=4326;POINT(10 20)
```

Untuk lebih memahami perbedaan antara GEOMETRY dan tipe GEOGRAPHY data, pertimbangkan menghitung jarak antara bandara Berlin (BER) dan bandara San Francisco (SFO) menggunakan

Sistem Geodetik Dunia (WGS) 84. Menggunakan tipe GEOGRAPHY data, hasilnya dalam meter. Saat menggunakan tipe GEOMETRY data dengan SRID 4326, hasilnya dalam derajat, yang tidak dapat dikonversi ke meter karena jarak satu derajat tergantung di mana geometri globe berada.

Perhitungan pada tipe GEOGRAPHY data sebagian besar digunakan untuk perhitungan bumi bulat yang realistis seperti area tepat suatu negara tanpa distorsi. Tetapi mereka jauh lebih mahal untuk dihitung. Oleh karena itu, ST_Transform dapat mengubah koordinat Anda ke sistem koordinat proyeksi lokal yang sesuai dan melakukan perhitungan pada tipe GEOMETRY data lebih cepat.

Menggunakan data spasial, Anda dapat menjalankan kueri untuk melakukan hal berikut:

- Temukan jarak antara dua titik.
- Periksa apakah satu wilayah (poligon) berisi lain.
- Periksa apakah satu linestring memotong linestring atau poligon lainnya.

Anda dapat menggunakan tipe GEOMETRY data untuk menyimpan nilai-nilai data spasial. GEOMETRYNilai di Amazon Redshift dapat mendefinisikan tipe data primitif geometri dua dimensi (2D), tiga dimensi (3DZ), dua dimensi dengan ukuran (3DM), dan empat dimensi (4D):

- Geometri dua dimensi (2D) ditentukan oleh dua koordinat Cartesian (x, y) dalam bidang.
- Geometri tiga dimensi (3DZ) ditentukan oleh tiga koordinat Cartesian (x, y, z) dalam ruang.
- Geometri dua dimensi dengan ukuran (3DM) ditentukan oleh tiga koordinat (x, y, m), di mana dua yang pertama adalah koordinat Cartesian dalam bidang dan yang ketiga adalah pengukuran.
- Geometri empat dimensi (4D) ditentukan oleh empat koordinat (x, y, z, m), di mana tiga yang pertama adalah koordinat Cartesian dalam ruang dan yang keempat adalah pengukuran.

Untuk informasi lebih lanjut tentang tipe data primitif geometri, lihat representasi Well-known teks geometri di Wikipedia.

Anda dapat menggunakan tipe GEOGRAPHY data untuk menyimpan nilai-nilai data spasial. GEOGRAPHYNilai di Amazon Redshift dapat mendefinisikan tipe data primitif geometri dua dimensi (2D), tiga dimensi (3DZ), dua dimensi dengan ukuran (3DM), dan empat dimensi (4D):

- Geometri dua dimensi (2D) ditentukan oleh koordinat bujur dan lintang pada spheroid.
- Geometri tiga dimensi (3DZ) ditentukan oleh koordinat bujur, lintang, dan ketinggian pada spheroid.
- Geometri dua dimensi dengan ukuran (3DM) ditentukan oleh tiga koordinat (bujur, lintang, ukuran), di mana dua yang pertama adalah koordinat sudut pada bola dan yang ketiga adalah pengukuran.

- Geometri empat dimensi (4D) ditentukan oleh empat koordinat (bujur, lintang, ketinggian, ukuran), di mana tiga yang pertama adalah bujur, lintang dan ketinggian, dan yang keempat adalah pengukuran.

Untuk informasi lebih lanjut tentang sistem koordinat geografis, lihat Sistem koordinat geografis dan Sistem koordinat bola di Wikipedia.

Tipe `GEOMETRY` dan `GEOGRAPHY` data memiliki subtipe berikut:

- `POINT`
- `LINESTRING`
- `POLYGON`
- `MULTIPOINT`
- `MULTILINESTRING`
- `MULTIPOLYGON`
- `GEOMETRYCOLLECTION`

Ada fungsi Amazon Redshift SQL yang mendukung representasi data geometris berikut:

- GeoJSON
- Well-known teks (WKT)
- Teks terkenal yang diperluas (EWKT)
- Well-known representasi biner (WKB)
- Biner terkenal yang diperluas (EWKB)

Anda dapat mentransmisikan antara `GEOMETRY` dan `GEOGRAPHY` tipe data.

SQL berikut melemparkan linestring dari a `GEOMETRY` ke a. `GEOGRAPHY`

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)')::geography);
```

```
 st_asewkt
-------------------------------------------
 SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

SQL berikut melemparkan linestring dari a GEOGRAPHY ke a. GEOMETRY

```
SELECT ST_AsEWKT(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)')::geometry);
```

```
 st_asewkt
-------------------------------------------
 SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

Amazon Redshift menyediakan banyak fungsi SQL untuk menanyakan data spasial. Kecuali ST_IsValid fungsi, fungsi spasial yang menerima GEOMETRY objek sebagai argumen mengharapkan GEOMETRY objek ini menjadi geometri yang valid. Jika GEOGRAPHY objek GEOMETRY atau tidak valid, maka perilaku fungsi spasial tidak ditentukan. Untuk informasi lebih lanjut tentang validitas, lihatValiditas geometris.

Untuk detail tentang fungsi SQL untuk menanyakan data spasial, lihatFungsi spasial.

Untuk detail tentang memuat data spasial, lihatMemuat kolom tipe data GEOMETRY atau GEOGRAPHY.

Topik

- Tutorial: Menggunakan fungsi SQL spasial dengan Amazon Redshift
- Memuat shapefile ke Amazon Redshift
- Terminologi untuk data spasial Amazon Redshift
- Pertimbangan saat menggunakan data spasial dengan Amazon Redshift

Tutorial: Menggunakan fungsi SQL spasial dengan Amazon Redshift

Tutorial ini menunjukkan cara menggunakan beberapa fungsi SQL spasial dengan Amazon Redshift.

Untuk melakukan ini, Anda menanyakan dua tabel menggunakan fungsi SQL spasial. Tutorial ini menggunakan data dari kumpulan data publik yang menghubungkan data lokasi akomodasi sewa dengan kode pos di Berlin, Jerman.

Topik

- Prasyarat

- Langkah 1: Buat tabel dan muat data uji
- Langkah 2: Kueri data spasial
- Langkah 3: Bersihkan sumber daya Anda

Prasyarat

Untuk tutorial ini, Anda memerlukan sumber daya berikut:

- Cluster dan database Amazon Redshift yang ada yang dapat Anda akses dan perbarui. Di cluster yang ada, Anda membuat tabel, memuat data sampel, dan menjalankan kueri SQL untuk mendemonstrasikan fungsi spasial. Cluster Anda harus memiliki setidaknya dua node. Untuk mempelajari cara membuat cluster, ikuti langkah-langkah di Panduan Mem ulai Amazon Redshift.
- Untuk menggunakan editor kueri Amazon Redshift, pastikan cluster Anda berada di Wil AWS ayah yang mendukung editor kueri. Untuk informasi selengkapnya, lihat Mengkueri database menggunakan editor kueri di Panduan Manajemen Amazon Redshift.
- AWS kredenSIAL untuk cluster Amazon Redshift Anda yang memungkinkannya memuat data pengujian dari Amazon S3. Untuk informasi tentang cara mengakses AWS layanan lain seperti Amazon S3, lihat Mengotor isasi Amazon Redshift untuk mengakses AWS layanan.
- Nama peran AWS Identity and Access Management (IAM)`mySpatialDemoRole`, yang memiliki kebijakan `AmazonS3ReadOnlyAccess` terkelola yang dilampirkan untuk membaca data Amazon S3. Untuk membuat peran dengan izin memuat data dari bucket Amazon S3, lihat Mengotor isasi operasi COPY, UNLOAD, dan CREATE EXTERNAL SCHEMA menggunakan peran IAM di Panduan Manajemen Amazon Redshift.
- Setelah Anda membuat peran IAM`mySpatialDemoRole`, peran tersebut memerlukan asosiasi dengan cluster Amazon Redshift Anda. Untuk informasi selengkapnya tentang cara membuat asosiasi tersebut, lihat Mengo torisasi operasi COPY, UNLOAD, dan CREATE EXTERNAL SCHEMA menggunakan peran IAM di Panduan Manajemen Amazon Redshift.

Langkah 1: Buat tabel dan muat data uji

Data sumber yang digunakan oleh tutorial ini ada dalam file bernama `accommodations.csv` dan`zipcodes.csv`.

`accommodations.csv`File ini adalah data sumber terbuka dari insideairbnb.com. `zipcodes.csv`File ini menyediakan kode pos yang merupakan data sumber terbuka dari lembaga statistik nasional Berlin-Brandenburg di Jerman (Amt für Statistik Berlin-Brandenburg). Kedua sumber

data disediakan di bawah lisensi Creative Commons. Data terbatas pada wilayah Berlin, Jerman. File-file ini terletak di bucket publik Amazon S3 untuk digunakan dengan tutorial ini.

Anda dapat mengunduh data sumber secara opsional dari tautan Amazon S3 berikut:

- Sumber data untuk `accommodations` tabel.
- Sumber data untuk `zipcode` tabel.

Gunakan prosedur berikut untuk membuat tabel dan memuat data uji.

Untuk membuat tabel dan memuat data uji

1. Buka editor kueri Amazon Redshift. Untuk informasi selengkapnya tentang bekerja dengan editor kueri, lihat Mengkueri database menggunakan editor kueri di Panduan Manajemen Amazon Redshift.
2. Jatuhkan tabel apa pun yang digunakan oleh tutorial ini jika sudah ada di database Anda. Untuk informasi selengkapnya, lihat Langkah 3: Bersihkan sumber daya Anda.
3. Buat `accommodations` tabel untuk menyimpan lokasi geografis setiap akomodasi (bujur dan lintang), nama listing, dan data bisnis lainnya.

   Tutorial ini mengeksplorasi penyewaan kamar di Berlin, Jerman. `shape`Kolom menyimpan titik-titik geografis lokasi akomodasi. Kolom lainnya berisi informasi tentang sewa.

   Untuk membuat `accommodations` tabel, jalankan pernyataan SQL berikut di editor kueri Amazon Redshift.

```
CREATE TABLE public.accommodations (
  id INTEGER PRIMARY KEY,
  shape GEOMETRY,
  name VARCHAR(100),
  host_name VARCHAR(100),
  neighbourhood_group VARCHAR(100),
  neighbourhood VARCHAR(100),
  room_type VARCHAR(100),
  price SMALLINT,
  minimum_nights SMALLINT,
  number_of_reviews SMALLINT,
  last_review DATE,
  reviews_per_month NUMERIC(8,2),
  calculated_host_listings_count SMALLINT,
```

```
  availability_365 SMALLINT
);
```

4. Buat `zipcode` tabel di editor kueri untuk menyimpan kode pos Berlin.

   Kode pos didefinisikan sebagai poligon di `wkb_geometry` kolom. Sisa kolom menjelaskan metadata spasial tambahan tentang kode pos.

   Untuk membuat `zipcode` tabel, jalankan pernyataan SQL berikut di editor kueri Amazon Redshift.

```
CREATE TABLE public.zipcode (
  ogc_field INTEGER PRIMARY KEY NOT NULL,
  wkb_geometry GEOMETRY,
  gml_id VARCHAR(256),
  spatial_name VARCHAR(256),
  spatial_alias VARCHAR(256),
  spatial_type VARCHAR(256)
 );
```

5. Muat tabel menggunakan data sampel.

   Data sampel untuk tutorial ini disediakan dalam bucket Amazon S3 yang memungkinkan akses baca ke semua AWS pengguna yang diautentikasi. Pastikan Anda memberikan kredentif valid AWS yang mengizinkan akses ke Amazon S3.

   Untuk memuat data uji ke tabel Anda, jalankan perintah COPY berikut. Ganti *`account-number`* dengan nomor re AWS kening Anda sendiri. Segmen string kredensional yang tertutup dalam tanda kutip tunggal tidak dapat berisi spasi atau jeda baris.

```
COPY public.accommodations
FROM 's3://redshift-downloads/spatial-data/accommodations.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
CREDENTIALS 'aws_iam_role=arn:aws:iam::account-number:role/mySpatialDemoRole';
```

```
COPY public.zipcode
FROM 's3://redshift-downloads/spatial-data/zipcode.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
CREDENTIALS 'aws_iam_role=arn:aws:iam::account-number:role/mySpatialDemoRole';
```

6. Pastikan setiap tabel dimuat dengan benar dengan menjalankan perintah berikut.

```
select count(*) from accommodations;
```

```
select count(*) from zipcode;
```

Hasil berikut menunjukkan jumlah baris di setiap tabel data uji.

Nama tabel	Baris
akomodasi	222.248
kode pos	190

Langkah 2: Kueri data spasial

Setelah tabel Anda dibuat dan dimuat, Anda dapat menanyakan mereka menggunakan pernyataan SQL SELECT. Kueri berikut menunjukkan beberapa informasi yang dapat Anda ambil. Anda dapat menulis banyak kueri lain yang menggunakan fungsi spasial untuk memenuhi kebutuhan Anda.

Untuk menanyakan data spasial

1. Kueri untuk mendapatkan jumlah total daftar yang disimpan dalam `accommodations` tabel, seperti yang ditunjukkan berikut. Sistem referensi spasial adalah World Geodetic System (WGS) 84, yang memiliki pengidentifikasi referensi spasial unik 4326.

```
SELECT count(*) FROM public.accommodations WHERE ST_SRID(shape) = 4326;
```

```
 count
-------
 22248
```

2. Ambil objek geometri dalam format teks terkenal (WKT) dengan beberapa atribut tambahan. Selain itu, Anda dapat memvalidasi jika data kode pos ini juga disimpan di Sistem Geodetik Dunia (WGS) 84, yang menggunakan ID referensi spasial (SRID) 4326. Data spasial harus disimpan dalam sistem referensi spasial yang sama agar dapat dioperasikan.

```
SELECT ogc_field, spatial_name, spatial_type, ST_SRID(wkb_geometry),
 ST_AsText(wkb_geometry)
FROM public.zipcode
ORDER BY spatial_name;
```

```
ogc_field  spatial_name  spatial_type  st_srid  st_astext
-------------------------------------------------------------
0          10115        Polygon        4326     POLYGON((...))
4          10117        Polygon        4326     POLYGON((...))
8          10119        Polygon        4326     POLYGON((...))
...
(190 rows returned)
```

3. Pilih poligon Berlin Mitte (10117), wilayah Berlin, dalam format GeoJSON, dimensinya, dan jumlah titik dalam poligon ini.

```
SELECT ogc_field, spatial_name, ST_AsGeoJSON(wkb_geometry),
 ST_Dimension(wkb_geometry), ST_NPoints(wkb_geometry)
FROM public.zipcode
WHERE spatial_name='10117';
```

```
ogc_field  spatial_name  spatial_type
 st_dimension  st_npoint
-------------------------------------------------------------------------------------------
4          10117         {"type":"Polygon", "coordinates":[[[...]]]}    2
    331
```

4. Jalankan perintah SQL berikut untuk melihat berapa banyak akomodasi yang berada dalam jarak 500 meter dari Gerbang Brandenburg.

```
SELECT count(*)
FROM public.accommodations
WHERE ST_DistanceSphere(shape, ST_GeomFromText('POINT(13.377704 52.516431)', 4326))
 < 500;
```

```
count
------
```

```
  29
```

5. Dapatkan lokasi kasar Gerbang Brandenburg dari data yang disimpan di akomodasi yang terdaftar sebagai terdekat dengan menjalankan kueri berikut.

   Kueri ini membutuhkan subselect. Ini mengarah ke jumlah yang berbeda karena lokasi yang diminta tidak sama dengan kueri sebelumnya karena lebih dekat ke akomodasi.

```
WITH poi(loc) as (
  SELECT st_astext(shape) FROM accommodations WHERE name LIKE '%brandenburg gate%'
)
SELECT count(*)
FROM accommodations a, poi p
WHERE ST_DistanceSphere(a.shape, ST_GeomFromText(p.loc, 4326)) < 500;
```

```
count
------
  60
```

6. Jalankan kueri berikut untuk menampilkan detail semua akomodasi di sekitar Gerbang Brandenburg, diurutkan berdasarkan harga dalam urutan menurun.

```
SELECT name, price, ST_AsText(shape)
FROM public.accommodations
WHERE ST_DistanceSphere(shape, ST_GeomFromText('POINT(13.377704 52.516431)', 4326))
 < 500
ORDER BY price DESC;
```

```
name                                                price  st_astext
-------------------------------------------------------------------------------------------
DUPLEX APARTMENT/PENTHOUSE in 5* LOCATION! 7583        300
 POINT(13.3826510209548 52.5159819722552)
DUPLEX-PENTHOUSE IN FIRST LOCATION! 7582               300
 POINT(13.3799997083855 52.5135918444834)
...
(29 rows returned)
```

7. Jalankan kueri berikut untuk mengambil akomodasi paling mahal dengan kode posnya.

```
SELECT
  a.price, a.name, ST_AsText(a.shape),
  z.spatial_name, ST_AsText(z.wkb_geometry)
FROM accommodations a, zipcode z
WHERE price = 9000 AND ST_Within(a.shape, z.wkb_geometry);
```

```
price   name                                st_astext
     spatial_name      st_astext
-------------------------------------------------------------------------------------------
9000    Ueber den Dächern Berlins Zentrum    POINT(13.334436985013
 52.4979779501538)    10777             POLYGON((13.3318284987227
 52.4956021172799,...
```

8. Hitung harga maksimum, minimum, atau median akomodasi dengan menggunakan subkueri.

   Kueri berikut mencantumkan harga rata-rata akomodasi berdasarkan kode pos.

```
SELECT
  a.price, a.name, ST_AsText(a.shape),
  z.spatial_name, ST_AsText(z.wkb_geometry)
FROM accommodations a, zipcode z
WHERE
  ST_Within(a.shape, z.wkb_geometry) AND
  price = (SELECT median(price) FROM accommodations)
ORDER BY a.price;
```

```
price name                             st_astext
 spatial_name   st_astext
-------------------------------------------------------------------------------------------

45    "Cozy room Berlin-Mitte"        POINT(13.3864349535358 52.5292016386514)
 10115          POLYGON((13.3658598465795 52.535659581048,...
...
(723 rows returned)
```

9. Jalankan kueri berikut untuk mengambil jumlah akomodasi yang terdaftar di Berlin. Untuk menemukan titik panas, ini dikelompokkan berdasarkan kode pos dan diurutkan berdasarkan jumlah pasokan.

```
SELECT z.spatial_name as zip, count(*) as numAccommodations
FROM public.accommodations a, public.zipcode z
WHERE ST_Within(a.shape, z.wkb_geometry)
GROUP BY zip
ORDER BY numAccommodations DESC;
```

```
zip   numaccommodations
---------------------------
10245 872
10247 832
10437 733
10115 664
...
(187 rows returned)
```

Langkah 3: Bersihkan sumber daya Anda

Cluster Anda terus menambah biaya selama masih berjalan. Ketika Anda telah menyelesaikan tutorial ini, Anda dapat menghapus cluster sampel Anda.

Jika Anda ingin menyimpan cluster tetapi memulihkan penyimpanan yang digunakan oleh tabel data uji, jalankan perintah berikut untuk menghapus tabel.

```
drop table public.accommodations cascade;
```

```
drop table public.zipcode cascade;
```

Memuat shapefile ke Amazon Redshift

Anda dapat menggunakan perintah COPY untuk menelan shapefile Esri yang disimpan di Amazon S3 ke dalam tabel Amazon Redshift. Sebuah shapefile menyimpan informasi lokasi geometris dan atribut fitur geografis dalam format vektor. Format shapefile dapat menggambarkan objek spasial secara spasial seperti titik, garis, dan poligon. Untuk informasi lebih lanjut tentang shapefile, lihat Shapefile di Wikipedia.

Perintah COPY mendukung parameter format data`SHAPEFILE`. Secara default, kolom pertama dari shapefile adalah kolom `GEOMETRY` atau`IDENTITY`. Semua kolom berikutnya mengikuti urutan yang ditentukan dalam shapefile. Namun, tabel target tidak perlu berada dalam tata letak yang tepat ini karena Anda dapat menggunakan pemetaan kolom COPY untuk menentukan urutan. Untuk informasi tentang dukungan shapefile perintah COPY, lihat. SHAPEFILE

Dalam beberapa kasus, ukuran geometri yang dihasilkan mungkin lebih besar dari maksimum untuk menyimpan geometri di Amazon Redshift. Jika demikian, Anda dapat menggunakan opsi COPY `SIMPLIFY` atau `SIMPLIFY AUTO` untuk menyederhanakan geometri selama konsumsi sebagai berikut:

- Tentukan `SIMPLIFY` *`tolerance`* untuk menyederhanakan semua geometri selama konsumsi menggunakan Ramer-Douglas-Peucker algoritma dan toleransi yang diberikan.
- Tentukan `SIMPLIFY AUTO` tanpa toleransi untuk menyederhanakan hanya geometri yang lebih besar dari ukuran maksimum menggunakan Ramer-Douglas-Peucker algoritma. Pendekatan ini menghitung toleransi minimum yang cukup besar untuk menyimpan objek dalam batas ukuran maksimum.
- Tentukan `SIMPLIFY AUTO` *`max_tolerance`* untuk menyederhanakan hanya geometri yang lebih besar dari ukuran maksimum menggunakan Ramer-Douglas-Peucker algoritma dan toleransi yang dihitung secara otomatis. Pendekatan ini memastikan bahwa toleransi tidak melebihi toleransi maksimum.

Untuk informasi tentang ukuran maksimum nilai `GEOMETRY` data, lihatPertimbangan saat menggunakan data spasial dengan Amazon Redshift.

Dalam beberapa kasus, toleransi cukup rendah sehingga catatan tidak dapat menyusut di bawah ukuran maksimum nilai `GEOMETRY` data. Dalam kasus ini, Anda dapat menggunakan `MAXERROR` opsi perintah COPY untuk mengabaikan semua atau hingga sejumlah kesalahan penyerapan.

Perintah COPY juga mendukung pemuatan shapefiles GZIP. Untuk melakukan ini, tentukan parameter COPY GZIP. Dengan opsi ini, semua komponen shapefile harus dikompresi secara independen dan berbagi akhiran kompresi yang sama.

Jika file deskripsi proyeksi (.prj) ada dengan shapefile, Redshift menggunakannya untuk menentukan id sistem referensi spasial (SRID). Jika SRID valid, geometri yang dihasilkan memiliki SRID ini ditetapkan. Jika nilai SRID yang terkait dengan geometri input tidak ada, geometri yang dihasilkan memiliki nilai SRID nol. Anda dapat menonaktifkan deteksi otomatis id sistem referensi spasial di tingkat sesi dengan menggunakan `SET read_srid_on_shapefile_ingestion` to`OFF`.

Kueri tampilan SYS_SPATIAL_SIMPLIFY atau SVL_SPATIAL_SIMPLIFY sistem untuk melihat catatan mana yang telah disederhanakan, bersama dengan toleransi yang dihitung. Saat Anda menentukanSIMPLIFY *tolerance*, tampilan ini berisi catatan untuk setiap operasi COPY. Jika tidak, itu berisi catatan untuk setiap geometri yang disederhanakan. Untuk informasi selengkapnya, lihat SYS_SPATIAL_SIMPLIFIKASI atau SVL_SPATIAL_MENYEDERHANAKAN.

Untuk contoh memuat shapefile, lihat. Memuat shapefile ke Amazon Redshift

Terminologi untuk data spasial Amazon Redshift

Istilah berikut digunakan untuk menggambarkan beberapa fungsi spasial Amazon Redshift.

Kotak pembatas

Kotak pembatas geometri atau geografi didefinisikan sebagai produk silang (lintas dimensi) dari luasan koordinat semua titik dalam geometri atau geografi. Untuk geometri dua dimensi, kotak pembatas adalah persegi panjang yang sepenuhnya mencakup semua titik dalam geometri. Misalnya, kotak pembatas poligon POLYGON((0 0,1 0,0 2,0 0)) adalah persegi panjang yang ditentukan oleh titik-titik (0, 0) dan (1, 2) sebagai sudut kiri bawah dan kanan atas. Amazon Redshift menghitung dan menyimpan kotak pembatas di dalam geometri untuk mempercepat predikat geometris dan gabungan spasial. Misalnya jika kotak pembatas dari dua geometri tidak berpotongan, maka kedua geometri ini tidak dapat berpotongan, dan mereka tidak dapat berada dalam kumpulan hasil gabungan spasial menggunakan predikat ST_intersects.

Anda dapat menggunakan fungsi spasial untuk menambahkan (AddBbox), drop (DropbBox), dan menentukan support (DukunganBBox) untuk kotak pembatas. Amazon Redshift mendukung pra-perhitungan kotak pembatas untuk semua subtipe geometri.

Contoh berikut menunjukkan cara memperbarui geometri yang ada dalam tabel untuk menyimpannya dengan kotak pembatas. Jika cluster Anda berada di cluster versi 1.0.26809 atau yang lebih baru, maka semua geometri baru dibuat dengan kotak pembatas yang telah dihitung sebelumnya secara default.

```
UPDATE my_table SET geom = AddBBox(geom) WHERE SupportsBBox(geom) = false;
```

Setelah Anda memperbarui geometri yang ada, kami sarankan Anda menjalankan perintah VACUUM pada tabel yang diperbarui. Untuk informasi selengkapnya, lihat VAKUM.

Untuk mengatur apakah geometri dikodekan dengan kotak pembatas selama sesi, lihat. default_geometri_encoding

Validitas geometris

Algoritma geometris yang digunakan oleh Amazon Redshift mengasumsikan bahwa geometri input adalah geometri yang valid. Jika input ke algoritma tidak valid, maka hasilnya tidak ditentukan. Bagian berikut menjelaskan definisi validitas geometris yang digunakan oleh Amazon Redshift untuk setiap subtipe geometri.

Poin

Poin dianggap valid jika salah satu kondisi berikut benar:

- Intinya adalah titik kosong.
- Semua koordinat titik adalah bilangan titik mengambang terbatas.

Sebuah titik bisa menjadi titik kosong.

Linestring

Linestring dianggap valid jika salah satu dari kondisi berikut benar:

- Linestring kosong; artinya, tidak mengandung poin.
- Semua titik dalam garis yang tidak kosong memiliki koordinat yang merupakan bilangan titik mengambang terbatas.
- Garis, jika tidak kosong, harus satu dimensi; artinya, ia tidak dapat merosot ke suatu titik.

Linestring tidak dapat berisi titik kosong.

Linestring dapat memiliki titik berurutan duplikat.

Garis dapat memiliki persimpangan diri.

Polygon

Sebuah poligon dianggap valid jika salah satu dari kondisi berikut benar:

- Poligon kosong; artinya, tidak mengandung cincin.
- Jika tidak kosong, poligon valid jika semua kondisi berikut benar:
  - Semua cincin poligon valid. Cincin dianggap valid jika semua kondisi berikut benar:
    - Semua titik cincin memiliki koordinat yang merupakan bilangan titik mengambang terbatas.
    - Cincin ditutup; yaitu, titik pertama dan titik terakhirnya bertepatan.

    - Cincin itu tidak memiliki persimpangan diri.
    - Cincin itu dua dimensi.
  - Cincin poligon harus memiliki orientasi yang konsisten. Artinya, jika Anda melintasi cincin apa pun, bagian dalam poligon berada di kanan atau ke kiri Anda.
  - Semua cincin interior harus berada di dalam cincin eksterior poligon.
  - Semua cincin interior harus memiliki orientasi yang sama, dan itu harus menjadi orientasi yang berlawanan dari cincin eksterior.
  - Cincin interior tidak dapat bersarang; yaitu, cincin interior tidak bisa berada di dalam cincin interior lain.
  - Cincin interior dan eksterior hanya dapat berpotongan pada sejumlah titik yang terbatas.
  - Bagian dalam poligon harus dihubungkan secara sederhana.

  Sebuah poligon tidak dapat berisi titik kosong.

Multipoint

  Multipoint dianggap valid jika salah satu dari kondisi berikut benar:

  - Multipoint kosong; artinya, tidak mengandung poin.
  - Multipoint tidak kosong, dan semua titik valid sesuai dengan definisi validitas titik.

  Multipoint dapat berisi satu atau lebih titik kosong.

  Multipoint dapat memiliki poin duplikat.

Multilinestring

  Multilinestring dianggap valid jika salah satu dari kondisi berikut benar:

  - Multilinestring kosong; artinya, tidak mengandung linestring.
  - Semua linestring dalam multilinestring yang tidak kosong valid sesuai dengan definisi validitas linestring.

  Multilinestring tidak kosong yang hanya terdiri dari linestring kosong dianggap valid.

  Garis kosong dalam multilinestring tidak mempengaruhi validitasnya.

  Multilinestring dapat memiliki linestring dengan titik berurutan duplikat.

  Multilinestring dapat memiliki persimpangan diri.

  Multilinestring tidak dapat berisi titik kosong.

Multipoligon

Multipoligon dianggap valid jika salah satu kondisi berikut benar:

- Multipoligon tidak mengandung poligon apapun (kosong).
- Multipoligon tidak kosong dan semua hal berikut benar:
  - Semua poligon dalam multipoligon valid.
  - Tidak ada dua poligon dalam multipoligon yang dapat berpotongan pada jumlah titik yang tak terbatas. Secara khusus, ini menyiratkan bahwa bagian dalam dua poligon tidak dapat berpotongan dan bahwa mereka hanya dapat menyentuh pada sejumlah titik yang terbatas.

Poligon kosong dalam multipoligon tidak membatalkan multipoligon.

Multipoligon tidak dapat berisi titik kosong.

Koleksi geometri

Kumpulan geometri dianggap valid jika salah satu kondisi berikut benar:

- Koleksi geometri kosong; artinya, tidak mengandung geometri apa pun.
- Semua geometri dalam kumpulan geometri yang tidak kosong valid.

Definisi ini masih berlaku, meskipun secara rekursif, untuk koleksi geometri bersarang.

Kumpulan geometri dapat berisi titik kosong dan multipoin dengan titik kosong.

Kesederhanaan geometris

Algoritma geometris yang digunakan oleh Amazon Redshift mengasumsikan bahwa geometri input adalah geometri yang valid. Jika input ke algoritma tidak valid, maka pemeriksaan kesederhanaan tidak ditentukan. Bagian berikut menjelaskan definisi kesederhanaan geometris yang digunakan oleh Amazon Redshift untuk setiap subtipe geometri.

Poin

Poin yang valid dianggap sederhana jika salah satu dari kondisi berikut benar:

- Poin yang valid selalu dianggap sederhana.
- Titik kosong dianggap sederhana.

Linestring

Linestring yang valid dianggap sederhana jika salah satu dari kondisi berikut benar:

- Linestring kosong.
- Linestring tidak kosong dan semua kondisi berikut benar:
  - Ini tidak memiliki poin berturut-turut duplikat.
  - Ia tidak memiliki persimpangan diri, kecuali mungkin untuk titik pertama dan titik terakhirnya, yang dapat bertepatan. Dengan kata lain, linestring tidak dapat memiliki persimpangan sendiri kecuali pada titik batas.

Polygon

Poligon yang valid dianggap sederhana jika tidak mengandung titik berturut-turut duplikat.

Multipoint

Multipoint yang valid dianggap sederhana jika salah satu dari kondisi berikut benar:

- Multipoint kosong; artinya, tidak mengandung poin.
- Tidak ada dua titik nonkosong dari multipoint yang bertepatan.

Multilinestring

Multilinestring yang valid dianggap sederhana jika salah satu dari kondisi berikut benar:

- Multilinestring kosong.
- Multilinestring tidak kosong dan semua kondisi berikut benar:
  - Semua garis lengkapnya sederhana.
  - Setiap dua garis dari multilinestring tidak berpotongan, kecuali pada titik-titik yang merupakan titik batas dari dua garis.

Multilinestring tidak kosong yang hanya terdiri dari linestring kosong dianggap kosong.

Garis kosong dalam multilinestring tidak mempengaruhi kesederhanaannya.

Garis tertutup dalam multilinestring tidak dapat berpotongan dengan linestring lain di multilinestring.

Multilinestring tidak dapat memiliki linestring dengan titik berturut-turut duplikat.

Multipoligon

Multipoligon yang valid dianggap sederhana jika tidak mengandung titik berturut-turut duplikat.

Koleksi geometri

Kumpulan geometri yang valid dianggap sederhana jika salah satu dari kondisi berikut benar:

- Koleksi geometri kosong; artinya, tidak mengandung geometri apa pun.
- Semua geometri dalam koleksi geometri yang tidak kosong sederhana.

Definisi ini masih berlaku, meskipun secara rekursif, untuk koleksi geometri bersarang.

H3

H3 adalah sistem grid pengindeksan geospasial hierarkis, yang menawarkan cara untuk mengindeks koordinat spasial hingga resolusi meter persegi. Data yang diindeks dapat digabungkan di seluruh kumpulan data yang berbeda dan dikumpulkan pada tingkat presisi yang berbeda. H3 memungkinkan berbagai algoritma dan pengoptimalan berdasarkan grid, termasuk tetangga terdekat, jalur terpendek, penghalusan gradien, dan banyak lagi. Indeks H3 mengacu pada sel yang dapat berupa segi enam atau segi lima. Ruang dibagi lagi secara hierarkis dengan resolusi. H3 mendukung 16 resolusi dari 0-15, inklusif. Dengan `0` menjadi yang paling kasar dan `15` menjadi yang terbaik.

Amazon Redshift menyediakan fungsi spasial H3 berikut:

- H3_Batas
- H3_Pusat
- H3_ FromLongLat
- H3_ FromPoint
- H3_ IsValid
- H3_Polyfill
- Resolusi H3_
- H3_ ToChildren
- H3_ ToParent

Pertimbangan saat menggunakan data spasial dengan Amazon Redshift

Berikut ini adalah pertimbangan saat menggunakan data spasial dengan Amazon Redshift:

- Ukuran maksimum `GEOGRAPHY` objek `GEOMETRY` atau adalah 1.048.447 byte.
- Amazon Redshift Spectrum tidak mendukung data spasial secara asli. Oleh karena itu, Anda tidak dapat membuat atau mengubah tabel eksternal dengan `GEOGRAPHY` kolom `GEOMETRY` atau.

- Tipe data untuk fungsi yang ditentukan pengguna Python (UDF) tidak mendukung tipe GEOGRAPHY data GEOMETRY atau.
- Anda tidak dapat menggunakan GEOGRAPHY kolom GEOMETRY atau sebagai kunci pengurutan atau kunci distribusi untuk tabel Amazon Redshift.
- Anda tidak dapat menggunakan GEOGRAPHY kolom GEOMETRY atau dalam klausa ORDER BY SQL, GROUP BY, atau DISTINCT.
- Anda tidak dapat menggunakan GEOMETRY atau GEOGRAPHY kolom di banyak fungsi SQL.
- Anda tidak dapat melakukan operasi UNLOAD pada GEOMETRY atau GEOGRAPHY kolom ke dalam setiap format. Anda dapat MEMBONGKAR GEOMETRY atau GEOGRAPHY kolom ke teks atau file nilai yang dipisahkan koma (CSV). Melakukan ini menulis GEOMETRY atau GEOGRAPHY data dalam format EWKB heksadesimal. Jika ukuran data EWKB lebih dari 4 MB, maka peringatan terjadi karena data nantinya tidak dapat dimuat ke dalam tabel.
- Pengkodean kompresi GEOMETRY atau GEOGRAPHY data yang didukung adalah RAW.
- Saat menggunakan driver JDBC atau ODBC, gunakan pemetaan tipe yang disesuaikan. Dalam hal ini, aplikasi klien harus memiliki informasi tentang parameter `ResultSet` objek GEOMETRY atau GEOGRAPHY objek mana. `ResultSetMetadata`Operasi mengembalikan tipeVARCHAR.
- Untuk menyalin tanggal geografis dari a`SHAPEFILE`, pertama-tama masukkan ke dalam GEOMETRY kolom, lalu masukkan objek ke GEOGRAPHY objek.

Fungsi nonspasial berikut dapat menerima input tipe GEOMETRY atauGEOGRAPHY, atau kolom tipe GEOMETRY atau: GEOGRAPHY

- Fungsi agregat COUNT
- Ekspresi kondisional COALESCE dan NVL
- Ekspresi CASE
- Pengkodean default untuk GEOMETRY dan GEOGRAPHY adalah RAW. Lihat informasi yang lebih lengkap di Pengkodean kompresi.

Mengkueri data dengan kueri federasi di Amazon Redshift

Dengan menggunakan kueri federasi di Amazon Redshift, Anda dapat menanyakan dan menganalisis data di seluruh database operasional, gudang data, dan danau data. Dengan fitur Federated Query, Anda dapat mengintegrasikan kueri dari Amazon Redshift pada data langsung di database eksternal dengan kueri di seluruh lingkungan Amazon Redshift dan Amazon S3 Anda. Kueri federasi dapat bekerja dengan database eksternal di Amazon RDS untuk PostgreSQL, Amazon Aurora PostgreSQL-Compatible Edition, Amazon RDS untuk MySQL, dan Amazon Aurora Edition. MySQL-Compatible

Anda dapat menggunakan kueri federasi untuk menggabungkan data langsung sebagai bagian dari aplikasi intelijen bisnis (BI) dan pelaporan. Misalnya, untuk mempermudah penyerapan data ke Amazon Redshift, Anda dapat menggunakan kueri federasi untuk melakukan hal berikut:

- Kueri database operasional secara langsung.
- Terapkan transformasi dengan cepat.
- Muat data ke dalam tabel target tanpa perlu pipeline ekstrak, transformasi, muat (ETL) yang kompleks.

Untuk mengurangi pergerakan data melalui jaringan dan meningkatkan kinerja, Amazon Redshift mendistribusikan sebagian komputasi untuk kueri federasi langsung ke database operasional jarak jauh. Amazon Redshift juga menggunakan kapasitas pemrosesan paralelnya untuk mendukung menjalankan kueri ini, sesuai kebutuhan.

Saat menjalankan kueri federasi, Amazon Redshift pertama-tama membuat koneksi klien ke instans DB cluster RDS atau Aurora DB dari node pemimpin untuk mengambil metadata tabel. Dari simpul komputasi, Amazon Redshift mengeluarkan subkueri dengan predikat ditekan ke bawah dan mengambil baris hasil. Amazon Redshift kemudian mendistribusikan baris hasil di antara node komputasi untuk diproses lebih lanjut.

Rincian tentang kueri yang dikirim ke database Amazon Aurora PostgreSQL atau database Amazon RDS untuk PostgreSQL dicatat dalam tampilan sistem. SVL_FEDERATED_QUERY

Topik

- Memulai dengan menggunakan kueri federasi ke PostgreSQL
- Memulai menggunakan kueri federasi ke PostgreSQL dengan AWS CloudFormation
- Memulai dengan menggunakan kueri federasi ke MySQL

- Membuat peran rahasia dan IAM untuk menggunakan kueri federasi
- Contoh menggunakan kueri federasi
- Perbedaan tipe data antara Amazon Redshift dan database PostgreSQL dan MySQL yang didukung
- Pertimbangan saat mengakses data federasi dengan Amazon Redshift

Memulai dengan menggunakan kueri federasi ke PostgreSQL

Untuk membuat kueri federasi, Anda mengikuti pendekatan umum ini:

1. Siapkan konektivitas dari cluster Amazon Redshift Anda ke instans DB Amazon RDS atau Aurora PostgreSQL Anda.

   Untuk melakukan ini, pastikan instans RDS PostgreSQL atau Aurora PostgreSQL DB Anda dapat menerima koneksi dari cluster Amazon Redshift Anda. Sebaiknya cluster Amazon Redshift dan instans Amazon RDS atau Aurora PostgreSQL berada di cloud pribadi virtual (VPC) dan grup subnet yang sama. Dengan cara ini, Anda dapat menambahkan grup keamanan untuk cluster Amazon Redshift ke aturan masuk grup keamanan untuk instans DB RDS atau Aurora PostgreSQL Anda.

   Anda juga dapat mengatur peering VPC atau jaringan lain yang memungkinkan Amazon Redshift membuat koneksi ke instans RDS atau Aurora PostgreSQL Anda. Untuk informasi selengkapnya tentang jaringan VPC, lihat berikut ini.

   - Apa itu peering VPC? dalam Panduan Pe ering Amazon VPC
   - Bekerja dengan instance DB di VPC di Panduan Pengguna Amazon RDS

   > **Note**
   >
   > Ada beberapa kasus di mana Anda harus mengaktifkan perutean VPC yang disempurnakan: Misalnya, jika cluster Amazon Redshift Anda berada di VPC yang berbeda dari instans RDS atau Aurora PostgreSQL Anda, atau jika mereka berada di VPC yang sama dan rute Anda memerlukannya. Jika tidak, Anda mungkin menerima kesalahan batas waktu saat menjalankan kueri federasi.

2. Siapkan rahasia AWS Secrets Manager untuk database RDS PostgreSQL dan Aurora PostgreSQL Anda. Kemudian referensi rahasia dalam kebijakan dan peran akses AWS Identity and Access

Management (IAM). Untuk informasi selengkapnya, lihat Membuat peran rahasia dan IAM untuk menggunakan kueri federasi.

> **Note**
>
> Jika cluster Anda menggunakan routing VPC yang disempurnakan, Anda mungkin perlu mengonfigurasi antarmuka titik akhir VPC. AWS Secrets Manager Ini diperlukan jika VPC dan subnet cluster Amazon Redshift Anda tidak memiliki akses ke titik akhir publik. AWS Secrets Manager Saat Anda menggunakan titik akhir antarmuka VPC, komunikasi antara cluster Amazon Redshift di VPC Anda dan AWS Secrets Manager dialihkan secara pribadi dari VPC Anda ke antarmuka titik akhir. Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

3. Terapkan peran IAM yang Anda buat sebelumnya ke cluster Amazon Redshift. Untuk informasi selengkapnya, lihat Membuat peran rahasia dan IAM untuk menggunakan kueri federasi.
4. Hubungkan ke database RDS PostgreSQL dan Aurora PostgreSQL Anda dengan skema eksternal. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL. Untuk contoh tentang cara menggunakan kueri federasi, lihatContoh menggunakan kueri federasi.
5. Jalankan kueri SQL Anda yang merujuk skema eksternal yang mereferensikan database RDS PostgreSQL dan Aurora PostgreSQL Anda.

Memulai menggunakan kueri federasi ke PostgreSQL dengan AWS CloudFormation

Anda dapat menggunakan kueri federasi untuk membuat kueri di seluruh database operasional. Dalam panduan memulai ini, Anda dapat mengotomatiskan penyiapan dengan menggunakan tumpukan sampel AWS CloudFormation untuk mengaktifkan kueri federasi dari cluster Amazon Redshift ke database tanpa server Aurora PostgreSQL. Anda dapat memulai dan menjalankan dengan cepat tanpa harus menjalankan pernyataan SQL untuk menyediakan sumber daya Anda.

Tumpukan membuat skema eksternal, mereferensikan instance Aurora PostgreSQL Anda, yang mencakup tabel dengan data sampel. Anda dapat menanyakan tabel dalam skema eksternal dari cluster Redshift Anda.

Jika sebaliknya Anda ingin memulai dengan kueri federasi dengan menjalankan pernyataan SQL untuk mengatur skema eksternal, tanpa menggunakan CloudFormation, lihatMemulai dengan menggunakan kueri federasi ke PostgreSQL.

Sebelum menjalankan CloudFormation tumpukan untuk kueri federasi, pastikan Anda memiliki database tanpa server Amazon Aurora PostgreSQL-Compatible Edition dengan API Data diaktifkan. Anda dapat mengaktifkan API Data di properti database. Jika Anda tidak dapat menemukan pengaturan, periksa kembali apakah Anda menjalankan instance tanpa server Aurora PostgreSQL. Pastikan juga bahwa Anda memiliki cluster Amazon Redshift yang menggunakan node RG atau RA3. Kami menyarankan agar cluster Redshift dan instance Aurora PostgreSQL tanpa server berada di cloud pribadi virtual (VPC) dan grup subnet yang sama. Dengan cara ini, Anda dapat menambahkan grup keamanan untuk cluster Amazon Redshift ke aturan masuk grup keamanan untuk instance database Aurora PostgreSQL Anda.

Untuk informasi selengkapnya tentang memulai menyiapkan cluster Amazon Redshift, lihat Mem ulai dengan gudang data yang disediakan Amazon Redshift. Untuk informasi selengkapnya tentang menyiapkan sumber daya dengan CloudFormation, lihat Apa itu AWS CloudFormation? . Untuk informasi selengkapnya tentang menyiapkan database cluster Aurora DB, lihat Membuat cluster Aurora DB Serverless v1 DB cluster.

Meluncurkan CloudFormation tumpukan untuk kueri federasi Redshift

Gunakan prosedur berikut untuk meluncurkan CloudFormation tumpukan Anda untuk Amazon Redshift guna mengaktifkan kueri federasi. Sebelum melakukannya, pastikan Anda telah menyiapkan cluster Amazon Redshift dan instans Aurora PostgreSQL tanpa server Anda.

Untuk meluncurkan CloudFormation tumpukan Anda untuk kueri federasi

1. Pilih Luncurkan tumpukan CFN di sini untuk meluncurkan CloudFormation layanan di Konsol Manajemen AWS.

   Jika Anda diminta, masuk.

   Proses pembuatan tumpukan dimulai, merujuk file CloudFormation template, yang disimpan di Amazon S3. CloudFormation Template adalah file teks dalam format JSON yang menyatakan AWS sumber daya yang membentuk tumpukan.
2. Pilih Berikutnya untuk memasukkan detail tumpukan.
3. Di bawah Param eter, untuk cluster, masukkan yang berikut ini:

   - Nama cluster Amazon Redshift, misalnya **`ra3-consumer-cluster`**
   - Nama database tertentu, misalnya **`dev`**
   - Nama pengguna database, misalnya **`consumeruser`**

Juga masukkan parameter untuk database cluster Aurora DB, termasuk pengguna, nama database, port, dan titik akhir. Sebaiknya gunakan cluster uji dan menguji database tanpa server, karena tumpukan membuat beberapa objek database.

Pilih Berikutnya.

Opsi tumpukan muncul.

4. Pilih Berikutnya untuk menerima pengaturan default.
5. Di bawah Kemampuan, pilih Saya mengakui bahwa AWS CloudFormation mungkin membuat sumber daya IAM.
6. Pilih Buat tumpukan.

Pilih Buat tumpukan. CloudFormation menyediakan sumber daya template, yang memakan waktu sekitar 10 menit, dan membuat skema eksternal.

Jika terjadi kesalahan saat tumpukan dibuat, lakukan hal berikut:

- Lihat tab CloudFormation Acara untuk informasi yang dapat membantu Anda mengatasi kesalahan.
- Pastikan Anda memasukkan nama, nama database, dan nama pengguna database yang benar untuk cluster Redshift. Periksa juga parameter untuk instance Aurora PostgreSQL.
- Pastikan cluster Anda memiliki node RG atau RA3.
- Pastikan database dan cluster Redshift Anda berada di subnet dan grup keamanan yang sama.

Mengkueri data dari skema eksternal

Untuk menggunakan prosedur berikut, pastikan Anda memiliki izin yang diperlukan untuk menjalankan kueri pada cluster dan database yang dijelaskan.

Untuk menanyakan database eksternal dengan kueri federasi

1. Hubungkan ke database Redshift yang Anda masukkan saat membuat tumpukan, menggunakan alat klien seperti editor kueri Redshift.
2. Kueri untuk skema eksternal yang dibuat oleh tumpukan.

```
select * from svv_external_schemas;
```

T SVV_EXTERNAL_SCHEMAS ampilan mengembalikan informasi tentang skema eksternal yang tersedia. Dalam hal ini, skema eksternal yang dibuat oleh tumpukan dikembalikan,`myfederated_schema`. Anda mungkin juga memiliki skema eksternal lain yang dikembalikan, jika Anda memiliki pengaturan. Tampilan juga mengembalikan database skema terkait. Basis data adalah database cluster Aurora DB yang Anda masukkan saat membuat tumpukan. Tumpukan menambahkan tabel ke database cluster Aurora DB, disebut`category`, dan tabel lain disebut`sales`.

3. Jalankan kueri SQL pada tabel dalam skema eksternal yang mereferensikan database Aurora PostgreSQL Anda. Contoh berikut menunjukkan query.

```
SELECT count(*) FROM myfederated_schema.category;
```

`category`Tabel mengembalikan beberapa catatan. Anda juga dapat mengembalikan catatan dari `sales` tabel.

```
SELECT count(*) FROM myfederated_schema.sales;
```

Untuk contoh lainnya, lihat Contoh menggunakan kueri federasi.

Memulai dengan menggunakan kueri federasi ke MySQL

Untuk membuat kueri federasi ke database MySQL, Anda mengikuti pendekatan umum ini:

1. Siapkan konektivitas dari cluster Amazon Redshift Anda ke instans Amazon RDS atau Aurora MySQL DB Anda.

   Untuk melakukan ini, pastikan bahwa instance RDS MySQL atau Aurora MySQL DB Anda dapat menerima koneksi dari cluster Amazon Redshift Anda. Sebaiknya cluster Amazon Redshift dan instans Amazon RDS atau Aurora MySQL berada di cloud pribadi virtual (VPC) dan grup subnet yang sama. Dengan cara ini, Anda dapat menambahkan grup keamanan untuk cluster Amazon Redshift ke aturan masuk grup keamanan untuk instans RDS atau Aurora MySQL DB Anda.

Anda juga dapat mengatur peering VPC atau jaringan lain yang memungkinkan Amazon Redshift membuat koneksi ke instance RDS atau Aurora MySQL Anda. Untuk informasi selengkapnya tentang jaringan VPC, lihat berikut ini.

- Apa itu peering VPC? dalam Panduan Pe ering Amazon VPC
- Bekerja dengan instance DB di VPC di Panduan Pengguna Amazon RDS

> **Note**
>
> Jika cluster Amazon Redshift Anda berada di VPC yang berbeda dari instans RDS atau Aurora MySQL Anda, aktifkan perutean VPC yang ditingkatkan. Jika tidak, Anda mungkin menerima kesalahan batas waktu saat menjalankan kueri federasi.

2. Siapkan rahasia AWS Secrets Manager untuk database RDS MySQL dan Aurora MySQL Anda. Kemudian referensi rahasia dalam kebijakan dan peran akses AWS Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat Membuat peran rahasia dan IAM untuk menggunakan kueri federasi.

   > **Note**
   >
   > Jika cluster Anda menggunakan perutean VPC yang disempurnakan, Anda mungkin perlu mengonfigurasi antarmuka titik akhir VPC. AWS Secrets Manager Ini diperlukan jika VPC dan subnet cluster Amazon Redshift Anda tidak memiliki akses ke titik akhir publik. AWS Secrets Manager Saat Anda menggunakan titik akhir antarmuka VPC, komunikasi antara cluster Amazon Redshift di VPC Anda dan AWS Secrets Manager dialihkan secara pribadi dari VPC Anda ke antarmuka titik akhir. Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

3. Terapkan peran IAM yang Anda buat sebelumnya ke cluster Amazon Redshift. Untuk informasi selengkapnya, lihat Membuat peran rahasia dan IAM untuk menggunakan kueri federasi.
4. Hubungkan ke database RDS MySQL dan Aurora MySQL Anda dengan skema eksternal. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL. Untuk contoh tentang cara menggunakan kueri federasi, lihatContoh menggunakan query federasi dengan MySQL.
5. Jalankan kueri SQL Anda yang merujuk skema eksternal yang mereferensikan database RDS MySQL dan Aurora MySQL Anda.

Membuat peran rahasia dan IAM untuk menggunakan kueri federasi

Langkah-langkah berikut menunjukkan cara membuat rahasia dan peran IAM untuk digunakan dengan kueri federasi.

Prasyarat

Pastikan Anda memiliki prasyarat berikut untuk membuat peran rahasia dan IAM untuk digunakan dengan kueri federasi:

- Instans RDS PostgreSQL, Aurora PostgreSQL DB, RDS MySQL, atau instance Aurora MySQL DB dengan nama pengguna dan otentikasi kata sandi.
- Cluster Amazon Redshift dengan versi pemeliharaan cluster yang mendukung kueri federasi.

Untuk membuat rahasia (nama pengguna dan kata sandi) dengan AWS Secrets Manager

1. Masuk ke konsol Secrets Manager dengan akun yang memiliki instans cluster RDS atau Aurora DB Anda.
2. Pilih Simpan rahasia baru.
3. Pilih ubin Cre dentials for RDS database. Untuk Nama Pengguna dan Kata Sandi, masukkan nilai untuk instance Anda. Konfirmasikan atau pilih nilai untuk kunci enkripsi. Kemudian pilih database RDS yang akan diakses rahasia Anda.

   > **Note**
   >
   > Sebaiknya gunakan kunci enkripsi default (`DefaultEncryptionKey`). Jika Anda menggunakan kunci enkripsi khusus, peran IAM yang digunakan untuk mengakses rahasia harus ditambahkan sebagai pengguna kunci.

4. Masukkan nama untuk rahasia, lanjutkan dengan langkah-langkah pembuatan dengan pilihan default, lalu pilih Store.
5. Lihat rahasia Anda dan catat nilai ARN Rahasia yang Anda buat untuk mengidentifikasi rahasia tersebut.

Untuk membuat kebijakan keamanan menggunakan rahasia

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Buat kebijakan dengan JSON yang mirip dengan berikut ini.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:my-rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
}
```

   Untuk mengambil rahasia, Anda perlu daftar dan membaca tindakan. Kami menyarankan Anda membatasi sumber daya ke rahasia spesifik yang Anda buat. Untuk melakukan ini, gunakan Nama Sumber Daya Amazon (ARN) dari rahasia untuk membatasi sumber daya. Anda juga dapat menentukan izin dan sumber daya menggunakan editor visual di konsol IAM.
3. Beri nama kebijakan dan selesaikan pembuatannya.

4. Arahkan ke peran IAM.
5. Buat peran IAM untuk Redshift - Dapat Disesuaikan.
6. Melampirkan kebijakan IAM yang baru saja Anda buat ke peran IAM yang sudah ada, atau buat peran IAM baru dan lampirkan kebijakan.
7. Pada tab Relasi kepercayaan pada peran IAM Anda, konfirmasikan bahwa peran tersebut berisi entitas `redshift.amazonaws.com` kepercayaan.
8. Perhatikan Peran ARN yang Anda buat. ARN ini memiliki akses ke rahasia.

Untuk melampirkan peran IAM ke cluster Amazon Redshift Anda

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster. Cluster untuk akun Anda di Wil AWS ayah saat ini terdaftar.
3. Pilih nama cluster dalam daftar untuk melihat detail selengkapnya tentang cluster.
4. Untuk T ind akan, pilih Kelola peran IAM. Halaman Kelola peran IAM muncul.
5. Tambahkan peran IAM Anda ke cluster.

Contoh menggunakan kueri federasi

Contoh berikut menunjukkan cara menjalankan kueri federasi. Jalankan SQL menggunakan klien SQL Anda yang terhubung ke database Amazon Redshift.

Contoh menggunakan kueri federasi dengan PostgreSQL

Contoh berikut menunjukkan cara mengatur kueri federasi yang mereferensikan database Amazon Redshift, database Aurora PostgreSQL, dan Amazon S3. Contoh ini menggambarkan cara kerja kueri federasi. Untuk menjalankannya di lingkungan Anda sendiri, ubah agar sesuai dengan lingkungan Anda. Untuk prasyarat untuk melakukan ini, lihatMemulai dengan menggunakan kueri federasi ke PostgreSQL.

Buat skema eksternal yang mereferensikan database Aurora PostgreSQL.

```
CREATE EXTERNAL SCHEMA apg
FROM POSTGRES
DATABASE 'database-1' SCHEMA 'myschema'
URI 'endpoint to aurora hostname'
```

```
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

Buat skema eksternal lain yang mereferensikan Amazon S3, yang menggunakan Amazon Redshift Spectrum. Juga, berikan izin untuk menggunakan skema ke`public`.

```
CREATE EXTERNAL SCHEMA s3
FROM DATA CATALOG
DATABASE 'default' REGION 'us-west-2'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-S3';

GRANT USAGE ON SCHEMA s3 TO public;
```

Tampilkan jumlah baris di tabel Amazon Redshift.

```
SELECT count(*) FROM public.lineitem;

  count
----------
25075099
```

Tampilkan jumlah baris dalam tabel Aurora PostgreSQL.

```
SELECT count(*) FROM apg.lineitem;

count
-------
11760
```

Tampilkan jumlah baris di Amazon S3.

```
SELECT count(*) FROM s3.lineitem_1t_part;

   count
------------
6144008876
```

Buat tampilan tabel dari Amazon Redshift, Aurora PostgreSQL, dan Amazon S3. Tampilan ini digunakan untuk menjalankan kueri federasi Anda.

```
CREATE VIEW lineitem_all AS
  SELECT
 l_orderkey,l_partkey,l_suppkey,l_linenumber,l_quantity,l_extendedprice,l_discount,l_tax,l_retu
        l_shipdate::date,l_commitdate::date,l_receiptdate::date,
 l_shipinstruct ,l_shipmode,l_comment
  FROM s3.lineitem_1t_part
  UNION ALL SELECT * FROM public.lineitem
  UNION ALL SELECT * FROM apg.lineitem
     with no schema binding;
```

Tampilkan jumlah baris dalam tampilan `lineitem_all` dengan predikat untuk membatasi hasil.

```
SELECT count(*) from lineitem_all WHERE l_quantity = 10;

   count
-----------
123373836
```

Cari tahu berapa banyak penjualan satu item yang ada di bulan Januari setiap tahun.

```
SELECT extract(year from l_shipdate) as year,
       extract(month from l_shipdate) as month,
       count(*) as orders
FROM lineitem_all
WHERE extract(month from l_shipdate) = 1
AND l_quantity < 2
GROUP BY 1,2
ORDER BY 1,2;

 year | month | orders
------+-------+---------
 1992 |     1 |  196019
 1993 |     1 | 1582034
 1994 |     1 | 1583181
 1995 |     1 | 1583919
 1996 |     1 | 1583622
 1997 |     1 | 1586541
 1998 |     1 | 1583198
 2016 |     1 |   15542
 2017 |     1 |   15414
 2018 |     1 |   15527
```

```
 2019 |     1 |     151
```

Contoh penggunaan nama kasus campuran

Untuk menanyakan database jarak jauh PostgreSQL yang didukung yang memiliki nama kasus campuran dari database, skema, tabel, atau kolom, lalu setel ke. `enable_case_sensitive_identifier` `true` Untuk informasi selengkapnya tentang parameter sesi ini, lihatenable_case_sensitive_identifier.

```
SET enable_case_sensitive_identifier TO TRUE;
```

Biasanya, nama database dan skema dalam huruf kecil. Contoh berikut menunjukkan bagaimana Anda dapat terhubung ke database jarak jauh PostgreSQL yang didukung yang memiliki nama huruf kecil untuk database dan skema dan nama huruf campuran untuk tabel dan kolom.

Buat skema eksternal yang mereferensikan database Aurora PostgreSQL yang memiliki nama database huruf kecil (`dblower`) dan nama skema huruf kecil (). `schemalower`

```
CREATE EXTERNAL SCHEMA apg_lower
FROM POSTGRES
DATABASE 'dblower' SCHEMA 'schemalower'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

Dalam sesi di mana kueri berjalan, setel `enable_case_sensitive_identifier` ke`true`.

```
SET enable_case_sensitive_identifier TO TRUE;
```

Jalankan query federasi untuk memilih semua data dari database PostgreSQL. Tabel (`MixedCaseTab`) dan kolom (`MixedCaseName`) memiliki nama kasus campuran. Hasilnya adalah satu baris (`Harry`).

```
select * from apg_lower."MixedCaseTab";
```

```
 MixedCaseName
-------
 Harry
```

Contoh berikut menunjukkan bagaimana Anda dapat terhubung ke database jarak jauh PostgreSQL yang didukung yang memiliki nama kasus campuran untuk database, skema, tabel, dan kolom.

Set `enable_case_sensitive_identifier` el ke `true` sebelum Anda membuat skema eksternal. Jika `enable_case_sensitive_identifier` tidak diset `true` el ke sebelum membuat skema eksternal, maka kesalahan database tidak ada terjadi.

Buat skema eksternal yang mereferensikan database Aurora PostgreSQL yang memiliki nama database campuran (`UpperDB`) dan skema (). `UpperSchema`

```
CREATE EXTERNAL SCHEMA apg_upper
FROM POSTGRES
DATABASE 'UpperDB' SCHEMA 'UpperSchema'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

Jalankan query federasi untuk memilih semua data dari database PostgreSQL. Tabel (`MixedCaseTab`) dan kolom (`MixedCaseName`) memiliki nama kasus campuran. Hasilnya adalah satu baris (`Harry`).

```
select * from apg_upper."MixedCaseTab";
```

```

 MixedCaseName
-------
 Harry
```

Contoh menggunakan query federasi dengan MySQL

Contoh berikut menunjukkan cara mengatur kueri federasi yang mereferensikan database Aurora MySQL. Contoh ini menggambarkan cara kerja kueri federasi. Untuk menjalankannya di lingkungan Anda sendiri, ubah agar sesuai dengan lingkungan Anda. Untuk prasyarat untuk melakukan ini, lihat[Memulai dengan menggunakan kueri federasi ke MySQL](#).

Contoh ini tergantung pada prasyarat berikut:

- Sebuah rahasia yang diatur di Secrets Manager untuk database Aurora MySQL. Rahasia ini dirujuk dalam kebijakan dan peran akses IAM. Untuk informasi selengkapnya, lihat Membuat peran rahasia dan IAM untuk menggunakan kueri federasi.
- Grup keamanan yang diatur menghubungkan Amazon Redshift dan Aurora MySQL.

Buat skema eksternal yang mereferensikan database Aurora MySQL.

```
CREATE EXTERNAL SCHEMA amysql
FROM MYSQL
DATABASE 'functional'
URI 'endpoint to remote hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

Jalankan contoh SQL select dari tabel Aurora MySQL untuk menampilkan satu baris dari tabel karyawan di Aurora MySQL.

```
SELECT level FROM amysql.employees LIMIT 1;

 level
-------
     8
```

Perbedaan tipe data antara Amazon Redshift dan database PostgreSQL dan MySQL yang didukung

Tabel berikut menunjukkan pemetaan tipe data Amazon Redshift ke tipe data Amazon RDS PostgreSQL atau Aurora PostgreSQL yang sesuai.

Jenis data Amazon Redshift	Jenis data RDS PostgreSQL atau Aurora PostgreSQL	Deskripsi
SMALLINT	SMALLINT	Bilangan bulat dua byte bertanda

Jenis data Amazon Redshift	Jenis data RDS PostgreSQL atau Aurora PostgreSQL	Deskripsi
INTEGER	INTEGER	Bilangan bulat empat byte bertanda
BIGINT	BIGINT	Bilangan bulat delapan byte bertanda
DECIMAL	DECIMAL	Numerik persis dari presisi yang dapat dipilih
REAL	REAL	Angka floating-point presisi tunggal
DOUBLE PRECISION	DOUBLE PRECISION	Angka floating-point presisi ganda
BOOLEAN	BOOLEAN	Logika Boolean () true/false
CHAR	CHAR	Fixed-length string karakter
VARCHAR	VARCHAR	Variable-length string karakter dengan batas yang ditentukan pengguna
DATE	DATE	Tanggal kalender (tahun, bulan, hari)
TIMESTAMP	TIMESTAMP	Tanggal dan waktu (tanpa zona waktu)
TIMESTAMPTZ	TIMESTAMPTZ	Tanggal dan waktu (dengan zona waktu)

Jenis data Amazon Redshift	Jenis data RDS PostgreSQL atau Aurora PostgreSQL	Deskripsi
GEOMETRY	GEOMETRI PostGIS	Data spasial

Tipe data RDS PostgreSQL dan Aurora PostgreSQL berikut dikonversi ke VARCHAR (64K) di Amazon Redshift:

- JSON, JSONB
- Array
- SEDIKIT, SEDIKIT BERVARIASI
- BYTEA
- Jenis komposit
- Jenis tanggal dan waktu INTERVAL, WAKTU, WAKTU DENGAN ZONA WAKTU
- Jenis yang disebutkan
- Jenis moneter
- Jenis alamat jaringan
- Tipe numerik SERIAL, BIGSERIAL, SMALLSERIAL, dan MONEY
- Jenis pengidentifikasi objek
- jenis pg_lsn
- Pseudotipe
- Jenis rentang
- Jenis pencarian teks
- TXID_SNAPSHOT
- UUID
- Jenis XML

Tabel berikut menunjukkan pemetaan tipe data Amazon Redshift ke tipe data Amazon RDS MySQL atau Aurora MySQL yang sesuai.

Jenis data Amazon Redshift	Jenis data RDS MySQL atau Aurora MySQL	Deskripsi
BOOLEAN	TINYINT(1)	Logical Boolean (benar atau salah)
SMALLINT	TINYINT (TIDAK DITANDATANGANI)	Bilangan bulat dua byte bertanda
SMALLINT	SMALLINT	Bilangan bulat dua byte bertanda
INTEGER	SMALLINT UNSIGNED	Bilangan bulat empat byte bertanda
INTEGER	MEDIUMINT (TIDAK DITANDATANGANI)	Bilangan bulat empat byte bertanda
INTEGER	INT	Bilangan bulat empat byte bertanda
BIGINT	INT UNSIGNED	Bilangan bulat delapan byte bertanda
BIGINT	BIGINT	Bilangan bulat delapan byte bertanda
DECIMAL	BIGINT UNSIGNED	Numerik persis dari presisi yang dapat dipilih
DECIMAL	DESIMAL (M, D)	Numerik persis dari presisi yang dapat dipilih
REAL	FLOAT	Angka floating-point presisi tunggal

Jenis data Amazon Redshift	Jenis data RDS MySQL atau Aurora MySQL	Deskripsi
DOUBLE PRECISION	DOUBLE	Angka floating-point presisi ganda
CHAR	CHAR	Fixed-length string karakter
VARCHAR	VARCHAR	Variable-length string karakter dengan batas yang ditentukan pengguna
DATE	DATE	Tanggal kalender (tahun, bulan, hari)
TIME	TIME	Waktu (tanpa zona waktu)
TIMESTAMP	TIMESTAMP	Tanggal dan waktu (tanpa zona waktu)
TIMESTAMP	DATETIME	Waktu (tanpa zona waktu)
VARCHAR(4)	YEAR	Karakter panjang variabel yang mewakili tahun

Terjadi kesalahan saat data TIME berada di luar jangkauan (00:00:00 — 24:00:00).

Jenis data RDS MySQL dan Aurora MySQL berikut dikonversi ke VARCHAR (64K) di Amazon Redshift:

- BIT
- BINARY
- VARBINARY

- GUMPALAN KECIL, GUMPALAN, GUMPALAN SEDANG, GUMPALAN PANJANG
- TEKS KECIL, TEKS, TEKS SEDANG, TEKS PANJANG
- ENUM
- SET
- SPASIAL

Pertimbangan saat mengakses data federasi dengan Amazon Redshift

Beberapa fitur Amazon Redshift tidak mendukung akses ke data federasi. Anda dapat menemukan batasan dan pertimbangan terkait berikut.

Berikut ini adalah batasan dan pertimbangan saat menggunakan kueri federasi dengan Amazon Redshift:

- Kueri federasi mendukung akses baca ke sumber data eksternal. Anda tidak dapat menulis atau membuat objek database di sumber data eksternal.
- Dalam beberapa kasus, Anda mungkin mengakses database cluster Amazon RDS atau Aurora DB di Wilayah yang berbeda AWS dari Amazon Redshift. Dalam kasus ini, Anda biasanya dikenakan latensi jaringan dan biaya penagihan untuk mentransfer data di seluruh Wil AWS ayah. Sebaiknya gunakan database global Aurora dengan titik akhir lokal di Wilayah yang sama AWS dengan cluster Amazon Redshift Anda. Database global Aurora menggunakan infrastruktur khusus untuk replikasi berbasis penyimpanan di dua AWS Wilayah mana pun dengan latensi tipikal kurang dari 1 detik.
- Pertimbangkan biaya mengakses cluster Amazon RDS atau Aurora DB. Misalnya, saat menggunakan fitur ini untuk mengakses cluster Aurora DB, biaya cluster Aurora DB didasarkan pada IOPS.
- Kueri federasi tidak mengaktifkan akses ke Amazon Redshift dari cluster RDS atau Aurora DB.
- Kueri federasi hanya tersedia di Wil AWS ayah di mana Amazon Redshift dan Amazon RDS atau Aurora DB cluster tersedia.
- Kueri federasi saat ini tidak mendukung`ALTER SCHEMA`. Untuk mengubah skema, gunakan `DROP` dan kemudian`CREATE EXTERNAL SCHEMA`.
- Kueri federasi tidak berfungsi dengan penskalaan konkurensi.
- Kueri federasi saat ini tidak mendukung akses melalui pembungkus data asing PostgreSQL.

- Kueri federasi ke RDS MySQL atau Aurora MySQL mendukung isolasi transaksi pada tingkat READ COMMITED.
- Jika tidak ditentukan, Amazon Redshift terhubung ke RDS untuk MySQL atau Aurora MySQL pada port 3306. Konfirmasikan nomor port MySQL sebelum membuat skema eksternal untuk MySQL.
- Jika tidak ditentukan, Amazon Redshift terhubung ke RDS PostgreSQL atau Aurora PostgreSQL pada port 5432. Konfirmasikan nomor port PostgreSQL sebelum membuat skema eksternal untuk PostgreSQL.
- Saat mengambil tipe data TIMESTAMP dan DATE dari MySQL, nilai nol diperlakukan sebagai NULL.
- Jika titik akhir pembaca database cluster Aurora DB digunakan, kesalahan “snapshot tidak valid” dapat terjadi. Ini dapat dihindari dengan salah satu metode berikut:
  - Gunakan titik akhir instans cluster Aurora DB tertentu (alih-alih menggunakan titik akhir cluster cluster Aurora DB). Metode ini menggunakan isolasi transaksi REPEATABLE READ untuk hasil dari database PostgreSQL.
  - Gunakan titik akhir pembaca cluster Aurora DB dan setel `pg_federation_repeatable_read` ke false untuk sesi tersebut. Metode ini menggunakan isolasi transaksi READ COMMITTED untuk hasil dari database PostgreSQL. Untuk informasi selengkapnya tentang titik akhir pembaca cluster Aurora DB, lihat Jenis titik akhir cluster Aurora DB di Panduan Pengguna Amazon Aurora. Untuk informasi tentang `pg_federation_repeatable_read`, lihat pg_federation_berulang_baca.

Berikut ini adalah pertimbangan untuk transaksi saat bekerja dengan kueri federasi ke database PostgreSQL:

- Jika kueri terdiri dari tabel federasi, node pemimpin memulai transaksi BACA HANYA REPEATABLE REPEATABLE READ pada database jarak jauh. Transaksi ini tetap berlaku selama transaksi Amazon Redshift.
- Node pemimpin membuat snapshot dari database jarak jauh dengan memanggil `pg_export_snapshot` dan membuat kunci baca pada tabel yang terpengaruh.
- Node komputasi memulai transaksi dan menggunakan snapshot yang dibuat di node pemimpin untuk mengeluarkan kueri ke database jarak jauh.

Versi database federasi yang didukung

Skema eksternal Amazon Redshift dapat mereferensikan database di RDS PostgreSQL eksternal atau Aurora PostgreSQL. Ketika itu terjadi, batasan ini berlaku:

- Saat membuat skema eksternal yang mereferensikan cluster Aurora DB, database Aurora PostgreSQL harus berada di versi 9.6, atau yang lebih baru.
- Saat membuat skema eksternal yang mereferensikan Amazon RDS, database Amazon RDS PostgreSQL harus berada di versi 9.6, atau yang lebih baru.

Skema eksternal Amazon Redshift dapat mereferensikan database di RDS MySQL eksternal atau Aurora MySQL. Ketika itu terjadi, batasan ini berlaku:

- Saat membuat skema eksternal yang mereferensikan cluster Aurora DB, database Aurora MySQL harus berada di versi 5.6 atau yang lebih baru.
- Saat membuat skema eksternal yang mereferensikan Amazon RDS, database RDS MySQL harus berada di versi 5.6 atau yang lebih baru.

Izin federasi Amazon Redshift

Izin federasi Amazon Redshift menyederhanakan pengelolaan izin di beberapa gudang data Redshift sehingga Anda dapat menentukan izin data sekali dan secara otomatis memberlakukannya di semua gudang di gudang Anda. Akun AWS Ini menghilangkan kebutuhan untuk mendefinisikan ulang, mengelola izin, dan kebijakan kontrol akses halus di beberapa gudang.

Saat Anda mendaftarkan gudang namespace/cluster Redshift AWS Glue Data Catalog, semua database dari gudang terdaftar secara otomatis namespaces/clusters dipasang di setiap gudang, memberikan penemuan data yang mulus tanpa konfigurasi manual.

Anda menentukan izin pada objek database menggunakan perintah Redshift SQL yang sudah dikenal, menentukan identitas global melalui AWS Identity and Access Management (IAM) atau. AWS IAM Identity Center Izin ini disimpan bersama dengan data gudang dan diberlakukan secara konsisten terlepas dari gudang mana yang mengeksekusi kueri.

Konsep Utama

- Gudang Redshift dengan izin federasi: Gudang produsen yang terdaftar dengan Katalog Data dan tempat data dan izin Redshift disimpan.
- Mengkonsumsi gudang Redshift: Setiap gudang yang menanyakan data dari gudang jarak jauh. Gudang yang dikonsumsi secara opsional dapat diaktifkan untuk izin federasi Redshift.
- Identitas global: IAM dan IAM Identity Center menyediakan identitas global di semua gudang yang diaktifkan untuk izin federasi Redshift. Pengguna melakukan otentikasi sekali melalui penyedia identitas mereka yang ada dan menerima akses yang konsisten berdasarkan identitas global mereka, terlepas dari gudang mana yang mereka sambungkan.
- Auto-mounting: Semua gudang yang diaktifkan untuk izin federasi Redshift secara otomatis terlihat di semua gudang dalam akun Anda. Kemampuan pemasangan otomatis ini memungkinkan penemuan katalog dan database untuk analitik lintas gudang.
- Propagasi identitas: Saat Anda menjalankan kueri lintas gudang, Redshift menyebarkan identitas global Anda (peran IAM atau pengguna Pusat Identitas IAM) ke gudang jarak jauh.
- Otorisasi di seluruh gudang: Izin federasi Redshift yang diaktifkan gudang jarak jauh memvalidasi izin Anda untuk kueri lintas gudang dan diberlakukan di gudang yang dikonsumsi.
- Kontrol akses berbutir halus: Kebijakan untuk Keamanan Tingkat Baris (RLS), Kebijakan Tingkat Kolom (CLP), dan Masking Data Dinamis (DDM) yang dapat diterapkan di seluruh gudang.

Manfaat

Administrasi yang disederhanakan

- Tentukan izin sekali di gudang
- Secara otomatis menerapkan izin yang sama di semua gudang yang dikonsumsi
- Hilangkan kebutuhan untuk mendefinisikan ulang, mengelola izin, dan kebijakan kontrol akses yang halus di beberapa gudang
- Mengurangi overhead administratif dan potensi kesalahan konfigurasi

Peningkatan keamanan dan kepatuhan

- Menjaga penegakan kebijakan keamanan yang konsisten di semua gudang
- Menerapkan kontrol akses berbutir halus di tabel, dan tingkat kolom
- Izin audit dari gudang mana pun
- Perkakas kepatuhan yang ditingkatkan dengan perintah SHOW tambahan

Peningkatan pengalaman pengguna

- Daftar sekali dan tidak perlu membuat berbagi data secara manual
- Masuk tunggal di semua gudang dan akses yang konsisten berdasarkan identitas global
- Penemuan namespace yang mulus tanpa konfigurasi katalog manual
- Tidak perlu mengelola akun pengguna lokal terpisah di setiap gudang

Skalabilitas horizontal

- Tambahkan gudang baru tanpa meningkatkan kompleksitas tata kelola
- Gudang konsumsi baru secara otomatis menegakkan kebijakan izin
- Analis segera melihat semua database dari gudang terdaftar

Kasus penggunaan

Isolasi beban kerja dengan tata kelola terpadu

Pisahkan sumber daya komputasi untuk beban kerja yang berbeda (ETL, analitik, pelaporan) sambil mempertahankan kebijakan keamanan yang konsisten di semua gudang.

Multi-team akses data

Aktifkan beberapa tim untuk mengakses data bersama dari gudang mereka sendiri dengan kontrol akses yang sesuai diberlakukan secara otomatis.

Arsitektur mesh data

Menerapkan pendekatan data mesh di mana beberapa sumber daya komputasi independen beroperasi pada data bersama dengan tata kelola terpadu.

Optimalisasi biaya

Skalakan sumber daya komputasi secara independen untuk kasus penggunaan yang berbeda sambil mempertahankan manajemen izin terpusat.

Pertimbangan saat menggunakan izin federasi Amazon Redshift

Berikut ini adalah pertimbangan dan batasan untuk berbagi data Amazon Redshift dengan AWS Glue Data Catalog menggunakan izin federasi. Untuk informasi umum tentang pertimbangan dan batasan berbagi data, lihat Per timbangan saat menggunakan berbagi data di Amazon Redshift.

Fitur ini hanya didukung dengan versi cluster 197 dan seterusnya.

Wilayah yang Tidak Didukung

- Africa (Cape Town)
- Asia Pasifik (Hyderabad)
- Europe (Milan)
- Eropa (Spanyol)
- Timur Tengah (UAE)

Persyaratan Lingkungan

Instans Redshift terdaftar dan konsumen harus memenuhi persyaratan berikut:

- Jenis instans: Cluster yang disediakan RG atau RA3 atau kelompok kerja tanpa server
- Wilayah: Sama Wilayah AWS
- Akun: Sama Akun AWS
- Enkripsi: Diaktifkan
- Tingkat isolasi: Isolasi snapshot

Objek yang tidak didukung

Instans konsumen tidak dapat mengakses objek berikut dari katalog izin federasi:

- UDF SQL, UDF Python, dan UDF Lambda
- Model ML
- Skema eksternal dibuat pada instance terdaftar

Coarse-grained Pembatasan Kontrol Akses

Grant hanya didukung pada tabel, database, skema, fungsi yang digunakan dengan notasi 3dot

Fine-grained pembatasan kontrol akses

Selain pembatasan kebijakan Row-level Keamanan (RLS) dan Dynamic Data Masking (DDM) standar di Amazon Redshift, instans konsumen tidak dapat mengakses objek yang dilindungi RLS atau DDM dari katalog izin federasi jika kebijakan berisi fungsi-fungsi sistem berikut:

- `user_is_member_of`
- `role_is_member_of`
- `user_is_member_of_role`

Catatan: Dalam rilis Redshift saat ini, metadata tabel terkait FGAC yang diakses saat menggunakan Gudang Redshift sementara terlihat di katalog.

Penemuan Metadata

- Perintah SHOW didukung untuk Kolom, Tabel, Prosedur Tersimpan, Fungsi, dan Parameter.

Formasi Danau

- Izin Lake Formation tidak didukung pada objek di katalog izin federasi Amazon Redshift.

Identitas

- Hanya pengguna yang terdaftar dengan IAM atau yang AWS IAM Identity Center dapat menanyakan objek di katalog izin federasi Amazon Redshift.
- Saat Amazon Redshift Cluster atau Amazon Redshift Serverless Namespace terdaftar dengan izin federasi Amazon Redshift, Anda tidak dapat mengelola tata kelola data untuk pengguna federasi IAM menggunakan grup federasi IAM. Ini termasuk kontrol akses granular yang dikonfigurasi sebelumnya pada objek melalui grup federasi IAM.
- Saat mendaftarkan Amazon Redshift Cluster atau Amazon Redshift Serverless Namespace yang ada dengan katalog izin federasi Amazon Redshift, semua AWS IAM Identity Center pengguna federasi, termasuk mereka yang sebelumnya memiliki akses, harus secara eksplisit diberikan hak CONNECT untuk mengakses cluster atau kelompok kerja. Untuk informasi selengkapnya tentang pemberian hak akses CONNECT, lihatthe section called “Hubungkan hak istimewa”.
- AWS Pengguna Federasi IAM yang terhubung ke cluster atau grup kerja Amazon Redshift menggunakan tag utama dan kredentif IAM sementara tidak dikenali sebagai identitas global dan tidak dapat mengakses katalog izin federasi Amazon Redshift. Hanya pengguna AWS IAM Identity Center federasi dan pengguna atau peran federasi AWS IAM yang diizinkan untuk menanyakan katalog izin federasi Amazon Redshift.
- Saat Amazon Redshift Cluster atau Amazon Redshift Serverless namespace terdaftar dengan izin federasi Amazon Redshift, batasan perintah GRANT berikut berlaku untuk pengguna atau peran federasi dan pengguna atau peran AWS IAM Identity Center federasi IAM: AWS
  - Anda tidak dapat memberikan peran federasi kepada pengguna atau peran apa pun. Satu pengecualian untuk aturan ini adalah Anda dapat memberikan peran database Redshift kepada pengguna federasi IAM.
  - Anda tidak dapat memberikan peran apa pun ke peran atau pengguna federasi. Satu pengecualian untuk aturan ini adalah Anda dapat memberikan peran yang ditentukan sistem kepada pengguna atau peran federasi.

Akses Mesin

- Akses dari mesin selain Redshift tidak didukung

Ubah Identitas Global Set Pengguna

- Didukung hanya pada “Pilih”, “Hapus”, “Perbarui”, “Tampilkan”, “Sisipkan”
- Peran IAM yang terkait dengan pengguna dengan menggunakan ALTER USER SET GLOBAL IDENTITY hanya digunakan ketika kueri terhadap Redshift Warehouse dengan Izin Federasi dan hanya ketika kueri menargetkan relasi, seperti kueri SELECT, UDPATE, dan DELETE.
- Peran IAM tersebut juga digunakan KUERI SHOW DATABASE, SHOW SCHEMAS dan SHOW TABLES terhadap sumber daya di Redshift Warehouse dengan Izin Federasi.
- Peran IAM tersebut tidak digunakan pada kueri definisi data seperti CREATE, ALTER dan DROP.

Pesan Kesalahan

- Setiap operasi yang tidak didukung terhadap database di katalog Amazon Redshift Federated Izin akan menampilkan kesalahan berikut:

```
Operation is not supported through datashares
```

Prasyarat

Persyaratan kebijakan IAM untuk pengaturan izin federasi Amazon Redshift

Dengan izin federasi Amazon Redshift, Anda dapat mengelola akses data secara terpusat di seluruh beban kerja analitik Anda, dengan izin yang dikelola oleh gudang Redshift secara langsung.

Untuk mengaktifkan izin federasi Amazon Redshift, izin IAM tertentu diperlukan di luar izin standar yang diperlukan untuk membuat cluster yang disediakan Redshift dan ruang nama tanpa server.

Untuk gudang yang disediakan Redshift untuk mengaktifkan izin federasi Redshift:

- `redshift:ModifyLakehouseConfiguration`
- `redshift:RegisterNamespace`

Untuk gudang Redshift Serverless untuk mengaktifkan izin federasi Redshift:

- `redshift-serverless:UpdateLakehouseConfiguration`
- `redshift:RegisterNamespace`

Untuk AWS Glue Data Catalog integrasi untuk membuat katalog dengan izin federasi Redshift:

- `glue:CreateCatalog`
- `glue:GetCatalog`

Untuk pendaftaran sumber daya Lake Formation sebagai pendaftaran satu kali untuk mengaktifkan federasi izin Redshift dari gudang jarak jauh dengan izin federasi:

- `lakeformation:RegisterResource`
- `lakeformation:RegisterResourceWithPrivilegedAccess`

Konfigurasi aplikasi Pusat Identitas IAM untuk gudang Redshift dengan izin federasi

Amazon Redshift mendukung propagasi identitas pusat identitas untuk meneruskan identitas pengguna dengan mulus antara instans Redshift AWS Lake Formation dan AWS Glue layanan. Kemampuan ini memerlukan konfigurasi aplikasi iDC khusus.

Izin IAM yang Diperlukan

Untuk membuat dan mengelola aplikasi pusat identitas untuk propagasi identitas pusat identitas, pastikan izin IAM Anda menyertakan izin berikut:

Untuk manajemen aplikasi Amazon Redshift iDC:

- `redshift:CreateRedshiftIdcApplication`
- `redshift:ModifyRedshiftIdcApplication`
- `redshift:DescribeRedshiftIdcApplications`

Untuk manajemen aplikasi iDC Lake Formation:

- `lakeformation:CreateLakeFormationIdentityCenterConfiguration`
- `lakeformation:DescribeLakeFormationIdentityCenterConfiguration`
- `lakeformation:UpdateLakeFormationIdentityCenterConfiguration`

Buat aplikasi dan konfigurasi iDC yang sesuai

Untuk membuat propagasi identitas untuk beban kerja analitik Anda, buat aplikasi Amazon Redshift iDC jenis Lakehouse. Ini mengelola izin tanpa memerlukan penugasan pengguna eksplisit. Gudang

Redshift yang ditautkan ke aplikasi ini memerlukan hak CONNECT bagi pengguna iDC untuk mengotentikasi koneksi.

Anda hanya dapat membuat satu aplikasi Amazon Redshift iDC tipe Lakehouse per. Akun AWS Aplikasi ini menangani propagasi identitas di semua gudang Redshift yang terintegrasi dengan Lake Formation dan AWS Glue layanan. Aplikasi ini hanya dapat digunakan dengan gudang Redshift yang terdaftar di AWS Glue Data Catalog.

Siapkan peran IAM yang diasumsikan oleh Redshift dan digunakan oleh propagasi identitas iDC

Pembuatan aplikasi Redshift Lakehouse iDC memerlukan peran IAM dari akun Anda dengan izin IAM tertentu. Peran IAM Anda yang digunakan dalam Aplikasi iDC Redshift Anda harus memiliki hubungan kepercayaan berikut untuk memungkinkan Redshift mengasumsikannya dan menetapkan konteks untuk propagasi identitas iDC.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "redshift-serverless.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetContext"
            ]
        }
    ]
}
```

Dan di bawah izin untuk peran IAM iDC Anda untuk mendukung propagasi identitas iDC.

- AmazonRedshiftFederatedAuthorizationKebijakan ini memungkinkan Amazon Redshift untuk menanyakan AWS Glue Data Catalog database melalui izin federasi.
- AWSIDC Tetapkan Kebijakan Konteks

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "sts:SetContext"
           ],
           "Resource": "*"
       }
   ]
}
```

- Menyiapkan Redshift sebagai aplikasi ter AWS kelola dengan AWS IAM Identity Center.
- Kebijakan SSO IAM pusat identitas AWSIDC:
  - `sso:DescribeApplication`- Diperlukan untuk membuat entri penyedia identitas (IdP) di katalog.
  - `sso:DescribeInstance`- Digunakan untuk membuat peran atau pengguna federasi IdP secara manual.

```
{
  "Sid": "VisualEditor1",
  "Effect": "Allow",
  "Action": [
    "sso:DescribeApplication",
    "sso:DescribeInstance"
  ],
  "Resource": [
    "arn:aws:sso:::instance/<IAM Identity Center Instance ID>",
    "arn:aws:sso::<AWS-account-id>:application/<IAM Identity Center Instance ID>/*"
  ]
}
```

Buat aplikasi Redshift iDC tipe Lakehouse baru

CLI

Buat aplikasi Lakehouse iDC Anda dengan menentukan jenis aplikasi Lakehouse dalam `create-redshift-idc-application` permintaan, yang menghilangkan kebutuhan akan penugasan

pengguna eksplisit di Pusat Identitas sambil mengaktifkan persyaratan `CONNECT` hak istimewa untuk otentikasi pengguna iDC:

```
aws redshift create-redshift-idc-application \
--idc-instance-arn <your_idc_instance_arn> \
--idc-display-name '<name_of_idc_application_display_on_idc_console>' \
--iam-role-arn <idc_carrier_role_arn> \
--application-type Lakehouse \
--redshift-idc-application-name '<name_of_idc_display_on_redshift_console>' \
--service-integrations '[
        {
            "LakeFormation":[
                {
                    "LakeFormationQuery":{"Authorization": "Enabled"}
                }
            ]
        },
        {
            "Redshift":[
                {
                    "Connect" : {
                        "Authorization": "Enabled"
                    }
                }
            ]
        }
    ]'
```

Konfigurasi ini memungkinkan penyebaran identitas tepercaya antara Redshift dan Lake Formation, memungkinkan pengguna mengakses data di seluruh layanan menggunakan kredentif Pusat Identitas mereka tanpa penetapan izin tambahan.

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke halaman koneksi Pusat Identitas IAM dan pilih Buat aplikasi.
3. Konfigurasikan pengaturan aplikasi idc umum Anda.
4. Pilih Konfigurasikan izin federasi Amazon Redshift menggunakan AWS IAM Identity Center (Direkomendasikan) untuk mengatur jenis aplikasi.
5. Integrasi propagasi identitas Lake Formation dan Redshift connect diaktifkan secara default.

6. Lengkapi pengaturan cluster yang tersisa dan pilih Buat aplikasi.

Ubah aplikasi Redshift iDC yang ada

Jika Anda memiliki aplikasi Redshift iDC yang sudah ada yang tidak mengaktifkan integrasi layanan yang diperlukan, Anda dapat memperbaruinya untuk mendukung propagasi identitas antara layanan dan. clusters/namespaces

CLI

Gunakan `modify-redshift-idc-application` perintah untuk mengaktifkan `LakeFormation:query` otorisasi dan otor `Redshift:Connect` isasi. Integrasi ini penting untuk propagasi identitas iDC lintas layanan dan lintas cluster:

```
aws redshift modify-redshift-idc-application \
--redshift-idc-application-arn '<arn_of_the_target_redshift_idc_application>' \
--service-integrations '[
        {
            "LakeFormation":[
                {
                    "LakeFormationQuery":{"Authorization": "Enabled"}
                }
            ]
        },
        {
            "Redshift":[
                {
                    "Connect" : {
                        "Authorization": "Enabled"
                    }
                }
            ]
        }
    ]'
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke halaman koneksi Pusat Identitas IAM dan pilih aplikasi IDC yang ada yang ingin Anda edit.

3. Pilih Integrasi propagasi identitas untuk mengaktifkan dan mengonfigurasi pengaturan lain dan pilih S impan perubahan.

Buat konfigurasi pusat identitas Lake Formation

Layanan Formasi Danau Anda memerlukan aplikasi iDC khusus jika belum dibuat. Anda juga harus mengaktifkan `Redshift:Connect` otorisasi agar konfigurasi berfungsi dengan baik.

CLI

Gunakan `create-lake-formation-identity-center-configuration` perintah untuk mengaktifkan `Redshift:Connect` otorisasi. Integrasi ini sangat penting untuk Lake Formation menyebarkan identitas iDC ke cluster Redshift dan Redshift Serverless Namespace

```
aws lakeformation  create-lake-formation-identity-center-configuration \
--instance-arn <your_idc_instance_arn> \
--service-integrations '[{
  "Redshift": [{
    "RedshiftConnect": {
      "Authorization": "ENABLED"
    }
  }]
}]'
```

Console

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Formasi Danau di https://console.aws.amazon.com/lakeformation/.
2. Di panel navigasi kiri, pilih integrasi Pusat Identi tas IAM.
3. Di halaman integrasi Pusat Identitas IAM, Anda dapat mengaktifkan propagasi identitas tepercaya untuk Amazon Redshift connect. Lake Formation menyebarkan identitas ke hilir berdasarkan izin yang efektif, sehingga aplikasi resmi dapat mengakses data atas nama pengguna.

Perbarui konfigurasi pusat identitas Lake Formation

Jika Anda telah mengonfigurasi aplikasi iDC Lake Formation yang tidak mengaktifkan integrasi layanan yang diperlukan, Anda dapat memperbaruinya untuk mendukung propagasi identitas antara layanan dan. clusters/namespaces

CLI

Gunakan `update-lake-formation-identity-center-configuration` perintah untuk mengaktifkan `Redshift:Connect` otorisasi. Integrasi ini penting untuk propagasi identitas iDC lintas layanan dan lintas cluster:

```
aws lakeformation update-lake-formation-identity-center-configuration \
--service-integrations '[{

  "Redshift": [{
    "RedshiftConnect": {
      "Authorization": "ENABLED"
    }
  }]
}]'
```

Console

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Formasi Danau di https://console.aws.amazon.com/lakeformation/.
2. Di panel navigasi kiri, pilih integrasi Pusat Identi tas IAM.
3. Di halaman integrasi Pusat Identitas IAM, Anda dapat mengaktifkan propagasi identitas tepercaya untuk Amazon Redshift connect. Lake Formation menyebarkan identitas ke hilir berdasarkan izin yang efektif, sehingga aplikasi resmi dapat mengakses data atas nama pengguna.

Prasyarat Formasi Danau

Pelanggan memerlukan `CREATE_CATALOG` izin Lake Formation untuk mengaktifkan AWS Glue Data Catalog dengan izin federasi Amazon Redshift.

1. Jika akun tersebut milik pelanggan Lake Formation yang ada, administrator Lake Formation harus secara eksplisit memberikan izin CREATE_CATALOG kepada setiap pembuat cluster. Gunakan contoh perintah CLI berikut:

```
aws lakeformation grant-permissions \
    --cli-input-json \
    '{
        "Principal": {
```

```
            "DataLakePrincipalIdentifier": "<PrincipalArn>"
        },
        "Resource": {
            "Catalog": {}
        },
        "Permissions": [
            "CREATE_CATALOG"
        ]
    }'
```

2. Jika akun belum pernah menggunakan Formasi Danau, verifikasi bahwa Pembuat Katalog disetel ke halaman IAMAllowedPrincipals Peran dan Tugas Administratif di konsol Formasi Danau. Jika tidak dikonfigurasi, atur Administrator Danau Data dengan mengikuti Buat administrator danau data. Atau, Anda dapat membuat Administrator Danau Data dengan kebijakan minimum yang diperlukan jika Anda hanya akan menggunakannya AWS Glue Data Catalog dengan izin federasi Amazon Redshift.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "lakeformation:PutDataLakeSettings",
                "lakeformation:GrantPermissions",
                "lakeformation:GetDataLakeSettings",
                "lakeformation:BatchGrantPermissions",
                "lakeformation:ListPermissions"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

3. Mintal DataLake ah Admin memberikan izin IAMAllowedPrincipals untuk Membuat Katalog. Akses dapat diberikan melalui tombol Grant untuk Pembuat Katalog di halaman Peran dan Tugas Administratif.

Konfigurasi jaringan untuk routing VPC yang ditingkatkan

Jika routing VPC yang disempurnakan diaktifkan, VPC Anda harus dapat menjangkau layanan Pusat Identitas AWS IAM. Jika tidak, propagasi identitas gagal. Untuk informasi selengkapnya tentang persyaratan jaringan, lihat Menggunakan Pusat Identitas AWS IAM dengan routing VPC yang disempurnakan.

Hubungkan hak istimewa

Sebagai bagian dari izin federasi Amazon Redshift, Amazon Redshift telah memperkenalkan hak akses CONNECT untuk mengelola akses pengguna AWS IAM Identity Center federasi ke grup kerja atau cluster Amazon Redshift. Fitur ini tersedia saat izin federasi Amazon Redshift diaktifkan pada workgroup atau cluster.

Hak istimewa ini memungkinkan administrator untuk mengontrol akses pengguna melalui izin terperinci di setiap kelompok kerja atau cluster Amazon Redshift di mana izin federasi Amazon Redshift diaktifkan. Administrator Amazon Redshift dapat menentukan pengguna AWS IAM Identity Center atau grup federasi mana yang memiliki akses untuk terhubung langsung ke grup kerja atau cluster Amazon Redshift, memberikan kontrol halus atas akses AWS IAM Identity Center pengguna di setiap kelompok kerja atau cluster.

Sintaksis

```
GRANT CONNECT [ON WORKGROUP]
TO <prefix>:<username> | ROLE <prefix>:<rolename> | PUBLIC;
```

MENGHUBUNGKAN [PADA KELOMPOK KERJA]

Memberikan izin untuk terhubung ke grup kerja. Izin CONNECT hanya berlaku untuk AWS IAM Identity Center identitas (pengguna dan peran).

UNTUK<prefix>: <username>

Menunjukkan pengguna AWS IAM Identity Center federasi yang menerima izin.

UNTUK PERAN<prefix>: <rolename>

Menunjukkan grup AWS IAM Identity Center federasi yang menerima izin.

PUBLIK

Memberikan izin CONNECT kepada semua pengguna AWS IAM Identity Center federasi, termasuk pengguna yang dibuat nanti.

Orientasi

Pendaftaran cluster pergeseran merah

Redshift mendukung pembuatan cluster baru atau memulihkan cluster dari snapshot dengan pendaftaran AWS Glue Data Catalog (GDC). Anda dapat menentukan bagian nama katalog GDC dari pendaftaran ini. Untuk mendukung propagasi identitas iDC, Anda dapat menentukan aplikasi Redshift iDC arn tipe Lakehouse untuk mengaktifkan propagasi identitas iDC.

Buat cluster baru dengan pendaftaran katalog data Glue

CLI

Untuk mendaftarkan cluster yang baru dibuat secara otomatis dengan Katalog Data, berikan nama katalog yang akan digunakan untuk membuat dan mendaftarkan Katalog Data Anda. `redshift-idc-application-arn`Parameternya opsional - sertakan jika Anda ingin menautkan cluster Anda dengan Aplikasi Redshift iDC tipe Lakehouse. Anda juga dapat membuat asosiasi aplikasi iDC ini di lain waktu.

```
aws redshift create-cluster \
    --cluster-identifier 'redshift-cluster' \
   --catalog-name 'glue-data-catalog-name' \
   --redshift-idc-application-arn 'arn:aws:redshift:us-
east-1:012345678912:redshiftidcapplication:3f966e50-f1b7-495c-8ace-bd0d6c3c3b17' \
   --<other_configurations_as_needed>
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke dasbor cluster yang disediakan dan pilih Buat cluster.
3. Konfigurasikan pengaturan cluster umum Anda.
4. Di AWS Glue Data Catalog bagian Daftar dengan, pilih Daftar dengan izin federasi Amazon Redshift.
   - Masukkan pengenal nama katalog.
   - (Disarankan) Pilih izin federasi Amazon Redshift yang digunakan untuk mengasosi AWS IAM Identity Center asikan dengan aplikasi IDC Redshift.
5. Lengkapi pengaturan cluster yang tersisa dan pilih Buat cluster.

Kembalikan cluster baru dengan AWS Glue Data Catalog pendaftaran

CLI

Untuk mengembalikan snapshot ke cluster baru dengan AWS Glue Data Catalog integrasi, berikan nama katalog yang akan digunakan untuk membuat dan mendaftarkan katalog Anda AWS Glue . `redshift-idc-application-arn`Parameternya opsional - sertakan jika Anda ingin menautkan cluster Anda dengan Aplikasi Redshift iDC tipe Lakehouse. Anda juga dapat membuat asosiasi aspplikasi iDC ini di lain waktu.

```
aws redshift restore-from-cluster-snapshot \
   --cluster-identifier 'redshift-cluster' \
   --catalog-name 'glue-data-catalog-name' \
   --snapshot-identifier 'redshift-cluster-snapshot' \
   --redshift-idc-application-arn 'arn:aws:redshift:us-
east-1:012345678912:redshiftidcapplication:3f966e50-f1b7-495c-8ace-bd0d6c3c3b17' \
   --<other_configurations_as_needed>
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke halaman snapshot yang disediakan. Dari tabel snapshot, pilih Puli hkan ke cluster yang disediakan dari menu tarik-turun Puli hkan snapshot.
3. Konfigurasikan pengaturan cluster umum.
4. Di AWS Glue Data Catalog bagian Daftar dengan, pilih Daftar dengan izin federasi Amazon Redshift.
   - Masukkan pengenal nama katalog.
   - (Disarankan) Pilih izin federasi Amazon Redshift yang digunakan untuk mengasosi AWS IAM Identity Center asikan dengan aplikasi IDC Redshift.
5. Lengkapi pengaturan cluster yang tersisa dan pilih Buat cluster.

Ubah cluster yang ada dengan AWS Glue Data Catalog pendaftaran

Jika cluster Redshift Anda sudah dikaitkan dengan Aplikasi Redshift iDC tipe non-lakehouse, hal berikut terjadi selama pendaftaran: AWS Glue Data Catalog

- Jika tidak ada ARN Aplikasi Redshift iDC yang disediakan, Aplikasi Redshift iDC yang ada di katalog Anda akan disetel ke status dinonaktifkan.
- Ketika Aplikasi iDC Redshift tipe Lakehouse dari AWS IAM Identity Center instance yang berbeda ditentukan, penyedia iDC saat ini dinonaktifkan
- Ketika Aplikasi Redshift iDC tipe Lakehouse dari instance yang sama disediakan AWS IAM Identity Center
  - Aplikasi Redshift iDC ARN di katalog Anda akan diubah menjadi ARN dari Aplikasi Redshift iDC tipe Lakehouse. Katalog yang diperbarui dapat diperiksa dengan menanyakan svv_identity_provider. Untuk informasi selengkapnya tentang svv_identity_provider, lihat svv_identity_provider. https://docs.aws.amazon.com/redshift/latest/dg/r_SVV_IDENTITY_PROVIDERS.html
  - AWS IAM Identity Center pengguna federasi yang sebelumnya memiliki akses ke cluster Redshift, harus secara eksplisit diberikan hak akses CONNECT oleh Admin untuk mengakses cluster. Untuk informasi selengkapnya tentang pemberian hak akses CONNECT, lihatthe section called “Hubungkan hak istimewa”.
  - Setelah mendaftar AWS Glue Data Catalog, identitas AWS IAM Identity Center federasi Anda yang ada dan sumber daya milik mereka tetap tidak berubah. Asosiasi namespace untuk identitas federasi ini juga dipertahankan.

CLI

Anda dapat menggunakan `modify-lakehouse-configuration` perintah untuk mendaftarkan cluster Anda ke AWS Glue Data Catalog, yang `catalog-name` digunakan untuk membuat dan mendaftarkan AWS Glue katalog Anda. Untuk mendukung propagasi identitas iDC, tentukan arn jenis rumah danau Anda RedshiftIdcApplication, ini memerlukan Aplikasi iDC Redshift tipe Lakehouse, silakan lihat Membuat aplikasi iDC Redshift tipe Lakehouse baru: Konfigurasi Aplikasi Pusat Identitas untuk Gudang Redshift dengan izin federasi.

```
aws redshift modify-lakehouse-configuration \
    --cluster-identifier 'redshift-cluster' \
    --lakehouse-registration Register \
    --catalog-name 'glue-data-catalog-name' \
    --lakehouse-idc-registration Associate \
    --lakehouse-idc-application-arn 'arn:aws:redshift:us-
east-1:012345678912:redshiftidcapplication:3f966e50-f1b7-495c-8ace-bd0d6c3c3b17' \
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke cluster yang disediakan yang ingin Anda daftarkan dan pilih.
3. Dari halaman detail cluster, pilih Daftar dengan AWS Glue Data Catalog dari menu tarik-turun T indakan.
4. Pilih Daftar dengan opsi izin federasi Amazon Redshift dan
   - Masukkan pengenal nama katalog.
   - (Disarankan) Pilih izin federasi Amazon Redshift yang digunakan AWS IAM Identity Center untuk mengasosiasikan dengan aplikasi IDC Redshift dan pilih Daftar.

Redshift Pendaftaran namespace tanpa server

Redshift Serverless memungkinkan ruang nama Serverless yang terpasang pada kelompok kerja untuk mendaftar. AWS Glue Data Catalog Perhatikan bahwa database Anda akan restart selama pembaruan ini.

Jika Redshift Serverless Namespace Anda sudah dikaitkan dengan Aplikasi iDC Redshift tipe non-lakehouse, hal berikut terjadi selama pendaftaran Katalog Data Glue:

- Jika tidak ada ARN Aplikasi Redshift iDC yang disediakan, Aplikasi Redshift iDC yang ada di katalog Anda akan disetel ke status dinonaktifkan.
- Ketika Aplikasi iDC Redshift tipe Lakehouse dari AWS IAM Identity Center instance yang berbeda ditentukan, penyedia iDC saat ini dinonaktifkan
- Ketika Aplikasi Redshift iDC tipe Lakehouse dari instance yang sama disediakan AWS IAM Identity Center
  - Aplikasi Redshift iDC ARN di katalog Anda akan diubah menjadi ARN dari Aplikasi Redshift iDC tipe Lakehouse. Katalog yang diperbarui dapat diperiksa dengan menanyakan svv_identity_provider. Untuk informasi selengkapnya tentang svv_identity_provider, lihat svv_identity_provider. https://docs.aws.amazon.com/redshift/latest/dg/r_SVV_IDENTITY_PROVIDERS.html
  - AWS IAM Identity Center pengguna federasi yang sebelumnya memiliki akses ke cluster Redshift, harus secara eksplisit diberikan hak akses CONNECT oleh Admin untuk mengakses

cluster. Untuk informasi selengkapnya tentang pemberian hak akses CONNECT, lihatthe section called “Hubungkan hak istimewa”.

- Setelah mendaftar AWS Glue Data Catalog, identitas AWS IAM Identity Center federasi Anda yang ada dan sumber daya milik mereka tetap tidak berubah. Asosiasi namespace untuk identitas federasi ini juga dipertahankan.

CLI

Anda dapat menggunakan `update-lakehouse-configuration` perintah untuk mendaftarkan namespace Redshift Serverless Anda ke AWS Glue Data Catalog, yang `catalog-name` digunakan untuk membuat dan mendaftarkan katalog lem Anda. Untuk mendukung propagasi identitas iDC, tentukan arn Aplikasi Redshift Idc tipe Lakehouse.

```
aws redshift-serverless update-lakehouse-configuration \
    --namespace-name 'serverless-namespace-name' \
    --lakehouse-registration Register \
    --catalog-name 'glue-data-catalog-name' \
    --lakehouse-idc-registration Associate \
    --lakehouse-idc-application-arn 'arn:aws:redshift:us-
east-1:012345678912:redshiftidcapplication:3f966e50-f1b7-495c-8ace-bd0d6c3c3b17'
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke cluster yang disediakan yang ingin Anda daftarkan dan pilih.
3. Dari halaman detail cluster, pilih Daftar dengan AWS Glue Data Catalog dari menu tarik-turun T indakan.
4. Pilih Daftar dengan opsi izin federasi Amazon Redshift dan
   - Masukkan pengenal nama katalog.
   - (Disarankan) Pilih izin federasi Amazon Redshift yang digunakan AWS IAM Identity Center untuk mengasosiasikan dengan aplikasi IDC Redshift dan pilih Daftar.

Aktifkan AWS IAM Identity Center propagasi identitas

Amazon Redshift mendukung propagasi identitas Identity Center (iDC) untuk meneruskan identitas pengguna iDC dengan mulus antara instans Redshift dan layanan/. AWS Lake FormationAWS Glue

Prasyarat

- Anda telah membuat Aplikasi Amazon Redshift iDC tipe Lakehouse, lihat konfigurasi AWS IAM Identity Center aplikasi untuk gudang Redshift dengan izin federasi.
- Anda memiliki Amazon Redshift Cluster atau Amazon Redshift Serverless Namespace yang terdaftar di. AWS Glue Data Catalog
  - Redshift Serverless Namespace memerlukan workgroup yang terpasang untuk melakukan operasi terkait.

Jika Redshift Cluster atau Redshift Serverless Namespace Anda sudah dikaitkan dengan Aplikasi Redshift iDC dari jenis selain Lakehouse, hal berikut terjadi selama pendaftaran: AWS Glue Data Catalog

- Jika tidak ada ARN Aplikasi Redshift iDC yang disediakan, Aplikasi Redshift iDC yang ada di katalog Anda akan disetel ke status dinonaktifkan.
- Ketika Aplikasi iDC Redshift tipe Lakehouse dari AWS IAM Identity Center instance yang berbeda ditentukan, penyedia iDC saat ini dinonaktifkan
- Ketika Aplikasi Redshift iDC tipe Lakehouse dari instance yang sama disediakan AWS IAM Identity Center
  - Aplikasi Redshift iDC ARN di katalog Anda akan diubah menjadi ARN dari Aplikasi Redshift iDC tipe Lakehouse. Katalog yang diperbarui dapat diperiksa dengan menanyakan svv_identity_provider. Untuk informasi selengkapnya tentang svv_identity_provider, lihat svv_identity_provider. https://docs.aws.amazon.com/redshift/latest/dg/r_SVV_IDENTITY_PROVIDERS.html
  - AWS IAM Identity Center pengguna federasi yang sebelumnya memiliki akses ke cluster Redshift, harus secara eksplisit diberikan hak akses CONNECT oleh Admin untuk mengakses cluster. Untuk informasi selengkapnya tentang pemberian hak akses CONNECT, lihatthe section called “Hubungkan hak istimewa”.
  - Setelah mendaftar AWS Glue Data Catalog, identitas AWS IAM Identity Center federasi Anda yang ada dan sumber daya milik mereka tetap tidak berubah. Asosiasi namespace untuk identitas federasi ini juga dipertahankan.

Aktifkan AWS IAM Identity Center propagasi identitas untuk cluster yang disediakan Amazon Redshift

Untuk Amazon Redshift Provisioned Cluster yang mendaftarkan namespace-nya AWS Glue Data Catalog, aplikasi ini memerlukan Aplikasi iDC Lakehouse Amazon Redshift yang tidak memerlukan penetapan pengguna AWS IAM Identity Center Identitas secara eksplisit ke aplikasi, hak akses masuk pengguna iDC dikelola oleh hak istimewa CONNECT di gudang Redshift.

CLI

Anda dapat menggunakan `modify-lakehouse-configuration` perintah untuk mengaktifkan propagasi identitas iDC untuk cluster Anda dengan izin federasi Redshift, tentukan arn jenis rumah danau Anda RedshiftIdcApplication, ini memerlukan Aplikasi iDC Redshift Lakehouse silakan lihat Buat aplikasi Redshift iDC tipe Lakehouse baru: Konfigurasi Aplikasi Pusat Identitas untuk Gudang Redshift dengan izin federasi.

```
aws redshift modify-lakehouse-configuration \
    --cluster-identifier 'redshift-cluster' \
    --lakehouse-idc-registration Associate \
    --lakehouse-idc-application-arn 'arn:aws:redshift:us-
east-1:012345678912:redshiftidcapplication:3f966e50-f1b7-495c-8ace-bd0d6c3c3b17' \
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke cluster yang disediakan yang ingin Anda daftarkan dan pilih.
3. Dari halaman detail cluster, pilih Daftar dengan AWS Glue Data Catalog dari menu tarik-turun T indakan.
4. Pilih Aktif kan dari izin federasi Amazon Redshift menggunakan AWS IAM Identity Center drop-down untuk mengaitkan aplikasi IDC dan pilih Simpan perubahan.

Aktifkan AWS IAM Identity Center propagasi identitas untuk ruang nama tanpa server Amazon Redshift

CLI

Anda dapat menggunakan `modify-lakehouse-configuration` perintah untuk mengaktifkan propagasi identitas iDC untuk namespace Anda dengan izin federasi Redshift, tentukan arn jenis rumah danau Anda RedshiftIdcApplication, ini memerlukan Aplikasi iDC Redshift Lakehouse silakan lihat Buat aplikasi Redshift iDC tipe Lakehouse baru: Konfigurasi Aplikasi Pusat Identitas untuk Gudang Redshift dengan izin federasi.

```
aws redshift modify-lakehouse-configuration \
    --cluster-identifier 'redshift-cluster' \
    --lakehouse-idc-registration Associate \
    --lakehouse-idc-application-arn 'arn:aws:redshift:us-
east-1:012345678912:redshiftidcapplication:3f966e50-f1b7-495c-8ace-bd0d6c3c3b17' \
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke namespace tanpa server yang ingin Anda edit registrasi dan pilih.
3. Dari halaman detail cluster, pilih Edit AWS Glue Data Catalog registrasi dari menu tarik-turun Tindakan.
4. Pilih Aktif kan dari izin federasi Amazon Redshift menggunakan AWS IAM Identity Center drop-down untuk mengaitkan aplikasi IDC dan pilih Simpan perubahan.

MENGUBAH IDENTITAS GLOBAL SET PENGGUNA

Selain IAM dan AWS IAM Identity Center kredensional, pengguna yang menjalankan kueri terhadap Gudang Redshift dengan izin federasi dapat melakukan otentikasi menggunakan peran IAM. Superuser dapat mengatur peran IAM untuk pengguna non-federasi lain untuk diasosiasikan secara otomatis pada pembuatan sesi, dan peran IAM ini akan diasumsikan saat membuat kueri terhadap Gudang Pergeseran Merah dengan Izin Federasi. Fungsi ini disediakan untuk memungkinkan pengguna iDC untuk men AWS gotentikasi secara non-interaktif.

Fitur ini berguna untuk kasus penggunaan berikut:

- Pelanggan yang memiliki pengaturan besar dan kompleks dengan pengguna gudang lokal yang ada serta pengguna dengan identitas global.
- Pelanggan yang menggunakan iDC, tetapi ingin dapat masuk secara otomatis tanpa tindakan browser interaktif untuk masuk.

Persyaratan dan pembatasan:

- Hanya pengguna super yang dapat mengatur peran IAM dengan`ALTER USER`.
- Peran IAM harus dilampirkan ke cluster.
- Peran IAM harus memiliki izin untuk mengakses sumber daya yang diperlukan untuk menjalankan kueri di gudang Redshift dengan izin federasi. Sebaiknya gunakan kebijakan `AmazonRedshiftFederatedAuthorization` AWS terkelola.
- Pengguna yang melakukan otentikasi dengan menggunakan peran GLOBAL IDENTITY IAM dapat menanyakan tampilan di Gudang Redshift dengan Izin Federasi, tetapi tidak dapat MEMBUAT, MENGUBAH, MENYEGARKAN, atau MELEPASKANNYA.

Sintaksis

Sintaks berikut menjelaskan `ALTER USER SET GLOBAL IDENTITY` perintah yang digunakan untuk menetapkan peran IAM bagi pengguna database non-federasi untuk menjalankan kueri terhadap Gudang Redshift dengan Izin Federasi.

```
ALTER USER username SET
GLOBAL IDENTITY IAM_ROLE 'arn:aws:iam::<AWS-account-id>:role/<role-name>'
```

Sekarang ketika diautentikasi sebagai pengguna target (dengan menghubungkan langsung sebagai*username*, atau dengan menggunakan SET SESSION AUTHORIZATION), Anda dapat memeriksa peran identitas global menggunakan

```
SHOW GLOBAL IDENTITY
```

Catatan, peran identitas global dikaitkan dengan pengguna pada pembuatan sesi. Jika Anda menetapkan identitas global untuk pengguna yang saat ini masuk, pengguna tersebut perlu menyambung kembali agar identitas global berlaku.

Perintah berikut dapat digunakan untuk menghapus peran IAM terkait.

```
ALTER USER username RESET GLOBAL IDENTITY
```

Parameter

nama pengguna

Nama pengguna. Tidak dapat menjadi pengguna federasi, seperti pengguna IAM atau AWS pengguna iDC.

<account-id><role-name>IAM_ROLE 'arn:aws:iam: ::role/ '

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi saat pengguna *username* menjalankan kueri di gudang Redshift dengan izin federasi. Peran ini harus memiliki izin yang diperlukan untuk menjalankan kueri. Sebaiknya gunakan Kebijakan AmazonRedshiftFederatedAuthorization AWS Terkelola.

Mengkueri katalog yang terdaftar di AWS Glue Data Catalog dengan izin federasi Redshift

Saat Anda mendaftarkan gudang data Amazon Redshift ke izin federasi Amazon Redshift AWS Glue Data Catalog menggunakan, database di namespace tersebut secara otomatis dipasang di semua instans Amazon Redshift di wilayah tersebut dan Wilayah. Akun AWS Hal ini memungkinkan query di seluruh lingkungan multi-gudang sambil menjaga keamanan melalui identitas global dan kebijakan kontrol akses halus (FGAC).

Prasyarat

Sebelum menanyakan database federasi, pastikan Anda memiliki:

- Kebijakan ter AmazonRedshiftFederatedAuthorization AWS kelola yang dilampirkan pada pengguna atau peran IAM Anda. Untuk kontrol akses yang halus, Anda dapat memberikan tindakan IAM tertentu dari kebijakan ini alih-alih melampirkan kebijakan lengkap.
- Izin yang tepat dikonfigurasi untuk objek katalog di gudang data sumber
- Akses ke gudang data Amazon Redshift di Wilayah Akun AWS dan Wilayah yang sama.

Mengkueri Database Federasi

Anda dapat menanyakan objek dalam database di bawah katalog terdaftar menggunakan sintaks tiga bagian untuk mengakses tabel:

```
database@namespace_catalog.schema.table
```

Contoh Kueri

```
SELECT * FROM my_database@my_namespace.sales.transactions
WHERE transaction_date >= '2024-01-01';
```

Selain itu, Anda dapat mengakses objek dari database federasi melalui database https://docs.aws.amazon.com/redshift/latest/dg/r_USE_command.html USE.

Database federasi juga tersedia untuk koneksi https://docs.aws.amazon.com/redshift/latest/dg/database-direct-connect.html langsung.

Keamanan dan Kontrol Akses

Identitas Global dan Penegakan FGAC

Saat Anda mengaktifkan gudang data Amazon Redshift dengan Izin Federasi, hibah dan kebijakan FGAC yang ditentukan pada objek ini secara otomatis diberlakukan di gudang yang dikonsumsi. Ini memastikan keamanan yang konsisten di seluruh lingkungan multi-gudang Anda.

Fitur Keamanan yang Didukung

Anda dapat mengonfigurasi kontrol keamanan berikut pada objek di gudang data Amazon Redshift, yang akan diterapkan di semua gudang yang dikonsumsi

- Column-Level Hak istimewa (CLP): Memberikan atau membatasi akses ke kolom tertentu
- Row-Level Keamanan (RLS): Kontrol akses ke baris tertentu berdasarkan atribut pengguna
- Dynamic Data Masking (DDM): Secara otomatis menutupi data sensitif berdasarkan izin pengguna

Mengelola kontrol akses di katalog izin federasi Amazon Redshift

Dengan izin federasi Amazon Redshift, pengguna dapat menentukan kontrol akses berbutir kasar dan halus dari gudang Redshift mana pun di. Akun AWS Coarse-grained izin mengelola akses ke tabel, tampilan, dan objek database, termasuk izin cakupan, sementara kontrol berbutir halus memungkinkan hak istimewa tingkat kolom dan penerapan kebijakan keamanan seperti keamanan tingkat baris (RLS) dan penyembunyian data dinamis (DDM).

Berikan/Cabut

Dengan izin federasi, Anda dapat menentukan izin pada akses tingkat tabel seperti akses untuk membaca, menulis data dalam tabel, dan tampilan di database Perizinan Federasi Redshift dari gudang Redshift mana pun.

```
GRANT SELECT ON "sales_db@finance-catalog".sales_schema.sales_table TO
 "IAMR:sales_analyst";

GRANT INSERT ON "sales_db@finance-catalog".sales_schema.sales_view TO
 "IAMR:sales_data_engineer";

REVOKE UPDATE ON "sales_db@finance-catalog".sales_schema.us_sales_view FROM
 "IAMR:us_sales_analyst";

REVOKE DELETE ON "sales_db@finance-catalog".sales_schema.us_sales_view FROM
 "IAMR:us_sales_analyst";
```

Izin federasi Amazon Redshift mendukung izin cakupan untuk memungkinkan Anda memberikan atau mencabut izin pada semua objek dari suatu jenis dalam database atau skema.

```
GRANT SELECT FOR TABLES IN SCHEMA "sales_db@finance-catalog".sales_schema TO
 "IAMR:sales_manager";

REVOKE UPDATE FOR TABLES IN SCHEMA sales_schmea DATABASE "sales_db@finance-catalog"
 FROM "IAMR:sales_analyst";
```

Anda dapat menentukan pemberian/mencabut izin akses pada database.

```
GRANT CREATE ON DATABASE "sales_db@finance-catalog" TO "IAMR:sales_admin";

REVOKE TEMP ON DATABASE "sales_db@finance-catalog" FROM "IAMR:sales_analyst";
```

Untuk informasi selengkapnya tentang sintaks yang didukung izin federasi Amazon Redshift untuk hibah, lihat. the section called “Izin yang dapat Anda berikan di katalog izin federasi Amazon Redshift”

Fine-grained kontrol akses

Anda dapat menentukan kontrol akses halus atas data sensitif menggunakan kontrol akses tingkat kolom, keamanan tingkat baris (RLS), dan penyamaran data dinamis (DDM) dalam database dengan izin federasi Amazon Redshift. Column-level kontrol akses memungkinkan untuk menentukan hak akses tingkat kolom berbutir lebih halus pada tabel dan tampilan. Pengguna super dan pengguna atau peran dengan `sys:secadmin` peran pada database dapat membuat kebijakan RLS dan DDM, melampirkan kebijakan ini ke relasi tertentu, dan mengaktifkan RLS pada relasi.

Column-level kontrol akses:

Anda dapat memberikan atau mencabut hak istimewa tertentu pada kolom individual tabel atau tampilan.

```
GRANT SELECT ON "sales_db@finance-catalog".sales_schema.sales_table(order_number,
 sales_date, purchase_amount, sale_amount) TO "IAMR:sales_revenue_analyst";

REVOKE UPDATE ON "sales_db@finance-catalog".sales_schema.us_sales_view(order_number,
 sales_date, purchase_amount, sale_amount) FROM "IAMR:sales_revenue_analyst";
```

Row-level keamanan (RLS):

Anda dapat mengaktifkan atau menonaktifkan keamanan tingkat baris untuk relasi.

```
ALTER TABLE "sales_db@finance-catalog".sales_schema.sales_table ROW LEVEL SECURITY ON;
```

Anda dapat membuat, mengubah, menghapus kebijakan RLS pada database.

```
-- Create an RLS policy
CREATE RLS POLICY "sales_db@finance-catalog".policy_america
WITH (region VARCHAR(10))
USING (region = 'USA');

-- Alter an RLS policy
ALTER RLS POLICY "sales_db@finance-catalog".policy_america
USING (region IN ('USA', 'CANADA', 'Mexico'));
```

```
-- Drop an RLS policy
DROP RLS POLICY "sales_db@finance-catalog".policy_america;
```

Anda dapat melampirkan atau melepaskan kebijakan RLS pada relasi.

```
-- Attach an RLS policy
ATTACH RLS POLICY "sales_db@finance-catalog".policy_america
ON "sales_db@finance-catalog".sales_schema.sales_table
TO "IAMR:america_sales_analyst";

-- Detach an RLS policy
DETACH RLS POLICY "sales_db@finance-catalog".policy_america
ON "sales_db@finance-catalog".sales_schema.sales_view_america
FROM "IAMR:global_sales_analyst";
```

Penyembunyian data dinamis (DDM):

Anda dapat membuat, mengubah, menghapus kebijakan masking pada database.

```
-- Create a masking policy
CREATE MASKING POLICY "sales_db@finance-catalog".hash_credit
WITH (credit_card varchar(256))
USING (sha2(credit_card + 'testSalt', 256));

-- Alter an masking policy
ALTER MASKING POLICY "sales_db@finance-catalog".hash_credit
USING (sha2(credit_card + 'otherTestSalt', 256));

-- Drop an masking policy
DROP MASKING POLICY "sales_db@finance-catalog".hash_credit;
```

Anda dapat melampirkan atau melepaskan kebijakan masking pada relasi.

```
-- Attach a masking policy
 ATTACH MASKING POLICY hash_credit
ON "sales_db@finance-catalog".sales_schema.transaction_table (credit_card)
TO "IAMR:sales_analyst" PRIORITY 30;

-- Detach a masking policy
DETACH MASKING POLICY hash_credit
ON "sales_db@finance-catalog".sales_schema.transaction_view (credit_card)
FROM "IAMR:transaction_analyst";
```

Seorang superuser atau pengguna dengan `sys:secadmin` hak istimewa dapat melihat kebijakan RLS dan DDM, serta lampiran mereka pada relasi dengan izin federasi, dengan menggunakan perintah. TAMPILKAN KEBIJAKAN

> **Note**
>
> - Fungsi yang ditentukan pengguna (UDF) di RLS, definisi kebijakan DDM tidak didukung dengan izin federasi Amazon Redshift.
> - Fungsi Redshift SQL user_is_member_of, role_is_member_of, user_is_member_of_role tidak didukung dengan izin federasi Amazon Redshift.

Izin yang dapat Anda berikan di katalog izin federasi Amazon Redshift

Pernyataan SQL untuk mendukung pengelolaan izin pada database dengan izin federasi Amazon Redshift pada sumber daya pada tingkat perincian yang berbeda. Sintaks mendukung referensi tabel lokal (ketika terhubung ke database yang berisi sumber daya atau ketika database sumber dalam USE) dan referensi lintas database yang sepenuhnya memenuhi syarat.

Catatan

- *username*dapat merujuk ke pengguna IAM, peran IAM, atau pengguna iDC ()AWS IAM Identity Center.
- *role_name*dapat merujuk ke grup iDC. Grup IAM tidak didukung.

Izin Berbutir Kasar

```
GRANT { SELECT | INSERT | UPDATE | DELETE | TRUNCATE} ON
[ TABLE ] { table_name | database@catalog.schema_name.table_name }
TO { username | ROLE role_name | PUBLIC }

REVOKE { SELECT | INSERT | UPDATE | DELETE | TRUNCATE } ON
[ TABLE ] { table_name | database@catalog.schema_name.table_name }
FROM { username | ROLE role_name | PUBLIC }
```

Hak istimewa Tingkat Kolom

```
GRANT { { SELECT | UPDATE | DELETE }
```

```
( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ] ( column_name [,...] ) }
ON { table_name | database@catalog.schema_name.table_name }
TO { username | ROLE role_name | PUBLIC }

REVOKE { { SELECT | UPDATE | DELETE }
( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ] ( column_name [,...] ) }
ON { table_name | database@catalog.schema_name.table_name }
FROM { username | ROLE role_name | PUBLIC }
```

Izin Terbatas

```
GRANT { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }
FOR SCHEMAS IN
DATABASE database@catalog
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

GRANT
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [ PRIVILEGES ] } }
FOR TABLES IN
{ SCHEMA schema_name [DATABASE database@catalog ] | DATABASE database@catalog }
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ] { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }

FOR SCHEMAS IN
DATABASE database@catalog
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ] { { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE
 | REFERENCES } [, ...] } | ALL [ PRIVILEGES ] } }
FOR TABLES IN
{ SCHEMA schema_name [ DATABASE database@catalog ] | DATABASE database@catalog }
FROM { username | ROLE role_name } [, ...]
```

Izin federasi Amazon Redshift - contoh ujung ke ujung

Berikut ini adalah contoh ujung ke ujung yang menunjukkan bagaimana Anda dapat membuat dan mengelola kebijakan tata kelola data yang komprehensif menggunakan Izin Federasi Amazon Redshift. Kebijakan ini mencakup Keamanan Tingkat Baris (RLS), Masking Data Dinamis (DDM), dan

Izin Tingkat Kolom (CLP) yang bekerja sama untuk mengontrol akses data berdasarkan peran dan kondisi pengguna.

Anda harus menjadi `superuser` atau memiliki peran sys:secadmin untuk menjalankan contoh ini.

Prasyarat

Contoh-contoh mengasumsikan bahwa peran "IAMR:role_name" di bawah ini sudah ada di akun, jika tidak silakan buat. Selain itu, gudang data Redshift terdaftar AWS Glue Data Catalog dengan nama "catalog_name" dan memiliki database "db_name".

Pada instance katalog berikan sys:secadmin Peran ke Peran IAM yang sesuai

```
-- Grant sys:secadmin role to relevant user (must be run on Redshift catalog instance)

        GRANT ROLE sys:secadmin TO "IAMR:AccountSecurityAdminrole_name";
```

Sisanya akan dijalankan di gudang komputasi Redshift

Jika Anda menggunakan pengguna IAM atau iDC, Anda dapat melewati dua langkah berikut untuk membuat pengguna gudang lokal dan pemetaan identitas global.

Langkah 1: Membuat gudang lokal yang diperlukan pengguna untuk menguji kebijakan tata kelola

```
-- Create test users.
CREATE USER alice WITH PASSWORD 'Alice_pass_1';
CREATE USER oscar WITH PASSWORD 'Oscar_pass_1';
CREATE USER sierra WITH PASSWORD 'Sierra_pass_1';
```

Langkah 2: Menyiapkan Pemetaan Peran Identitas Global IAM

```
-- Map local users to IAM roles (executed by superuser).

-- Make user sierra a sys:secadmin by setting the global identity
-- to `IAMR:AccountSecurityAdmin`.
-- This role has been granted secadmin privilege on Redshift catalog instance.

ALTER USER sierra SET GLOBAL IDENTITY
IAM_ROLE 'arn:aws:iam::123456789012:role/AccountSecurityAdmin';
```

```
ALTER USER alice SET GLOBAL IDENTITY
IAM_ROLE 'arn:aws:iam::123456789012:role/Analyst';

ALTER USER oscar SET GLOBAL IDENTITY
IAM_ROLE 'arn:aws:iam::123456789012:role/Operator';

-- Verify global identity settings.
SET SESSION AUTHORIZATION sierra;
SHOW GLOBAL IDENTITY;

SET SESSION AUTHORIZATION alice;
SHOW GLOBAL IDENTITY;

SET SESSION AUTHORIZATION oscar;
SHOW GLOBAL IDENTITY;

-- Reset to default session.
RESET SESSION AUTHORIZATION;
```

Menyiapkan Lingkungan

Pertama, buat tabel dan isi dengan contoh data pelanggan dan tabel pencarian untuk kebijakan tata kelola.

```
-- Create the main customer table.
CREATE TABLE db_name@catalog_name.public.customers (
    id INTEGER,
    name VARCHAR(50),
    email VARCHAR(100),
    region VARCHAR(20),
    revenue DECIMAL(10,2)
);

-- Populate with sample customer data.
INSERT INTO db_name@catalog_name.public.customers VALUES
(1, 'John Smith', 'john@email.com', 'US', 1000.00),
(2, 'Jane Doe', 'jane@email.com', 'EU', 500.00),
(3, 'Mike Johnson', 'mike@email.com', 'US', 2000.00);

-- Grant basic table access.
GRANT ALL ON db_name@catalog_name.public.customers TO PUBLIC;
```

```
-- Create lookup table for region-based policies.
CREATE TABLE db_name@catalog_name.public.lookup_regions (
    region_code VARCHAR(20),
    allowed BOOLEAN
);

INSERT INTO db_name@catalog_name.public.lookup_regions VALUES
('US', TRUE),
('EU', FALSE),
('APAC', TRUE);

GRANT ALL ON db_name@catalog_name.public.lookup_regions TO PUBLIC;

-- Create lookup table for revenue-based masking.
CREATE TABLE db_name@catalog_name.public.lookup_revenue_tiers (base_revenue INTEGER);

INSERT INTO db_name@catalog_name.public.lookup_revenue_tiers VALUES (1000), (2000);
GRANT ALL ON db_name@catalog_name.public.lookup_revenue_tiers TO PUBLIC;
```

Menyiapkan Izin Tingkat Kolom (CLP)

Konfigurasikan akses tingkat kolom untuk pengguna yang berbeda untuk mengontrol kolom mana yang dapat diakses pengguna.

```
-- Grant specific column access to Analyst i.e. `alice`.
GRANT SELECT (id, region) ON db_name@catalog_name.public.customers
    TO "IAMR:Analyst";

-- Grant different column access to Operator i.e. `oscar`.
GRANT SELECT (id, name, revenue) ON db_name@catalog_name.public.customers
    TO "IAMR:operator";
```

Membuat Kebijakan Keamanan Tingkat Baris (RLS)

Buat kebijakan RLS untuk mengontrol baris mana yang dapat dilihat pengguna berdasarkan hibah dan kondisi data mereka.

```
-- Switch to admin user to create policies.
SET SESSION AUTHORIZATION sierra;

-- Create simple RLS policy: Analysts see only US customers.
CREATE RLS POLICY db_name@catalog_name.us_only
```

```
WITH (region VARCHAR(20))
USING (region = 'US');

-- Attach the policy to the Analyst i.e. `alice`.
ATTACH RLS POLICY db_name@catalog_name.us_only
ON db_name@catalog_name.public.customers
TO "IAMR:Analyst";

-- Enable row level security on the table.
ALTER TABLE db_name@catalog_name.public.customers ROW LEVEL SECURITY ON;

-- Create advanced RLS policy using lookup table.
CREATE RLS POLICY db_name@catalog_name.region_lookup_policy
WITH (region VARCHAR(20)) AS r
USING (r.region IN (
    SELECT region_code
    FROM public.lookup_regions
    WHERE allowed = TRUE
));

-- Attach the lookup-based policy to Operator i.e. `oscar`.
ATTACH RLS POLICY db_name@catalog_name.region_lookup_policy
ON db_name@catalog_name.public.customers
TO "IAMR:Operator";
```

Membuat Kebijakan Dynamic Data Masking (DDM)

Buat kebijakan masking untuk mengaburkan data sensitif berdasarkan peran dan kondisi pengguna.

```
-- Create masking policy for PII data (names and emails).
CREATE MASKING POLICY db_name@catalog_name.mask_pii
WITH (DATA VARCHAR(100))
USING (SHA2(DATA + 'secret', 256)::TEXT);

-- Attach masking to name and email columns for all users.
ATTACH MASKING POLICY db_name@catalog_name.mask_pii
ON db_name@catalog_name.public.customers (name)
TO PUBLIC;

ATTACH MASKING POLICY db_name@catalog_name.mask_pii
ON db_name@catalog_name.public.customers (email)
TO PUBLIC;
```

```
-- Create conditional masking policy for revenue data.
CREATE MASKING POLICY db_name@catalog_name.conditional_mask
WITH (revenue DECIMAL(10,2))
USING (CASE WHEN revenue IN (SELECT base_revenue
                                    FROM public.lookup_revenue_tiers)
            THEN revenue ELSE 0.00 END);

-- Attach conditional masking to Analyst i.e. `alice` with priority.
ATTACH MASKING POLICY db_name@catalog_name.conditional_mask
ON db_name@catalog_name.public.customers (revenue)
TO "IAMR:Analyst"
PRIORITY 20;
```

Available/Applied Kebijakan Audit

Gunakan perintah SHOW untuk mengonfirmasi bahwa kebijakan tata kelola Anda telah dikonfigurasi dengan benar.

```
-- Show all RLS policies in the database.
SHOW RLS POLICIES FROM DATABASE db_name@catalog_name LIMIT 10;

-- Show RLS policies for specific users and tables.
SHOW RLS POLICIES ON db_name@catalog_name.public.customers FOR "IAMR:Analyst" LIMIT 10;

SHOW RLS POLICIES ON db_name@catalog_name.public.customers FOR "IAMR:Operator" LIMIT
 10;
SHOW RLS POLICIES ON db_name@catalog_name.public.customers FOR PUBLIC LIMIT 10;

-- Show all masking policies in the database.
SHOW MASKING POLICIES FROM DATABASE db_name@catalog_name LIMIT 10;

-- Show Masking policies for specific users and tables.
SHOW MASKING POLICIES ON db_name@catalog_name.public.customers FOR "IAMR:Analyst" LIMIT
 10;
SHOW MASKING POLICIES ON db_name@catalog_name.public.customers FOR "IAMR:Operator"
 LIMIT 10;
SHOW MASKING POLICIES ON db_name@catalog_name.public.customers FOR PUBLIC LIMIT 10;
```

Menguji Pola Akses

Uji bagaimana pengguna yang berbeda melihat data berdasarkan peran mereka dan kebijakan yang diterapkan.

```
-- Test as analyst: Only US customers, only id/region columns,
-- conditional revenue masking.
SET SESSION AUTHORIZATION alice;
SELECT id, region FROM db_name@catalog_name.public.customers ORDER BY id;
SELECT id, region, revenue FROM db_name@catalog_name.public.customers ORDER BY id;

-- Test as operator: Allowed regions only, masked names, specific columns.
SET SESSION AUTHORIZATION oscar;
SELECT id, name, revenue FROM db_name@catalog_name.public.customers ORDER BY id;
SELECT id, name, region FROM db_name@catalog_name.public.customers ORDER BY id;

-- Test as admin: Full access to all data.
SET SESSION AUTHORIZATION sierra;
SELECT * FROM db_name@catalog_name.public.customers ORDER BY id;
```

Mengubah Kebijakan

Ubah kebijakan yang ada untuk mengubah perilakunya tanpa membuatnya kembali.

```
-- Switch back to admin user.
SET SESSION AUTHORIZATION sierra;

-- Alter the PII masking policy to use simple string replacement.
ALTER MASKING POLICY db_name@catalog_name.mask_pii
USING ('***MASKED***'::TEXT);

-- Alter the conditional masking policy to use different threshold.
ALTER MASKING POLICY db_name@catalog_name.conditional_mask
USING (CASE WHEN revenue >= 500.00 THEN revenue ELSE -1.00 END);

-- Alter the RLS policy to show only disallowed regions.
ALTER RLS POLICY db_name@catalog_name.region_lookup_policy
USING (r.region IN (
    SELECT region_code
    FROM db_name@catalog_name.public.lookup_regions
    WHERE allowed = FALSE
));
```

Kebijakan Pelepasan dan Penghapusan

Hapus kebijakan saat tidak diperlukan lagi.

```
-- Detach RLS policies from users.
DETACH RLS POLICY db_name@catalog_name.us_only
ON db_name@catalog_name.public.customers
FROM "IAMR:Analyst";

DETACH RLS POLICY db_name@catalog_name.region_lookup_policy
ON db_name@catalog_name.public.customers
FROM "IAMR:Operator";

-- Detach masking policies.
DETACH MASKING POLICY db_name@catalog_name.mask_pii
ON db_name@catalog_name.public.customers (name)
FROM PUBLIC;

DETACH MASKING POLICY db_name@catalog_name.mask_pii
ON db_name@catalog_name.public.customers (email)
FROM PUBLIC;

DETACH MASKING POLICY db_name@catalog_name.conditional_mask
ON db_name@catalog_name.public.customers (revenue)
FROM "IAMR:Analyst";
```

Bersihkan: Matikan kebijakan RLS dan hapus

```
-- Turn off Row-level security.
ALTER TABLE db_name@catalog_name.public.customers ROW LEVEL SECURITY OFF;

-- Drop policies.
DROP RLS POLICY db_name@catalog_name.us_only CASCADE;
DROP RLS POLICY db_name@catalog_name.region_lookup_policy CASCADE;
DROP MASKING POLICY db_name@catalog_name.mask_pii CASCADE;
DROP MASKING POLICY db_name@catalog_name.conditional_mask CASCADE;

-- Drop tables.
DROP TABLE db_name@catalog_name.public.customers;
DROP TABLE db_name@catalog_name.public.lookup_regions;
DROP TABLE db_name@catalog_name.public.lookup_revenue_tiers;
```

Setel ulang identitas global

```
-- TO rest the global identity.
```

```
ALTER USER alice RESET GLOBAL IDENTITY;
ALTER USER oscar RESET GLOBAL IDENTITY;
ALTER USER sierra RESET GLOBAL IDENTITY;
```

Konfigurasi pengguna federasi

Dengan izin federasi Amazon Redshift, pengguna yang diautentikasi dengan kredentif IAM atau IAM Identity Center (iDC) dapat memperoleh pengalaman yang konsisten di semua gudang Amazon Redshift mereka.

Saat pengguna terhubung ke cluster Redshift apa pun, parameter konfigurasi mereka—seperti format tanggal, jalur pencarian, dan preferensi zona waktu—secara otomatis diterapkan ke sesi mereka. Ini memastikan bahwa pengalaman pengguna yang sama dipertahankan di semua cluster, tanpa memerlukan konfigurasi ulang manual, memberikan pengalaman yang mulus dan dipersonalisasi.

Contoh

Pertimbangkan Alex, seorang analis data yang telah mengonfigurasi format tanggal dan zona waktu pilihan mereka di database Redshift utama mereka. Ketika Alex menjalankan kueri di sana, semua nilai tanggal muncul dalam format yang disukai.

Sekarang, ketika Alex terhubung ke database bersama atau dipasang secara otomatis di gudang Redshift lain, preferensi yang sama diterapkan secara konsisten. Alex melihat hasil dalam format tanggal dan zona waktu yang sama, memastikan pengalaman yang konsisten di seluruh lingkungan.

Konfigurasi Pengguna yang konsisten di seluruh Amazon Redshift Warehouse

Konfigurasi tingkat pengguna berikut secara otomatis disinkronkan di seluruh cluster Redshift untuk pengguna yang diautentikasi dengan kredentif IAM atau AWS IAM Identity Center (iDC):

- gaya tanggal
- enable_case_sensitive_identifier
- ekstra_float_digit
- jalur penelusuran
- cast_super_null_on_error
- default_geometri_encoding

- deskripsi_field_name_in_uppercase
- downcase_delimited_identifier
- enable_case_sensitive_super_atribut
- mengaktifkan_pembulatan numerik
- mengaktifkan_hasil_cache_untuk_sesi
- error_pada_pembaruan_nondeterministik
- interval_terban_composite_literal
- json_serialization_enable
- json_serialization_parse_bersarang_string
- mv_enable_aqmv_untuk_sesi
- navigasi_super_null_on_error
- parse_super_null_on_error
- spektrum_enable_pseudo_column
- enable_spectrum_oid
- spectrum_query_maxerror
- menyimpan_proc_log_min_pesan-pesan
- analisis_ambang_persen
- enable_vacuum_boost
- pg_federation_berulang_baca

Manajemen Koneksi dan Konfigurasi

- Saat menghubungkan ke Amazon Redshift menggunakan JDBC, beberapa konfigurasi tingkat sesi juga dapat diterapkan secara otomatis. Untuk detail selengkapnya, lihat dokumentasi konfigurasi tingkat sesi JDBC.
- Pengguna juga dapat menentukan konfigurasi tingkat pengguna persisten menggunakan AL TER USER.

Menonaktifkan AWS IAM Identity Center perambatan

Sebelum Anda dapat menonaktifkan AWS IAM Identity Center propagasi, Anda harus memiliki Amazon Redshift Cluster atau Amazon Redshift Serverless Namespace yang telah terdaftar AWS

Glue Data Catalog dan dikaitkan dengan Aplikasi iDC Lakehouse Redshift. Namespace Amazon Redshift Serverless memerlukan workgroup yang dilampirkan untuk melakukan operasi terkait.

Nonaktifkan AWS IAM Identity Center Propagasi Identitas untuk cluster yang disediakan Amazon Redshift

Saat menonaktifkan Propagasi AWS IAM Identity Center Identitas untuk Amazon Redshift Provisioned Cluster Anda, Aplikasi Lakehouse Redshift iDC yang dilampirkan padanya akan ditandai sebagai dinonaktifkan di cluster.

CLI

Anda dapat menggunakan `modify-lakehouse-configuration` perintah untuk menonaktifkan propagasi identitas iDC untuk cluster Anda dengan Perizinan Federasi Redshift, perhatikan perintah itu tidak menghapus penyedia iDC dari cluster Anda tetapi memasukkannya ke mode dinonaktifkan.

```
aws redshift modify-lakehouse-configuration \
    --cluster-identifier 'redshift-cluster' \
    --lakehouse-idc-registration Disassociate \
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke cluster yang disediakan yang ingin Anda edit pendaftaran dan pilih.
3. Dari halaman detail cluster, pilih Edit AWS Glue Data Catalog registrasi dari menu tarik-turun Tindakan.
4. Pilih N onaktifkan dari izin federasi Amazon Redshift menggunakan AWS IAM Identity Center drop-down untuk memisahkan aplikasi IDC dan pilih Simpan perubahan.

Nonaktifkan AWS IAM Identity Center Propagasi Identitas untuk ruang nama Tanpa Server Amazon Redshift

CLI

Anda dapat menggunakan `modify-lakehouse-configuration` perintah untuk menonaktifkan propagasi identitas iDC untuk namespace Anda dengan Perizinan Federasi Redshift, perhatikan

perintah itu tidak menghapus penyedia iDC dari cluster Anda tetapi memasukkannya ke mode dinonaktifkan.

```
aws redshift-serverless update-lakehouse-configuration \
--namespace-name 'serverless-namespace-name' \
--lakehouse-idc-registration Disassociate \
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke namespace tanpa server yang ingin Anda edit registrasi dan pilih.
3. Dari halaman detail namespace, pilih E dit AWS Glue Data Catalog registrasi dari menu tarik-turun Tind akan.
4. Pilih N onaktifkan dari izin federasi Amazon Redshift menggunakan AWS IAM Identity Center drop-down untuk memisahkan aplikasi IDC dan pilih Simpan perubahan.

Deregister dari AWS Glue Data Catalog

Persyaratan Kebijakan IAM untuk Deregistrasi Izin Federasi Amazon Redshift

Untuk membatalkan pendaftaran cluster atau namespace tanpa server Anda AWS Glue Data Catalog, izin IAM di bawah ini diperlukan.

Untuk Cluster yang Disediakan Redshift

- `redshift:ModifyLakehouseConfiguration`
- `redshift:DregisterNamespace`

Untuk Redshift Serverless

- `redshift-serverless:UpdateLakehouseConfiguration`
- `redshift:DregisterNamespace`

Untuk AWS Glue Data Catalog Integrasi

- `glue:DeleteCatalog`

- `glue:GetCatalog`

Untuk Pendaftaran Sumber Daya Formasi Danau

- `lakeformation:DeregisterResource`

Batalkan Pendaftaran Redshift dari AWS Glue Data Catalog

CLI

Anda dapat menggunakan `modify-lakehouse-configuration` perintah untuk membatalkan pendaftaran cluster Anda dari AWS Glue Data Catalog, jika Anda memiliki penyedia iDC yang terkait dengan cluster Anda, itu akan menempatkan penyedia iDC di cluster ke mode dinonaktifkan.

```
aws redshift modify-lakehouse-configuration \
--cluster-identifier 'redshift-cluster' \
--lakehouse-registration Deregister
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke cluster yang disediakan yang ingin Anda hapus registrasi dan pilih.
3. Dari halaman detail cluster, pilih Deregister AWS Glue Data Catalog dari menu drop-down Ac tions dan pilih Deregister.

Hapus pendaftaran namespace Redshift Serverless dari AWS Glue Data Catalog

CLI

Anda dapat menggunakan `update-lakehouse-configuration` perintah untuk membatalkan pendaftaran namespace Redshift Serverless Anda dari AWS Glue Data Catalog, jika Anda memiliki penyedia iDC yang terkait dengan cluster Anda, itu akan menempatkan penyedia iDC di cluster ke mode dinonaktifkan.

```
aws redshift modify-lakehouse-configuration \
--cluster-identifier 'redshift-cluster' \
```

```
--lakehouse-registration Deregister
```

Console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke cluster namespace tanpa server yang ingin Anda hapus registrasi dan pilih.
3. Dari halaman detail cluster, pilih Deregister AWS Glue Data Catalog dari menu drop-down Ac tions dan pilih Deregister.

Spektrum Pergeseran Merah Amazon

Bagian ini menjelaskan cara menggunakan Redshift Spectrum untuk membaca data secara efisien dari Amazon S3.

Menggunakan Amazon Redshift Spectrum, Anda dapat menanyakan dan mengambil data terstruktur dan semi-terstruktur secara efisien dari file di Amazon S3 tanpa harus memuat data ke tabel Amazon Redshift. Kueri Redshift Spectrum menggunakan paralelisme besar-besaran untuk berjalan sangat cepat terhadap kumpulan data besar. Sebagian besar pemrosesan terjadi di lapisan Redshift Spectrum, dan sebagian besar data tetap ada di Amazon S3. Beberapa cluster dapat secara bersamaan menanyakan kumpulan data yang sama di Amazon S3 tanpa perlu membuat salinan data untuk setiap cluster.

Topik

- Ikhtisar Amazon Redshift Spectrum
- Memulai dengan Amazon Redshift Spectrum
- Kueri Tabel Amazon S3 dari Amazon Redshift
- Kebijakan IAM untuk Amazon Redshift Spectrum
- Spektrum Pergeseran Merah dan AWS Lake Formation
- File data untuk kueri di Amazon Redshift Spectrum
- Skema eksternal di Amazon Redshift Spectrum
- Tabel eksternal untuk Redshift Spectrum
- Menggunakan tabel Apache Iceberg dengan Amazon Redshift
- Performa kueri Amazon Redshift Spectrum
- Opsi penanganan data
- Contoh: Melakukan subquery berkorelasi di Redshift Spectrum
- Metrik di Amazon Redshift Spectrum
- Pemecahan masalah kueri di Amazon Redshift Spectrum
- Tutorial: Menanyakan data bersarang dengan Amazon Redshift Spectrum

Ikhtisar Amazon Redshift Spectrum

Topik ini menjelaskan detail penggunaan Redshift Spectrum untuk membaca secara efisien dari Amazon S3.

> **Note**
>
> Instans Amazon Redshift DC2 dan RA3 menggunakan Redshift Spectrum untuk menjalankan kueri data lake pada armada server khusus di luar cluster. Instans Amazon Redshift RG dan Amazon Redshift Serverless menyertakan mesin kueri data lake terintegrasi yang berjalan pada sumber daya komputasi cluster sendiri. Untuk informasi, lihat Mengku eri danau data Anda.

Amazon Redshift Spectrum berada di server khusus Amazon Redshift yang independen dari cluster Anda. Amazon Redshift mendorong banyak tugas komputasi intensif, seperti penyaringan predikat dan agregasi, hingga ke lapisan Redshift Spectrum. Dengan demikian, kueri Redshift Spectrum menggunakan kapasitas pemrosesan cluster Anda jauh lebih sedikit daripada kueri lainnya. Redshift Spectrum juga menskalakan dengan cerdas. Berdasarkan permintaan kueri Anda, Redshift Spectrum berpotensi menggunakan ribuan instans untuk memanfaatkan pemrosesan paralel besar-besaran.

Anda membuat tabel Redshift Spectrum dengan menentukan struktur untuk file Anda dan mendaftarkannya sebagai tabel dalam katalog data eksternal. Katalog data eksternal dapat berupa AWS Glue, katalog data yang disertakan dengan Amazon Athena, atau metastore Apache Hive Anda sendiri. Anda dapat membuat dan mengelola tabel eksternal baik dari Amazon Redshift menggunakan perintah bahasa definisi data (DDL) atau menggunakan alat lain yang terhubung ke katalog data eksternal. Perubahan pada katalog data eksternal segera tersedia untuk salah satu cluster Amazon Redshift Anda.

Secara opsional, Anda dapat mempartisi tabel eksternal pada satu atau lebih kolom. Mendefinisikan partisi sebagai bagian dari tabel eksternal dapat meningkatkan kinerja. Perbaikan terjadi karena pengoptimal kueri Amazon Redshift menghilangkan partisi yang tidak berisi data untuk kueri.

Pandangan yang terwujud pada tabel Spectrum dapat sangat meningkatkan biaya dan kinerja. Untuk informasi selengkapnya, lihat Tampilan terwujud pada tabel danau data eksternal di Amazon Redshift Spectrum.

Setelah tabel Redshift Spectrum Anda ditentukan, Anda dapat menanyakan dan bergabung dengan tabel seperti yang Anda lakukan pada tabel Amazon Redshift lainnya. Redshift Spectrum tidak

mendukung operasi pembaruan pada tabel eksternal. Anda dapat menambahkan tabel Redshift Spectrum ke beberapa cluster Amazon Redshift dan menanyakan data yang sama di Amazon S3 dari cluster mana pun di Wilayah yang sama AWS . Saat Anda memperbarui file data Amazon S3, data segera tersedia untuk kueri dari salah satu cluster Amazon Redshift Anda.

Katalog AWS Glue Data yang Anda akses mungkin dienkripsi untuk meningkatkan keamanan. Jika AWS Glue katalog dienkripsi, Anda memerlukan kunci AWS Key Management Service (AWS KMS) AWS Glue untuk mengakses AWS Glue katalog. AWS Glue enkripsi katalog tidak tersedia di semua Wil AWS ayah. Untuk daftar Wilayah yang didukung AWS , lihat Enkripsi dan Akses Aman untuk AWS Glue di Panduan AWS Glue Pengembang. Untuk informasi selengkapnya tentang enkripsi Katalog AWS Glue Data, lihat Mengenkripsi Katalog AWS Glue Data Anda di Panduan PengAWS Glue embang.

> **Note**
>
> Anda tidak dapat melihat detail tabel Redshift Spectrum menggunakan sumber daya yang sama yang Anda gunakan untuk tabel Amazon Redshift standar, seperti,, PG_CLASS PG_TABLE_DEFSTV_TBL_PERM, atau information_schema. Jika intelijen bisnis atau alat analitik Anda tidak mengenali tabel eksternal Redshift Spectrum, konfigurasikan aplikasi Anda untuk menanyakan SVV_EXTERNAL_TABEL danSVV_KOLON_EKSTERNAL_.

Wilayah Spektrum Pergeseran Merah Amazon

Redshift Spectrum tersedia di Wilayah AWS tempat Amazon Redshift tersedia, kecuali ditentukan lain dalam dokumentasi khusus Wilayah. Untuk Wilayah AWS ketersediaan di Wilayah komersial, lihat Titik akhir layanan untuk Redshift API di. Referensi Umum Amazon Web

Batasan Amazon Redshift Spectrum

Topik ini menjelaskan batasan untuk menggunakan Redshift Spectrum.

Perhatikan pertimbangan berikut saat Anda menggunakan Redshift Spectrum:

- Untuk cluster yang disediakan RA3 dan DC2 menggunakan Redshift Spectrum, cluster dan bucket Amazon S3 harus berada di Wilayah yang sama. AWS Cluster RG yang disediakan dan Amazon Redshift Serverless menyertakan mesin kueri data lake terintegrasi yang berjalan pada sumber daya komputasi cluster sendiri dan mendukung kueri data Amazon S3 di seluruh Wilayah. AWS

- Redshift Spectrum tidak mendukung routing VPC yang disempurnakan dengan klaster yang disediakan RA3 dan DC2. Untuk mengakses data Amazon S3 Anda, Anda mungkin perlu melakukan langkah-langkah konfigurasi tambahan. Untuk informasi selengkapnya, lihat Redshift Spectrum dan perutean VPC yang disempurnakan di Panduan Manajemen Amazon Redshift.
- Redshift Spectrum mendukung alias titik akses Amazon S3. Untuk informasi selengkapnya, lihat Menggunakan alias gaya bucket untuk titik akses Anda di Panduan Pengguna Layanan Penyimpanan S ederhana Amazon. Namun, Redshift Spectrum tidak mendukung VPC dengan alias titik akses Amazon S3. Untuk informasi selengkapnya, lihat Redshift Spectrum dan perutean VPC yang disempurnakan di Panduan Manajemen Amazon Redshift.
- Anda tidak dapat melakukan operasi pembaruan atau penghapusan pada tabel eksternal. Untuk membuat tabel eksternal baru dalam skema yang ditentukan, Anda dapat menggunakan CREATE EXTERNAL TABLE. Untuk informasi selengkapnya tentang MEMBUAT TABEL EKSTERNAL, lihatCREATE EXTERNAL TABLE. Untuk memasukkan hasil kueri SELECT ke tabel eksternal yang ada di katalog eksternal, Anda dapat menggunakan INSERT (tabel eksternal). Untuk informasi selengkapnya tentang INSERT (tabel eksternal), lihatINSERT (tabel eksternal).
- Kecuali Anda menggunakan AWS Glue Data Catalog yang diaktifkan untuk Formasi AWS Danau, Anda tidak dapat mengontrol izin pengguna pada tabel eksternal. Sebagai gantinya, Anda dapat memberikan dan mencabut izin pada skema eksternal. Untuk informasi lebih lanjut tentang bekerja dengan For AWS masi Danau, lihatSpektrum Pergeseran Merah dan AWS Lake Formation.
- Untuk menjalankan kueri Redshift Spectrum, pengguna database harus memiliki izin untuk membuat tabel sementara di database. Contoh berikut memberikan izin sementara pada database `spectrumdb` ke grup `spectrumusers` pengguna.

  ```
  grant temp on database spectrumdb to group spectrumusers;
  ```

  Untuk informasi selengkapnya, lihat HIBAH.
- Saat menggunakan Katalog Data Athena atau Katalog AWS Glue Data sebagai penyimpanan metadata, lihat Ku ota dan Batas di Panduan Manajemen Amazon Redshift.
- Redshift Spectrum tidak mendukung Amazon EMR dengan Kerberos.

Memulai dengan Amazon Redshift Spectrum

Dalam tutorial ini, Anda mempelajari cara menggunakan Amazon Redshift Spectrum untuk menanyakan data langsung dari file di Amazon S3. Jika Anda sudah memiliki cluster dan klien SQL, Anda dapat menyelesaikan tutorial ini dengan pengaturan minimal.

> **Note**
>
> Pada cluster yang disediakan RG, kueri data lake berjalan pada sumber daya komputasi cluster sendiri daripada pada server Redshift Spectrum khusus. Demikian pula, Amazon Redshift Serverless memproses kueri data lake menggunakan kapasitas komputasinya sendiri. Untuk informasi tentang penggunaan kueri danau data terintegrasi di RG dan Redshift Serverless, lihat Mengkueri danau data Anda.

> **Note**
>
> Kueri Redshift Spectrum dikenakan biaya tambahan. Biaya menjalankan kueri sampel dalam tutorial ini adalah nominal. Untuk informasi selengkapnya tentang harga, lihat harga https://aws.amazon.com/redshift/pricing/#redshift-spectrum-pricing Amazon Redshift Spectrum.

Prasyarat

Untuk menggunakan Redshift Spectrum, Anda memerlukan cluster Amazon Redshift dan klien SQL yang terhubung ke cluster Anda sehingga Anda dapat menjalankan perintah SQL. Cluster dan file data di Amazon S3 harus sama Wilayah AWS.

Untuk informasi tentang cara membuat cluster Amazon Redshift, lihat Mem ulai dengan gudang data yang disediakan Amazon Redshift di Panduan Memulai Amazon Redshift. Untuk informasi tentang cara menyambung ke cluster, lihat Mengh ubungkan ke gudang data Amazon Redshift di Panduan Memulai Amazon Redshift.

Dalam beberapa contoh berikut, data sampel berada di Wilayah AS Timur (Virginia Utara) (`us-east-1`), jadi Anda memerlukan cluster yang juga ada di`us-east-1`. Atau, Anda dapat menggunakan Amazon S3 untuk menyalin objek data dari bucket dan folder berikut ke bucket Anda di Wilayah AWS tempat cluster Anda berada:

- `s3://redshift-downloads/tickit/spectrum/customers/*`
- `s3://redshift-downloads/tickit/spectrum/sales_partition/*`
- `s3://redshift-downloads/tickit/spectrum/sales/*`
- `s3://redshift-downloads/tickit/spectrum/salesevent/*`

Jalankan perintah Amazon S3 yang serupa dengan berikut ini untuk menyalin data sampel yang terletak di AS Timur (Virginia Utara) ke Anda Wilayah AWS. Sebelum menjalankan perintah, buat bucket dan folder di bucket agar sesuai dengan perintah salin Amazon S3 Anda. Output dari perintah sal *bucket-name* in Amazon S3 mengonfirmasi bahwa file disalin ke yang Anda inginkan Wilayah AWS.

```
aws s3 cp s3://redshift-downloads/tickit/spectrum/ s3://bucket-name/tickit/spectrum/ --
copy-props none --recursive
```

Memulai dengan Redshift Spectrum menggunakan AWS CloudFormation

Sebagai alternatif dari langkah-langkah berikut, Anda dapat mengakses DataLake AWS CloudFormation template Redshift Spectrum untuk membuat tumpukan dengan bucket Amazon S3 yang dapat Anda kueri. Untuk informasi selengkapnya, lihat Luncurkan Anda AWS CloudFormation susun dan kemudian kueri data Anda di Amazon S3.

Memulai dengan Redshift Spectrum langkah demi langkah

Untuk mulai menggunakan Amazon Redshift Spectrum, ikuti langkah-langkah berikut:

- Langkah 1. Membuat peran IAM untuk Amazon Redshift
- Langkah 2: Mengaitkan peran IAM dengan cluster Anda
- Langkah 3: Buat skema eksternal dan tabel eksternal
- Langkah 4: Kueri data Anda di Amazon S3

Langkah 1. Membuat peran IAM untuk Amazon Redshift

Cluster Anda memerlukan otorisasi untuk mengakses Katalog Data eksternal Anda di AWS Glue atau Amazon Athena dan file data Anda di Amazon S3. Untuk memberikan otorisasi itu, Anda mereferensi AWS Identity and Access Management kan peran (IAM) yang dilampirkan ke cluster Anda. Untuk informasi selengkapnya tentang menggunakan peran dengan Amazon Redshift, lihat Mengotor isasi Operasi COPY dan UNLOAD Menggunakan Peran IAM.

> **Note**
>
> Dalam kasus tertentu, Anda dapat memigrasikan Katalog Data Athena Anda ke Katalog AWS Glue Data. Anda dapat melakukan ini jika cluster Anda berada di Wil AWS ayah yang AWS

Glue didukung dan Anda memiliki tabel eksternal Redshift Spectrum di Katalog Data Athena. Untuk menggunakan Katalog AWS Glue Data dengan Redshift Spectrum, Anda mungkin perlu mengubah kebijakan IAM Anda. Untuk informasi selengkapnya, lihat Meng upgrade ke Katalog AWS Glue Data di Panduan Pengguna Athena.

Saat Anda membuat peran untuk Amazon Redshift, pilih salah satu pendekatan berikut:

- Jika Anda menggunakan Redshift Spectrum dengan Katalog Data Athena atau AWS Glue Katalog Data, ikuti langkah-langkah yang diuraikan di. Cara membuat peran IAM untuk Amazon Redshift
- Jika Anda menggunakan Redshift Spectrum dengan AWS Glue Data Catalog yang diaktifkan untuk AWS Lake Formation, ikuti langkah-langkah yang diuraikan dalam prosedur ini:
  - Untuk membuat peran IAM untuk Amazon Redshift menggunakan AWS Glue Data Catalog diaktifkan untuk AWS Lake Formation
  - Untuk memberikan izin SELECT pada tabel untuk melakukan kueri di database Lake Formation

Cara membuat peran IAM untuk Amazon Redshift

1. Buka konsol IAM.
2. Di panel navigasi, pilih Peran.
3. Pilih Buat peran.
4. Pilih AWS layanan sebagai entitas tepercaya, lalu pilih Redshift sebagai kasus penggunaan.
5. Di bawah Use case for other Layanan AWS, pilih Redshift - Disesuaikan, lalu pilih Ber ikutnya.
6. Halaman kebijakan Tam bah izin muncul. Pilih `AmazonS3ReadOnlyAccess` dan`AWSGlueConsoleFullAccess`, jika Anda menggunakan Katalog AWS Glue Data. Atau pilih `AmazonAthenaFullAccess` apakah Anda menggunakan Katalog Data Athena. Pilih Berikutnya.

> **Note**
>
> `AmazonS3ReadOnlyAccess`Kebijakan ini memberikan akses read-only cluster ke semua bucket Amazon S3. Untuk memberikan akses hanya ke bucket data AWS sampel, buat kebijakan baru dan tambahkan izin berikut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::redshift-downloads/*"
        }
    ]
}
```

7. Untuk Nama Per an, masukkan nama untuk peran Anda, misalnya**myspectrum_role**.
8. Tinjau informasi, lalu pilih Buat peran.
9. Di panel navigasi, pilih Peran. Pilih nama peran baru Anda untuk melihat ringkasan, lalu salin Role ARN ke clipboard Anda. Nilai ini adalah Nama Sumber Daya Amazon (ARN) untuk peran yang baru saja Anda buat. Anda menggunakan nilai itu saat membuat tabel eksternal untuk mereferensikan file data Anda di Amazon S3.

Untuk membuat peran IAM untuk Amazon Redshift menggunakan AWS Glue Data Catalog diaktifkan untuk AWS Lake Formation

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Pilih Buat kebijakan.
4. Pilih untuk membuat kebijakan pada tab JSON.
5. Tempel di dokumen kebijakan JSON berikut, yang memberikan akses ke Katalog Data tetapi menolak izin administrator untuk Formasi Danau.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RedshiftPolicyForLF",
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "lakeformation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

6. Setelah selesai, pilih T injau untuk meninjau kebijakan. Validator kebijakan melaporkan kesalahan sintaksis.
7. Pada halaman Kebijakan Tinjauan, untuk Nama masukkan **`myspectrum_policy`** untuk memberi nama kebijakan yang Anda buat. Masukkan Deskripsi (opsional). Ulas Ringkasan kebijakan untuk melihat izin yang diberikan oleh kebijakan Anda. Kemudian pilih Buat kebijakan untuk menyimpan pekerjaan Anda.

   Setelah membuat kebijakan, Anda dapat memberikan akses ke pengguna Anda.

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Buat peran untuk penyedia identitas pihak ketiga (federasi) dalam Panduan Pengguna IAM.
- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Buat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Untuk memberikan izin SELECT pada tabel untuk melakukan kueri di database Lake Formation

1. Buka konsol Formasi Danau di https://console.aws.amazon.com/lakeformation/.
2. Di panel navigasi, pilih Izin data lake, lalu pilih Grant.
3. Ikuti petunjuk dalam Mem berikan izin tabel menggunakan metode sumber daya bernama di Panduan Peng AWS Lake Formation embang. Saat diminta, berikan informasi berikut:
   - Untuk peran IAM, pilih peran IAM yang Anda buat. `myspectrum_role` Saat Anda menjalankan Amazon Redshift Query Editor, ia menggunakan peran IAM ini untuk izin ke data.

   > **Note**
   >
   > Untuk memberikan izin SELECT pada tabel dalam Katalog Data yang diaktifkan Lake Formation untuk kueri, lakukan hal berikut:
   >
   > - Daftarkan jalur untuk data di Formasi Danau.
   > - Berikan izin pengguna ke jalur itu di Formasi Danau.
   > - Tabel yang dibuat dapat ditemukan di jalur yang terdaftar di Formasi Danau.

4. PilihIzin.

> **Important**
>
> Sebagai praktik terbaik, izinkan akses hanya ke objek Amazon S3 yang mendasarinya melalui izin Lake Formation. Untuk mencegah akses yang tidak disetujui, hapus izin yang diberikan kepada objek Amazon S3 di luar Danau Formation. Jika sebelumnya Anda mengakses objek Amazon S3 sebelum menyiapkan Lake Formation, hapus kebijakan IAM atau izin bucket yang sebelumnya telah disiapkan. Untuk informasi selengkapnya, lihat Mengupgrade AWS Glue Izin Data ke AWS Lake Formation Model dan Izin Formasi Danau.

Langkah 2: Mengaitkan peran IAM dengan cluster Anda

Sekarang Anda memiliki peran IAM yang memberi wewenang kepada Amazon Redshift untuk mengakses Katalog Data eksternal dan Amazon S3 untuk Anda. Pada titik ini, Anda harus mengaitkan peran itu dengan cluster Amazon Redshift Anda.

Untuk mengaitkan peran IAM dengan cluster

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih nama cluster yang ingin Anda perbarui.
3. Untuk T ind akan, pilih Kelola peran IAM. Halaman peran IAM muncul.
4. Pilih Enter ARN lalu masukkan peran ARN atau IAM, atau pilih peran IAM dari daftar. Kemudian pilih Tambahkan peran IAM untuk menambahkannya ke daftar peran IAM Terlampir.
5. Pilih Selesai untuk mengaitkan peran IAM dengan cluster. Cluster dimodifikasi untuk menyelesaikan perubahan.

Langkah 3: Buat skema eksternal dan tabel eksternal

Buat tabel eksternal dalam skema eksternal. Skema eksternal mereferensikan database dalam katalog data eksternal dan menyediakan peran IAM ARN yang mengotorisasi cluster Anda untuk mengakses Amazon S3 atas nama Anda. Anda dapat membuat database eksternal di Katalog Data Amazon Athena, AWS Glue Data Catalog, atau metastore Apache Hive, seperti Amazon EMR. Untuk contoh ini, Anda membuat database eksternal di Katalog Data Amazon Athena saat Anda membuat skema eksternal Amazon Redshift. Untuk informasi selengkapnya, lihat Skema eksternal di Amazon Redshift Spectrum.

Untuk membuat skema eksternal dan tabel eksternal

1. Untuk membuat skema eksternal, ganti peran IAM ARN dalam perintah berikut dengan peran ARN yang Anda buat pada langkah 1. Kemudian jalankan perintah di klien SQL Anda.

```
create external schema myspectrum_schema
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
create external database if not exists;
```

2. Untuk membuat tabel eksternal, jalankan perintah CREATE EXTERNAL TABLE berikut.

> **Note**
>
> Cluster Anda dan bucket Amazon S3 harus sama Wilayah AWS. Untuk contoh perintah CREATE EXTERNAL TABLE ini, bucket Amazon S3 dengan data sampel terletak di AS Timur (Virginia Utara) Wilayah AWS. Untuk melihat data sumber, unduh sales_ts.000 file.
>
> Anda dapat memodifikasi contoh ini untuk berjalan di yang berbeda Wilayah AWS. Buat bucket Amazon S3 sesuai keinginan Anda Wilayah AWS. Salin data penjualan dengan perintah salin Amazon S3. Kemudian perbarui opsi lokasi dalam `CREATE EXTERNAL TABLE` perintah contoh ke bucket Anda.
>
> ```
> aws s3 cp s3://redshift-downloads/tickit/spectrum/sales/ s3://bucket-name/
> tickit/spectrum/sales/ --copy-props none --recursive
> ```
>
> Output dari perintah sal *bucket-name* in Amazon S3 mengonfirmasi bahwa file telah disalin ke yang Anda inginkan Wilayah AWS.
>
> ```
> copy: s3://redshift-downloads/tickit/spectrum/sales/sales_ts.000 to
>  s3://bucket-name/tickit/spectrum/sales/sales_ts.000
> ```

```
create external table myspectrum_schema.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/'
table properties ('numRows'='172000');
```

Langkah 4: Kueri data Anda di Amazon S3

Setelah tabel eksternal dibuat, Anda dapat menanyakan tabel tersebut menggunakan pernyataan SELECT yang sama yang Anda gunakan untuk menanyakan tabel Amazon Redshift lainnya. Kueri pernyataan SELECT ini termasuk menggabungkan tabel, mengumpulkan data, dan memfilter predikat.

Untuk menanyakan data Anda di Amazon S3

1. Dapatkan jumlah baris dalam tabel MYSPECTRUM_SCHEMA.SALES .

```
select count(*) from myspectrum_schema.sales;
```

```
count
------
172462
```

2. Simpan tabel fakta yang lebih besar di Amazon S3 dan tabel dimensi yang lebih kecil di Amazon Redshift, sebagai praktik terbaik. Jika Anda memuat data sampel di Load data, Anda memiliki tabel bernama EVENT di database Anda. Jika tidak, buat tabel EVENT dengan menggunakan perintah berikut.

```
create table event(
eventid integer not null distkey,
venueid smallint not null,
catid smallint not null,
dateid smallint not null sortkey,
eventname varchar(200),
starttime timestamp);
```

3. Muat tabel EVENT dengan mengganti peran IAM ARN dalam perintah COPY berikut dengan peran ARN yang Anda buat. Langkah 1. Membuat peran IAM untuk Amazon Redshift Anda dapat secara opsional mengunduh dan melihat data sumber untuk `allevents_pipe.txt` dari bucket Amazon S3 di Wilayah AWS `us-east-1`.

```
copy event from 's3://redshift-downloads/tickit/allevents_pipe.txt'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
delimiter '|' timeformat 'YYYY-MM-DD HH:MI:SS' region 'us-east-1';
```

Contoh berikut menggabungkan tabel Amazon S3 eksternal MYSPECTRUM_ SCHEMA.SALES dengan EVENT tabel Amazon Redshift lokal untuk menemukan total penjualan untuk 10 acara teratas.

```
select top 10 myspectrum_schema.sales.eventid,
 sum(myspectrum_schema.sales.pricepaid) from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
eventid | sum
--------+---------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
   6471 | 47997.00
   2118 | 47863.00
    984 | 46780.00
   7851 | 46661.00
   5638 | 46280.00
```

4. Lihat rencana kueri untuk kueri sebelumnya. Perhatikan`S3 Seq Scan`,`S3 HashAggregate`, dan `S3 Query Scan` langkah-langkah yang dijalankan terhadap data di Amazon S3.

```
explain
select top 10 myspectrum_schema.sales.eventid,
 sum(myspectrum_schema.sales.pricepaid)
from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
QUERY PLAN
```

```
------------------------------------------------------------------------------
XN Limit  (cost=1001055770628.63..1001055770628.65 rows=10 width=31)


  ->  XN Merge  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


        Merge Key: sum(sales.derived_col2)


        ->  XN Network  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


              Send to leader


              ->  XN Sort  (cost=1001055770628.63..1001055770629.13 rows=200
 width=31)

                    Sort Key: sum(sales.derived_col2)


                    ->  XN HashAggregate  (cost=1055770620.49..1055770620.99
 rows=200 width=31)

                          ->  XN Hash Join DS_BCAST_INNER
 (cost=3119.97..1055769620.49 rows=200000 width=31)

                                Hash Cond: ("outer".derived_col1 = "inner".eventid)


                                ->  XN S3 Query Scan sales  (cost=3010.00..5010.50
 rows=200000 width=31)

                                      ->  S3 HashAggregate  (cost=3010.00..3010.50
 rows=200000 width=16)

                                            ->  S3 Seq Scan myspectrum_schema.sales
 location:"s3://redshift-downloads/tickit/spectrum/sales" format:TEXT
 (cost=0.00..2150.00 rows=172000 width=16)
                                                  Filter: (pricepaid > 30.00)
```

```
                                ->  XN Hash  (cost=87.98..87.98 rows=8798 width=4)

                                      ->  XN Seq Scan on event  (cost=0.00..87.98
rows=8798 width=4)
```

Luncurkan Anda AWS CloudFormation susun dan kemudian kueri data Anda di Amazon S3

Setelah membuat cluster Amazon Redshift dan terhubung ke cluster, Anda dapat menginstal DataLake AWS CloudFormation template Redshift Spectrum dan kemudian menanyakan data Anda.

CloudFormation menginstal DataLake template Redshift Spectrum Getting Started dan membuat tumpukan yang mencakup hal-hal berikut:

- Peran bernama `myspectrum_role` terkait dengan cluster Redshift Anda
- Skema eksternal bernama `myspectrum_schema`
- Tabel eksternal yang diberi `sales` nama dalam bucket Amazon S3
- Tabel Redshift bernama `event` dimuat dengan data

Untuk meluncurkan tumpukan Redshift Spectrum Getting Started DataLake CloudFormation

1. Pilih Luncurkan tumpukan CFN. Kon CloudFormation sol terbuka dengan DataLake.yml template yang dipilih.

   Anda juga dapat mengunduh dan menyesuaikan template Redshift Spectrum Getting Star DataLake CloudFormation ted CFN, lalu buka CloudFormation console (https://console.aws.amazon.com/cloudformation) dan buat tumpukan dengan template yang disesuaikan.
2. Pilih Berikutnya.
3. Di bawah Parameter, masukkan nama cluster Amazon Redshift, nama database, dan nama pengguna database Anda.
4. Pilih Berikutnya.

   Opsi tumpukan muncul.
5. Pilih Berikutnya untuk menerima pengaturan default.

6. Tinjau informasi dan di bawah Kemampuan, dan pilih Saya mengakui bahwa AWS CloudFormation mungkin membuat sumber daya IAM.
7. Pilih Buat tumpukan.

Jika terjadi kesalahan saat tumpukan sedang dibuat, lihat informasi berikut:

- Lihat tab CloudFormation Acara untuk informasi yang dapat membantu Anda mengatasi kesalahan.
- Hapus DataLake CloudFormation tumpukan sebelum mencoba operasi lagi.
- Pastikan Anda terhubung ke database Amazon Redshift Anda.
- Pastikan Anda memasukkan informasi yang benar untuk nama cluster Amazon Redshift, nama database, dan nama pengguna database.

Kueri data Anda di Amazon S3

Anda menanyakan tabel eksternal menggunakan pernyataan SELECT yang sama yang Anda gunakan untuk menanyakan tabel Amazon Redshift lainnya. Kueri pernyataan SELECT ini termasuk menggabungkan tabel, mengumpulkan data, dan memfilter predikat.

Kueri berikut mengembalikan jumlah baris dalam tabel `myspectrum_schema.sales` eksternal.

```
select count(*) from myspectrum_schema.sales;
```

```
count
------
172462
```

Bergabunglah dengan tabel eksternal dengan tabel lokal

Contoh berikut menggabungkan tabel eksternal `myspectrum_schema.sales` dengan tabel lokal `event` untuk menemukan total penjualan untuk 10 acara teratas.

```
select top 10 myspectrum_schema.sales.eventid, sum(myspectrum_schema.sales.pricepaid)
 from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
```

```
order by 2 desc;
```

```
eventid | sum
--------+---------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
   6471 | 47997.00
   2118 | 47863.00
    984 | 46780.00
   7851 | 46661.00
   5638 | 46280.00
```

Lihat paket kueri

Lihat rencana kueri untuk kueri sebelumnya. Perhatikan`S3 Seq Scan`,`S3 HashAggregate`, dan `S3 Query Scan` langkah-langkah yang dijalankan pada data di Amazon S3.

```
explain
select top 10 myspectrum_schema.sales.eventid, sum(myspectrum_schema.sales.pricepaid)
from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
QUERY PLAN


------------------------------------------------------------------------------
XN Limit  (cost=1001055770628.63..1001055770628.65 rows=10 width=31)


  ->  XN Merge  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


        Merge Key: sum(sales.derived_col2)
```

```
        ->  XN Network  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

              Send to leader

              ->  XN Sort  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

                    Sort Key: sum(sales.derived_col2)

                    ->  XN HashAggregate  (cost=1055770620.49..1055770620.99 rows=200
width=31)

                          ->  XN Hash Join DS_BCAST_INNER  (cost=3119.97..1055769620.49
rows=200000 width=31)

                                Hash Cond: ("outer".derived_col1 = "inner".eventid)

                                ->  XN S3 Query Scan sales  (cost=3010.00..5010.50
rows=200000 width=31)

                                      ->  S3 HashAggregate  (cost=3010.00..3010.50
rows=200000 width=16)

                                            ->  S3 Seq Scan spectrum.sales
location:"s3://redshift-downloads/tickit/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)
                                                  Filter: (pricepaid > 30.00)

                                ->  XN Hash  (cost=87.98..87.98 rows=8798 width=4)

                                      ->  XN Seq Scan on event  (cost=0.00..87.98
rows=8798 width=4)
```

Kueri Tabel Amazon S3 dari Amazon Redshift

Amazon Redshift terintegrasi dengan bucket tabel Amazon S3, sehingga Anda dapat mengakses sumber daya tabel S3 menggunakan Amazon Redshift. Baik Anda baru memulai atau mengelola

ribuan tabel di lingkungan Iceberg Anda, bucket tabel menyederhanakan pengelolaan danau data pada skala apa pun. Untuk informasi selengkapnya, lihat Bucket tabel.

Topik ini menjelaskan cara memulai dengan Amazon S3 Tables dan Redshift dan mengakses objek Tabel S3 menggunakan Amazon Redshift.

Prasyarat

Sebelum menanyakan Tabel S3 dari Amazon Redshift, Anda harus mengintegrasikan Tabel S3 dengan. AWS Glue Data Catalog Untuk petunjuk, lihat Meng integrasikan Tabel Amazon S3 dengan AWS Glue Data Catalog.

Setelah Tabel S3 terintegrasi dengan AWS Glue Data Catalog, kepala IAM dengan Tabel S3 yang diperlukan dan izin AWS Glue IAM dapat menemukan Tabel S3 melalui. AWS Glue Data Catalog

> **Note**
>
> Metode 3 (Auto-mounted awsdatacatalog) memiliki prasyarat tambahan. Lihat the section called “Metode 3: Auto-mounted awsdatacatalog” untuk detail.

Kueri Tabel S3 dari Amazon Redshift

Untuk memulai kueri Tabel S3, ikuti langkah-langkah berikut:

- Langkah 1: Buat peran IAM untuk Amazon Redshift
- Langkah 2: Lampirkan peran IAM ke cluster Amazon Redshift Anda
- Langkah 3: Kueri Tabel S3 dari Amazon Redshift

Langkah 1: Buat peran IAM untuk Amazon Redshift

Cluster Anda memerlukan otorisasi untuk mengakses katalog Tabel S3 eksternal di AWS Glue. Untuk memberikan otorisasi tersebut, Amazon Redshift menggunakan peran IAM yang dilampirkan ke cluster Anda. Buat peran IAM dengan izin kebijakan berikut.

> **Note**
>
> Dalam kebijakan dan contoh di bawah ini, ganti `us-west-2` dengan Wil AWS ayah Anda dan `111122223333` dengan ID AWS akun Anda.

Untuk membuat kebijakan:

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan.
3. Pilih Buat kebijakan.
4. Pilih tab JSON.
5. Tempel di dokumen kebijakan JSON berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GlueDataCatalogPermissions",
      "Effect": "Allow",
      "Action": [
        "glue:GetCatalog",
        "glue:GetDatabase",
        "glue:GetTable",
        "glue:GetTables",
        "glue:UpdateTable",
        "glue:DeleteTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:111122223333:catalog",
        "arn:aws:glue:us-west-2:111122223333:catalog/s3tablescatalog",
        "arn:aws:glue:us-west-2:111122223333:catalog/s3tablescatalog/*",
        "arn:aws:glue:us-west-2:111122223333:database/s3tablescatalog/*/*",
        "arn:aws:glue:us-west-2:111122223333:table/s3tablescatalog/*/*/*",
        "arn:aws:glue:us-west-2:111122223333:database/*",
        "arn:aws:glue:us-west-2:111122223333:table/*/*"
      ]
    },
    {
      "Sid": "S3TablesDataAccessPermissions",
      "Effect": "Allow",
```

```
      "Action": [
        "s3tables:GetTableBucket",
        "s3tables:GetNamespace",
        "s3tables:GetTable",
        "s3tables:GetTableMetadataLocation",
        "s3tables:GetTableData",
        "s3tables:ListTableBuckets",
        "s3tables:CreateTable",
        "s3tables:PutTableData",
        "s3tables:UpdateTableMetadataLocation",
        "s3tables:ListNamespaces",
        "s3tables:ListTables",
        "s3tables:DeleteTable"
      ],
      "Resource": [
        "arn:aws:s3tables:us-west-2:111122223333:bucket/*",
        "arn:aws:s3tables:us-west-2:111122223333:bucket/*/table/*"
      ]
    }
  ]
}
```

6. Pilih Tinjau kebijakan.
7. Pada halaman Kebijakan Tinjauan, `GlueCatalogS3Tables_Policy` masukkan Nama. Secara opsional, masukkan deskripsi. Tinjau ringkasan kebijakan, lalu pilih Buat kebijakan.

Untuk membuat peran IAM untuk Amazon Redshift:

1. Buka konsol IAM.
2. Di panel navigasi, pilih Peran.
3. Pilih Buat peran.
4. Pilih AWS layanan sebagai entitas tepercaya, lalu pilih Redshift sebagai kasus penggunaan.
5. Di bawah Kasus penggunaan untuk AWS layanan lain, pilih Redshift - Disesuaikan, lalu pilih Ber ikutnya.
6. Pada halaman Tambah izin, lampirkan `GlueCatalogS3Tables_Policy` kebijakan yang Anda buat di atas. Pilih Berikutnya.
7. Untuk Nama Per an, masukkan nama untuk peran Anda, misalnya`RedshiftS3TablesRole`.
8. Tinjau informasi, lalu pilih Buat peran.

9. Di panel navigasi, pilih Peran. Pilih nama peran baru Anda untuk melihat ringkasan, lalu salin Role ARN ke clipboard Anda. Anda akan menggunakan ARN ini saat membuat skema eksternal untuk ruang nama Tabel S3.

> **Note**
>
> Cakupan ke sumber daya tertentu dengan mengganti wildcard dengan Nama Sumber Daya Amazon (ARN) yang tepat.

Langkah 2: Lampirkan peran IAM ke cluster Amazon Redshift Anda

Kaitkan peran IAM yang Anda konfigurasikan di Langkah 1 dengan cluster Amazon Redshift Anda.

Menggunakan Konsol AWS Manajemen:

1. Masuk ke Konsol AWS Manajemen dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster yang ingin Anda perbarui.
3. Untuk Tind akan, pilih Kelola peran IAM untuk menampilkan daftar peran IAM saat ini yang terkait dengan cluster.
4. Pada halaman Kelola peran IAM, pilih peran IAM untuk ditambahkan, lalu pilih Tam bah peran IAM.
5. Pilih Selesai untuk menyimpan perubahan Anda.

Menggunakan AWS CLI:

Jalankan perintah berikut untuk mengaitkan peran IAM dengan cluster atau namespace yang ada. Ganti `my-redshift-cluster` atau `my-redshift-namespace` dengan pengidentifikasi cluster atau namespace Anda dan `111122223333` dengan ID AWS akun Anda.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier my-redshift-cluster \
    --add-iam-roles arn:aws:iam::111122223333:role/RedshiftS3TablesRole

-- for serverless
aws redshift-serverless update-namespace \
    --namespace-name my-redshift-namespace \
```

```
    --iam-roles "arn:aws:iam::111122223333:role/RedshiftS3TablesRole"
```

Untuk informasi selengkapnya, lihat Mengaitkan peran IAM dengan cluster di Panduan Manajemen Amazon Redshift.

Langkah 3: Kueri Tabel S3 dari Amazon Redshift

Saat Anda mengintegrasikan Tabel S3 dengan AWS Glue Data Catalog, layanan membuat struktur katalog federasi yang memetakan sumber daya Tabel S3 ke objek AWS Glue katalog:

- Bucket tabel S3 menjadi katalog di AWS Glue Data Catalog.
- Sebuah namespace S3 menjadi AWS Glue database.
- Tabel S3 menjadi objek AWS Glue tabel.

Integrasi menciptakan hierarki berikut:

- Katalog federasi: `s3tablescatalog` (dibuat secara otomatis)
- Katalog anak: Setiap ember tabel S3 menjadi katalog anak di bawahnya`s3tablescatalog`.
- Database: Setiap namespace S3 dalam bucket tabel menjadi database.
- Tabel: Setiap tabel S3 dalam namespace menjadi tabel.

Misalnya, jika Anda memiliki bucket tabel S3 bernama `analytics-bucket` dengan namespace `sales` yang berisi tabel`transactions`, jalur lengkap di dalamnya AWS Glue Data Catalog adalah:`s3tablescatalog/analytics-bucket/sales/transactions`.

Buat tautan sumber daya

Sebelum menggunakan salah satu dari tiga metode kueri di bawah ini, Anda harus membuat tautan sumber daya di AWS Glue Data Catalog. Tautan sumber daya memungkinkan Amazon Redshift mereferensikan database Tabel S3 melalui katalog standar.

Menggunakan AWS Glue konsol:

1. Buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi, pilih Basis data.
3. Pilih Buat, lalu pilih Taut an sumber daya.
4. Pada halaman Buat tautan sumber daya, berikan informasi berikut:

- Nama tautan sumber daya: Masukkan nama untuk tautan sumber daya (misalnya,`sales_resource_link`).
- Database bersama: Masukkan jalur database Tabel S3 (misalnya,`s3tablescatalog/analytics-bucket/sales`).
- Pemilik database bersama: Masukkan ID AWS akun Anda.
- ID katalog database bersama: Masukkan ID katalog dalam format`<account-id>:s3tablescatalog/<bucket-name>`.

5. Pilih Buat.

Menggunakan AWS CLI:

```
aws glue create-database \
  --region us-west-2 \
  --cli-input-json '{
        "CatalogId": "111122223333",
        "DatabaseInput": {
            "Name": "sales_resource_link",
            "TargetDatabase": {
                "CatalogId": "111122223333:s3tablescatalog/analytics-bucket",
                "DatabaseName": "sales"
            }
        }
  }'
```

Perintah ini membuat tautan sumber daya yang `sales_resource_link` dinamai di Katalog AWS Glue Data default Anda yang menunjuk ke `sales` database di bucket tabel S3`analytics-bucket`.

Setelah tautan sumber daya dibuat, Amazon Redshift menyediakan tiga metode untuk menanyakan Tabel S3. Pilih metode yang paling sesuai dengan kasus penggunaan Anda.

> **Note**
>
> Untuk membuat tautan sumber daya di tingkat database, Redshift Administrator harus memiliki `AWS Glue:CreateDatabase` izin pada katalog default dan database yang sedang dibuat.

Metode 1: BUAT SKEMA EKSTERNAL

Gunakan `CREATE EXTERNAL SCHEMA` untuk membuat skema eksternal yang mereferensikan database Tabel S3 Anda. Metode ini memberikan kontrol eksplisit atas penamaan skema dan konfigurasi.

Untuk detail sintaks lengkap, lihat MEMBUAT SKEMA EKSTERNAL di Panduan Pengembang Database Amazon Redshift.

Contoh

Gunakan nama database dan ID katalog dari Langkah 3. Ganti `111122223333` dengan ID AWS akun Anda.

```
CREATE EXTERNAL SCHEMA s3tables_schema
FROM DATA CATALOG DATABASE 'sales_resource_link'
IAM_ROLE 'arn:aws:iam::111122223333:role/RedshiftS3TablesRole'
REGION 'us-west-2'
CATALOG_ID '111122223333';

SELECT * FROM s3tables_schema.transactions;
```

Metode 2: BUAT DATABASE DARI ARN

Gunakan `CREATE DATABASE` dengan `FROM ARN` klausa untuk membuat database federasi yang secara langsung mereferensikan tautan AWS Glue sumber daya Anda. Metode ini secara otomatis memet AWS Glue akan database ke database Redshift.

Untuk detail sintaks lengkap, lihat MEMBUAT DATABASE di Panduan Pengembang Database Amazon Redshift.

Contoh

Ganti `111122223333` dengan ID AWS akun Anda.

```
CREATE DATABASE s3tables_db
FROM ARN 'arn:aws:glue:us-west-2:111122223333:database/sales_resource_link'
WITH DATA CATALOG SCHEMA analytics_schema
IAM_ROLE 'arn:aws:iam::111122223333:role/RedshiftS3TablesRole';

SELECT * FROM s3tables_db.analytics_schema.transactions;
```

Metode 3: Auto-mounted awsdatacatalog

Amazon Redshift dapat secara otomatis memasang AWS Glue Data Catalog database, termasuk tautan sumber daya Tabel S3, melalui `awsdatacatalog` database. Metode ini memerlukan akses federasi ke Spectrum (FAS) untuk diaktifkan pada cluster Anda.

Prasyarat

Untuk menggunakan `awsdatacatalog` database yang dipasang secara otomatis, Anda harus mengaktifkan akses federasi ke Spectrum. Hal ini memungkinkan Amazon Redshift menggunakan kredentif identitas federasi untuk mengakses AWS Glue Data Catalog dan sumber data eksternal.

Untuk mengaktifkan akses federasi ke Spectrum:

1. Hubungkan ke cluster Redshift Anda menggunakan identitas IAM dengan izin berikut:
   - `redshift:GetClusterCredentialsWithIAM`(untuk cluster yang disediakan) atau `redshift-serverless:GetCredentials` (untuk Tanpa Server)
   - `AmazonS3ReadOnlyAccess`
   - `AWSGlueConsoleFullAccess`
   - Izin Tabel S3 (seperti yang didefinisikan pada Langkah 1)
2. Saat Anda terhubung dengan identitas IAM, Amazon Redshift secara otomatis membuat pengguna database yang diawali dengan `IAM:` (untuk pengguna) atau `IAMR:` (untuk peran).
3. Sebagai administrator cluster, berikan izin pengguna federasi untuk mengakses skema eksternal. Ganti `my_user` dengan peran IAM atau nama pengguna Anda:

```
GRANT ALL ON SCHEMA awsdatacatalog TO "IAMR:my_user";
```

Untuk petunjuk terperinci tentang cara menyiapkan akses federasi, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift Spectrum di Panduan Manajemen Amazon Redshift.

Tabel Kueri S3

Setelah akses federasi dikonfigurasi, verifikasi skema yang dipasang dan kueri Tabel S3 Anda.

Verifikasi skema yang dipasang:

```
SHOW SCHEMAS FROM DATABASE awsdatacatalog;
```

Kueri Tabel S3 menggunakan nama tautan sumber daya dari Langkah 3:

```
SELECT * FROM awsdatacatalog.sales_resource_link.transactions;
```

Kebijakan IAM untuk Amazon Redshift Spectrum

Topik ini menjelaskan izin IAM yang diperlukan untuk menggunakan Redshift Spectrum.

Secara default, Amazon Redshift Spectrum menggunakan AWS Glue Data Catalog di Wil AWS ayah yang mendukung AWS Glue. Di Wil AWS ayah lain, Redshift Spectrum menggunakan Katalog Data Athena. Cluster Anda memerlukan otorisasi untuk mengakses katalog data eksternal Anda di AWS Glue atau Athena dan file data Anda di Amazon S3. Anda memberikan otorisasi tersebut dengan mereferensikan peran AWS Identity and Access Management (IAM) yang dilampirkan ke cluster Anda. Jika Anda menggunakan metastore Apache Hive untuk mengelola katalog data Anda, Anda tidak perlu memberikan akses ke Athena.

Anda dapat mengrantai peran sehingga cluster Anda dapat mengambil peran lain yang tidak melekat pada cluster. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Amazon Redshift Spectrum.

AWS Glue Katalog yang Anda akses mungkin dienkripsi untuk meningkatkan keamanan. Jika AWS Glue katalog dienkripsi, Anda memerlukan AWS KMS kunci AWS Glue untuk mengakses Katalog AWS Glue Data. Untuk informasi selengkapnya, lihat Mengenkripsi Katalog AWS Glue Data Anda di Panduan PengAWS Glue embang.

Topik

- Izin Amazon S3
- Cross-account Izin Amazon S3
- Kebijakan untuk memberikan atau membatasi akses menggunakan Redshift Spectrum
- Kebijakan untuk memberikan izin minimum
- Mengrantai peran IAM di Amazon Redshift Spectrum
- Mengontrol akses ke AWS Glue Katalog Data

Izin Amazon S3

Minimal, cluster Anda memerlukan akses GET dan LIST ke bucket Amazon S3 Anda. Jika bucket Anda tidak berada di AWS akun yang sama dengan cluster Anda, bucket Anda juga harus

mengotorisasi cluster Anda untuk mengakses data. Untuk informasi selengkapnya, lihat Men
gotorisasi Amazon Redshift untuk Mengakses AWS Layanan Lain atas Nama Anda.

> **Note**
>
> Bucket Amazon S3 tidak dapat menggunakan kebijakan bucket yang membatasi akses hanya dari titik akhir VPC tertentu.

Kebijakan berikut memberikan akses GET dan LIST ke bucket Amazon S3 apa pun. Kebijakan ini memungkinkan akses ke bucket Amazon S3 untuk operasi Redshift Spectrum serta COPY.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": ["s3:Get*", "s3:List*"],
  "Resource": "*"
 }]
}
```

Kebijakan berikut memberikan akses GET dan LIST ke bucket Amazon S3 Anda yang diberi nama`amzn-s3-demo-bucket`.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": ["s3:Get*", "s3:List*"],
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
 }]
}
```

Cross-account Izin Amazon S3

Untuk memberikan izin Redshift Spectrum untuk mengakses data dalam bucket Amazon S3 milik AWS akun lain, tambahkan kebijakan berikut ke bucket Amazon S3. Untuk informasi selengkapnya, lihat Mem berikan Iz Cross-Account in Bucket.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Example permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/spectrumrole"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::bucketname",
                "arn:aws:s3:::bucketname/*"
            ]
        }
    ]
}
```

Kebijakan untuk memberikan atau membatasi akses menggunakan Redshift Spectrum

Kebijakan contoh berikut memungkinkan akses ke bucket Amazon S3 untuk Amazon Redshift. Saat menambahkan kebijakan ini ke peran IAM, sebaiknya gunakan peran IAM yang hanya digunakan untuk Amazon Redshift. Kami juga menyarankan agar Anda tidak menggunakan kembali peran yang ada atau menambahkan izin tambahan ke peran IAM ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Kebijakan untuk memberikan izin minimum

Kebijakan berikut memberikan izin minimum yang diperlukan untuk menggunakan Redshift Spectrum dengan Amazon S3, AWS Glue, dan Athena.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucketname",
                "arn:aws:s3:::bucketname/folder1/folder2/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": "*"
        }
    ]
}
```

Jika Anda menggunakan Athena untuk katalog data AWS Glue, kebijakan memerlukan akses Athena penuh. Kebijakan berikut memberikan akses ke sumber daya Athena. Jika database eksternal Anda berada di metastore Hive, Anda tidak memerlukan akses Athena.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": ["athena:*"],
```

```
  "Resource": ["*"]
 }]
}
```

Mengrantai peran IAM di Amazon Redshift Spectrum

Saat Anda melampirkan peran ke cluster Anda, cluster Anda dapat mengambil peran itu untuk mengakses Amazon S3, Athena, dan AWS Glue atas nama Anda. Jika peran yang dilampirkan ke cluster Anda tidak memiliki akses ke sumber daya yang diperlukan, Anda dapat mengrantai peran lain, mungkin milik akun lain. Cluster Anda kemudian untuk sementara mengambil peran rantai untuk mengakses data. Anda juga dapat memberikan akses lintas akun dengan mengrantai peran. Anda dapat merangkai maksimal 10 peran. Setiap peran dalam rantai mengambil peran berikutnya dalam rantai, sampai cluster mengambil peran di akhir rantai.

Untuk rantai peran, Anda membangun hubungan kepercayaan antara peran. Peran yang mengambil peran lain harus memiliki kebijakan izin yang memungkinkannya untuk mengambil peran yang ditentukan. Pada gilirannya, peran yang melewati izin harus memiliki kebijakan kepercayaan yang memungkinkannya meneruskan izinnya ke peran lain. Untuk informasi selengkapnya, lihat Mengh [ubungkan Peran IAM di Amazon Redshift.]

Saat menjalankan perintah CREATE EXTERNAL SCHEMA, Anda dapat mengrantai peran dengan menyertakan daftar ARN peran yang dipisahkan koma.

> **Note**
>
> Daftar peran yang dirantai tidak boleh menyertakan spasi.

Dalam contoh berikut, `MyRedshiftRole` dilampirkan ke cluster. `MyRedshiftRole`mengambil peran`AcmeData`, yang menjadi milik akun`111122223333`.

```
create external schema acme from data catalog
database 'acmedb' region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole,arn:aws:iam::111122223333:role/
AcmeData';
```

Mengontrol akses ke AWS Glue Katalog Data

Jika digunakan AWS Glue untuk katalog data, Anda dapat menerapkan kontrol akses halus ke Katalog AWS Glue Data dengan kebijakan IAM Anda. Misalnya, Anda mungkin ingin mengekspos hanya beberapa database dan tabel ke peran IAM tertentu.

Bagian berikut menjelaskan kebijakan IAM untuk berbagai tingkat akses ke data yang disimpan dalam Katalog AWS Glue Data.

Topik

- Kebijakan untuk operasi database
- Kebijakan untuk operasi tabel
- Kebijakan untuk operasi partisi

Kebijakan untuk operasi database

Jika Anda ingin memberi pengguna izin untuk melihat dan membuat database, mereka memerlukan hak akses ke database dan Katalog AWS Glue Data.

Contoh kueri berikut membuat database.

```
CREATE EXTERNAL SCHEMA example_db
FROM DATA CATALOG DATABASE 'example_db' region 'us-west-2'
IAM_ROLE 'arn:aws:iam::redshift-account:role/spectrumrole'
CREATE EXTERNAL DATABASE IF NOT EXISTS
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk membuat database.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "glue:GetDatabase",
                "glue:CreateDatabase"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:catalog"
            ]
        }
    ]
}
```

Contoh kueri berikut mencantumkan database saat ini.

```
SELECT * FROM SVV_EXTERNAL_DATABASES WHERE
databasename = 'example_db1' or databasename = 'example_db2';
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk mencantumkan database saat ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabases"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:database/example_db1",
                "arn:aws:glue:us-west-2:111122223333:database/example_db2",
                "arn:aws:glue:us-west-2:111122223333:catalog"
            ]
        }
    ]
}
```

Kebijakan untuk operasi tabel

Jika Anda ingin memberi pengguna izin untuk melihat, membuat, melepas, mengubah, atau mengambil tindakan lain pada tabel, mereka memerlukan beberapa jenis akses. Mereka membutuhkan akses ke tabel itu sendiri, database milik mereka, dan katalog.

Contoh kueri berikut membuat tabel eksternal.

```
CREATE EXTERNAL TABLE example_db.example_tbl0(
    col0 INT,
    col1 VARCHAR(255)
) PARTITIONED BY (part INT) STORED AS TEXTFILE
LOCATION 's3://test/s3/location/';
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk membuat tabel eksternal.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateTable"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Contoh berikut kueri setiap daftar tabel eksternal saat ini.

```
SELECT * FROM svv_external_tables
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

```
SELECT * FROM svv_external_columns
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

```
SELECT parameters FROM svv_external_tables
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk mencantumkan tabel eksternal saat ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl1"
            ]
        }
    ]
}
```

Contoh kueri berikut mengubah tabel yang ada.

```
ALTER TABLE example_db.example_tbl0
SET TABLE PROPERTIES ('numRows' = '100');
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk mengubah tabel yang ada.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTable",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Contoh kueri berikut menjatuhkan tabel yang ada.

```
DROP TABLE example_db.example_tbl0;
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk menghapus tabel yang ada.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:DeleteTable"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Kebijakan untuk operasi partisi

Jika Anda ingin memberi pengguna izin untuk melakukan operasi tingkat partisi (lihat, buat, lepas, ubah, dan sebagainya), mereka memerlukan izin ke tabel milik partisi tersebut. Mereka juga memerlukan izin ke database terkait dan Katalog AWS Glue Data.

Contoh kueri berikut membuat partisi.

```
ALTER TABLE example_db.example_tbl0
ADD PARTITION (part=0) LOCATION 's3://test/s3/location/part=0/';
ALTER TABLE example_db.example_t
ADD PARTITION (part=1) LOCATION 's3://test/s3/location/part=1/';
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk membuat partisi.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTable",
                "glue:BatchCreatePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Contoh kueri berikut mencantumkan partisi saat ini.

```
SELECT * FROM svv_external_partitions
WHERE schemname = 'example_db' AND
tablename = 'example_tbl0'
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk mencantumkan partisi saat ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetPartitions",
                "glue:GetTables",
                "glue:GetTable"
```

```
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Contoh kueri berikut mengubah partisi yang ada.

```
ALTER TABLE example_db.example_tbl0 PARTITION(part='0')
SET LOCATION 's3://test/s3/new/location/part=0/';
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk mengubah partisi yang ada.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetPartition",
                "glue:UpdatePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Contoh kueri berikut menjatuhkan partisi yang ada.

```
ALTER TABLE example_db.example_tbl0 DROP PARTITION(part='0');
```

Kebijakan IAM berikut memberikan izin minimum yang diperlukan untuk menghapus partisi yang ada.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:DeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:111122223333:catalog",
                "arn:aws:glue:us-west-2:111122223333:database/example_db",
                "arn:aws:glue:us-west-2:111122223333:table/example_db/
example_tbl0"
            ]
        }
    ]
}
```

Spektrum Pergeseran Merah dan AWS Lake Formation

Topik ini menjelaskan cara menggunakan Redshift Spectrum dengan Formasi Danau. Lake Formation adalah layanan untuk berbagi data analitik.

Anda dapat menggunakannya AWS Lake Formation untuk menentukan dan menerapkan kebijakan akses tingkat database, tabel, dan kolom secara terpusat ke data yang disimpan di Amazon S3. Setelah data Anda terdaftar dengan AWS Glue Data Catalog diaktifkan dengan Lake Formation, Anda dapat menanyainya dengan menggunakan beberapa layanan, termasuk Redshift Spectrum.

Lake Formation menyediakan keamanan dan tata kelola Katalog Data. Dalam Lake Formation, Anda dapat memberikan dan mencabut izin ke objek Katalog Data, seperti database, tabel, kolom, dan penyimpanan Amazon S3 yang mendasarinya.

> **Important**
>
> Anda hanya dapat menggunakan Redshift Spectrum dengan Katalog Data yang diaktifkan Formasi Danau di Wil AWS ayah di mana Formasi Danau tersedia. Untuk daftar Wilayah yang tersedia, lihat AWS Lake Formation titik akhir dan kuota di Referensi Umum AWS.

Dengan menggunakan Redshift Spectrum dengan Lake Formation, Anda dapat melakukan hal berikut:

- Gunakan Lake Formation sebagai tempat terpusat tempat Anda memberikan dan mencabut izin dan kebijakan kontrol akses pada semua data Anda di danau data. Lake Formation menyediakan hierarki izin untuk mengontrol akses ke database dan tabel dalam Katalog Data. Untuk informasi selengkapnya, lihat Ikhtisar izin Formasi Danau di Panduan AWS Lake Formation Pengembang.
- Buat tabel eksternal dan jalankan kueri pada data di danau data. Sebelum pengguna di akun Anda dapat menjalankan kueri, administrator akun data lake mendaftarkan jalur Amazon S3 Anda yang ada yang berisi data sumber dengan Lake Formation. Administrator juga membuat tabel dan memberikan izin kepada pengguna Anda. Akses dapat diberikan pada database, tabel, atau kolom. Administrator dapat menggunakan filter data di Lake Formation untuk memberikan kontrol akses granular atas data sensitif Anda yang disimpan di Amazon S3. Untuk informasi selengkapnya, lihat Menggunakan filter data untuk keamanan tingkat baris dan tingkat sel.

  Setelah data terdaftar di Katalog Data, setiap kali pengguna mencoba menjalankan kueri, Lake Formation memverifikasi akses ke tabel untuk prinsipal tertentu. Lake Formation menjual kredenSIAL sementara ke Redshift Spectrum, dan kueri berjalan.
- Jalankan kueri Redshift Spectrum terhadap akun otomatis AWS Glue Data Catalog menggunakan kredentif IAM yang diperoleh dengan `GetCredentials` atau`GetClusterCredentials`, dan kelola izin Lake Formation oleh pengguna database (atau). IAMR:username IAM:username

Saat Anda menggunakan Redshift Spectrum dengan Katalog Data diaktifkan untuk Formasi Danau, salah satu dari berikut ini harus ada:

- Peran IAM yang terkait dengan cluster yang memiliki izin ke Katalog Data.

- Identitas IAM federasi yang dikonfigurasi untuk mengelola akses ke sumber daya eksternal. Untuk informasi selengkapnya, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift.

> **Important**
>
> Anda tidak dapat mengrantai peran IAM saat menggunakan Redshift Spectrum dengan Katalog Data diaktifkan untuk Formasi Danau.

Untuk mempelajari lebih lanjut tentang langkah-langkah yang diperlukan AWS Lake Formation untuk mengatur penggunaan dengan Redshift Spectrum, lihat Tutorial: Membuat danau data dari sumber JDBC di Lake Formation di Panduan Peng AWS Lake Formation embang. Secara khusus, lihat Kueri data di danau data menggunakan Amazon Redshift Spectrum untuk detail tentang integrasi dengan Redshift Spectrum. Data dan AWS sumber daya yang digunakan dalam topik ini tergantung pada langkah-langkah sebelumnya dalam tutorial.

Menggunakan filter data untuk keamanan tingkat baris dan tingkat sel

Anda dapat menentukan filter data AWS Lake Formation untuk mengontrol tingkat baris dan akses tingkat sel kueri Redshift Spectrum ke data yang ditentukan dalam Katalog Data Anda. Untuk mengatur ini, Anda melakukan tugas-tugas berikut:

- Buat filter data di Lake Formation dengan informasi berikut:
  - Spesifikasi kolom dengan daftar kolom untuk disertakan atau dikecualikan dari hasil kueri.
  - Ekspresi filter baris yang menentukan baris yang akan disertakan dalam hasil kueri.

  Untuk informasi selengkapnya tentang cara membuat filter data, lihat Filter data di Lake Formation di Panduan Peng AWS Lake Formation embang.
- Buat tabel eksternal di Amazon Redshift yang mereferensikan tabel di Katalog Data yang diaktifkan Formasi Danau Anda. Untuk detail tentang cara menanyakan tabel Formasi Danau menggunakan Redshift Spectrum, lihat Ku eri data di danau data menggunakan Amazon Redshift Spectrum di Panduan Peng AWS Lake Formation embang.

Setelah tabel didefinisikan di Amazon Redshift, Anda dapat menanyakan tabel Lake Formation dan hanya mengakses baris dan kolom yang diizinkan oleh filter data.

Untuk panduan terperinci tentang cara mengatur keamanan tingkat baris dan tingkat sel di Lake Formation, lalu kueri menggunakan Redshift Spectrum, lihat Menggunakan Amazon Redshift Spectrum dengan kebijakan keamanan tingkat baris dan tingkat sel yang ditentukan di. AWS Lake Formation

File data untuk kueri di Amazon Redshift Spectrum

Bagian ini menjelaskan cara membuat file data di Amazon S3 dalam format yang didukung Redshift Spectrum.

File data yang Anda gunakan untuk kueri di Amazon Redshift Spectrum biasanya merupakan jenis file yang sama dengan yang Anda gunakan untuk aplikasi lain. Misalnya, jenis file yang sama digunakan dengan Amazon Athena, Amazon EMR, dan Quick. Anda dapat menanyakan data dalam format aslinya langsung dari Amazon S3. Untuk melakukan ini, file data harus dalam format yang didukung Redshift Spectrum dan ditempatkan di bucket Amazon S3 yang dapat diakses oleh cluster Anda.

Bucket Amazon S3 dengan file data dan cluster Amazon Redshift harus berada di Wilayah yang sama AWS . Untuk informasi tentang Wil AWS ayah yang didukung, lihatWilayah Spektrum Pergeseran Merah Amazon.

Format data untuk Redshift Spectrum

Redshift Spectrum mendukung format data terstruktur dan semi-terstruktur berikut.

Format file	Berkolom	Mendukung pembacaan paralel	Unit terpisah
Parquet	Ya	Ya	Grup baris
ORC	Ya	Ya	Garis
RCFile	Ya	Ya	Grup baris
TextFile	Tidak	Ya	Baris
SequenceFile	Tidak	Ya	Baris atau blok
RegexSerde	Tidak	Ya	Baris

Format file	Berkolom	Mendukung pembacaan paralel	Unit terpisah
OpenCSV	Tidak	Ya	Baris
AVRO	Tidak	Ya	Blokir
Ion	Tidak	Tidak	N/A
JSON	Tidak	Tidak	N/A

Pada tabel sebelumnya, judul menunjukkan yang berikut:

- Columnar — Apakah format file secara fisik menyimpan data dalam struktur berorientasi kolom dibandingkan dengan yang berorientasi baris.
- Mendukung pembacaan paralel - Apakah format file mendukung membaca blok individual dalam file. Membaca blok individual memungkinkan pemrosesan file yang terdistribusi di beberapa permintaan Redshift Spectrum independen alih-alih harus membaca file lengkap dalam satu permintaan.
- Unit terpisah — Untuk format file yang dapat dibaca secara paralel, unit split adalah potongan data terkecil yang dapat diproses oleh permintaan Spektrum Redshift tunggal.

> **Note**
>
> Nilai stempel waktu dalam file teks harus dalam format`yyyy-MM-dd HH:mm:ss.SSSSSS`, seperti yang ditunjukkan nilai stempel waktu berikut:. `2017-05-01 11:30:59.000000`

Sebaiknya gunakan format file penyimpanan kolumnar, seperti Apache Parquet. Amazon Redshift mendukung format file data Apache Parquet v1. Dengan format file penyimpanan kolumnar, Anda dapat meminimalkan transfer data dari Amazon S3 dengan memilih hanya kolom yang Anda butuhkan.

Jenis kompresi untuk Redshift Spectrum

Untuk mengurangi ruang penyimpanan, meningkatkan kinerja, dan meminimalkan biaya, kami sangat menyarankan Anda mengompres file data Anda. Redshift Spectrum mengenali jenis kompresi file berdasarkan ekstensi file.

Redshift Spectrum mendukung jenis dan ekstensi kompresi berikut.

Algoritma Kompresi	Ekstensi File	Mendukung Pembacaan Paralel
Gzip	.gz	Tidak
Bzip 2	.bz2	Ya
Snappy	.lincah	Tidak

Amazon Redshift juga mendukung kompresi Zstandard (zstd) untuk file Apache Parquet dan tabel Apache Iceberg.

Anda dapat menerapkan kompresi pada tingkat yang berbeda. Paling umum, Anda mengompres seluruh file atau mengompres blok individual dalam file. Mengompresi format kolom pada tingkat file tidak menghasilkan manfaat kinerja.

Agar Redshift Spectrum dapat membaca file secara paralel, berikut ini harus benar:

- Format file mendukung pembacaan paralel.
- Kompresi tingkat file, jika ada, mendukung pembacaan paralel.

Tidak masalah apakah unit terpisah individu dalam file dikompresi menggunakan algoritma kompresi yang dapat dibaca secara paralel, karena setiap unit split diproses oleh satu permintaan Redshift Spectrum. Contohnya adalah file Snappy-compressed parket. Grup baris individu dalam file Parquet dikompresi menggunakan Snappy, tetapi struktur tingkat atas file tetap tidak terkompresi. Dalam hal ini, file dapat dibaca secara paralel karena setiap permintaan Redshift Spectrum dapat membaca dan memproses grup baris individu dari Amazon S3.

Enkripsi untuk Redshift Spectrum

Redshift Spectrum secara transparan mendekripsi file data yang dienkripsi menggunakan opsi enkripsi berikut:

- Server-side enkripsi (SSE-S3) menggunakan kunci AES-256 enkripsi yang dikelola oleh Amazon S3.
- Server-side enkripsi dengan kunci yang dikelola oleh AWS Key Management Service (SSE-KMS).

Redshift Spectrum tidak mendukung enkripsi sisi klien Amazon S3. Untuk informasi selengkapnya tentang enkripsi sisi server, lihat Mel indungi Data Menggunakan Server-Side Enkripsi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Amazon Redshift menggunakan pemrosesan paralel massif (MPP) untuk mencapai eksekusi cepat dari kueri kompleks yang beroperasi pada sejumlah besar data. Redshift Spectrum memperluas prinsip yang sama untuk menanyakan data eksternal, menggunakan beberapa instance Redshift Spectrum sesuai kebutuhan untuk memindai file. Tempatkan file dalam folder terpisah untuk setiap tabel.

Anda dapat mengoptimalkan data Anda untuk pemrosesan paralel dengan melakukan hal berikut:

- Jika format file atau kompresi Anda tidak mendukung pembacaan secara paralel, pecahkan file besar menjadi banyak file yang lebih kecil. Sebaiknya gunakan ukuran file antara 64 MB dan 1 GB.
- Simpan semua file dengan ukuran yang sama. Jika beberapa file jauh lebih besar dari yang lain, Redshift Spectrum tidak dapat mendistribusikan beban kerja secara merata.

Skema eksternal di Amazon Redshift Spectrum

Topik ini menjelaskan cara membuat dan menggunakan skema eksternal dengan Redshift Spectrum. Skema eksternal adalah kumpulan tabel yang Anda gunakan sebagai referensi untuk mengakses data di luar cluster Amazon Redshift Anda. Tabel ini berisi metadata tentang data eksternal yang dibaca Redshift Spectrum.

Semua tabel eksternal harus dibuat dalam skema eksternal, yang Anda buat menggunakan BUAT SKEMA EKSTERNAL pernyataan.

> **Note**
>
> Beberapa aplikasi menggunakan istilah database dan skema secara ber gantian. Amazon Redshift menggunakan istilah “skema”.

Skema eksternal Amazon Redshift mereferensikan database eksternal dalam katalog data eksternal. Anda dapat membuat database eksternal di Amazon Redshift, di Amazon Athena, di AWS Glue Data Catalog, atau di metastore Apache Hive, seperti Amazon EMR. Jika Anda membuat database eksternal di Amazon Redshift, database berada di Katalog Data Athena. Untuk membuat database dalam metastore Hive, Anda perlu membuat database di aplikasi Hive Anda.

Amazon Redshift memerlukan otorisasi untuk mengakses Katalog Data di Athena dan file data di Amazon S3 atas nama Anda. Untuk memberikan otorisasi itu, pertama-tama Anda membuat peran AWS Identity and Access Management (IAM). Kemudian Anda melampirkan peran ke cluster Anda dan memberikan Amazon Resource Name (ARN) untuk peran tersebut dalam `CREATE EXTERNAL SCHEMA` pernyataan Amazon Redshift. Untuk informasi selengkapnya tentang otorisasi, lihatKebijakan IAM untuk Amazon Redshift Spectrum.

Untuk membuat database eksternal pada saat yang sama Anda membuat skema eksternal, tentukan `FROM DATA CATALOG` dan sertakan `CREATE EXTERNAL DATABASE` klausa dalam `CREATE EXTERNAL SCHEMA` pernyataan Anda.

Contoh berikut membuat skema eksternal bernama `spectrum_schema` menggunakan database eksternal`spectrum_db`.

```
create external schema spectrum_schema from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
create external database if not exists;
```

Jika Anda mengelola katalog data menggunakan Athena, tentukan nama database Athena dan Wil AWS ayah tempat Katalog Data Athena berada.

Contoh berikut membuat skema eksternal menggunakan `sampledb` database default di Katalog Data Athena.

```
create external schema athena_schema from data catalog
```

```
database 'sampledb'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
region 'us-east-2';
```

> **Note**
>
> `region`Parameter meref AWS erensikan Wilayah tempat Katalog Data Athena berada, bukan lokasi file data di Amazon S3.

Jika Anda mengelola katalog data menggunakan metastore Hive, seperti Amazon EMR, grup keamanan Anda harus dikonfigurasi untuk mengizinkan lalu lintas antar cluster.

Dalam pernyataan CREATE EXTERNAL SCHEMA, tentukan `FROM HIVE METASTORE` dan sertakan URI metastore dan nomor port. Contoh berikut membuat skema eksternal menggunakan database metastore Hive bernama`hive_db`.

```
create external schema hive_schema
from hive metastore
database 'hive_db'
uri '172.10.10.10' port 99
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
```

Untuk melihat skema eksternal untuk cluster Anda, kueri tabel katalog PG_EXTERNAL_SCHEMA atau tampilan SVV_EXTERNAL_SCHEMAS. Contoh berikut menanyakan SVV_EXTERNAL_SCHEMAS, yang bergabung dengan PG_EXTERNAL_SCHEMA dan PG_NAMESPACE.

```
select * from svv_external_schemas
```

Untuk sintaks perintah lengkap dan contoh, lihatBUAT SKEMA EKSTERNAL.

Bekerja dengan katalog eksternal di Amazon Redshift Spectrum

Metadata untuk database eksternal Amazon Redshift Spectrum dan tabel eksternal disimpan dalam katalog data eksternal. Secara default, metadata Redshift Spectrum disimpan dalam Katalog Data Athena. Anda dapat melihat dan mengelola database dan tabel Redshift Spectrum di konsol Athena Anda.

Anda juga dapat membuat dan mengelola database eksternal dan tabel eksternal menggunakan bahasa definisi data Hive (DDL) menggunakan Athena atau metastore Hive, seperti Amazon EMR.

> **Note**
>
> Sebaiknya gunakan Amazon Redshift untuk membuat dan mengelola database eksternal dan tabel eksternal di Redshift Spectrum.

Melihat database Redshift Spectrum di Athena dan AWS Glue

Anda dapat membuat database eksternal dengan menyertakan klausa CREATE EXTERNAL DATABASE IF NOT EXISTS sebagai bagian dari pernyataan CREATE EXTERNAL SCHEMA Anda. Dalam kasus seperti itu, metadata database eksternal disimpan dalam Katalog Data Anda. Metadata untuk tabel eksternal yang Anda buat memenuhi syarat oleh skema eksternal juga disimpan dalam Katalog Data Anda.

Athena dan AWS Glue memelihara Katalog Data untuk setiap yang didukung Wilayah AWS. Untuk melihat metadata tabel, masuk ke Athena atau AWS Glue konsol. Di Athena, pilih Sumber data, Anda AWS Glue, lalu lihat detail database Anda. Di AWS Glue, pilih Database, database eksternal Anda, lalu lihat detail database Anda.

Jika Anda membuat dan mengelola tabel eksternal menggunakan Athena, daftarkan database menggunakan CREATE EXTERNAL SCHEMA. Misalnya, perintah berikut mendaftarkan database Athena bernama`sampledb`.

```
create external schema athena_sample
from data catalog
database 'sampledb'
iam_role 'arn:aws:iam::123456789012:role/mySpectrumRole'
region 'us-east-1';
```

Saat Anda menanyakan tampilan sistem SVV_EXTERNAL_TABLES, Anda melihat tabel di `sampledb` database Athena dan juga tabel yang Anda buat di Amazon Redshift.

```
select * from svv_external_tables;
```

```
schemaname    | tablename        | location
```

```
--------------+-----------------
+-----------------------------------------------------
athena_sample | elb_logs         | s3://athena-examples/elb/plaintext
athena_sample | lineitem_1t_csv  | s3://myspectrum/tpch/1000/lineitem_csv

athena_sample | lineitem_1t_part | s3://myspectrum/tpch/1000/lineitem_partition

spectrum      | sales            | s3://redshift-downloads/tickit/spectrum/sales

spectrum      | sales_part       | s3://redshift-downloads/tickit/spectrum/sales_part
```

Mendaftarkan database metastore Apache Hive

Jika Anda membuat tabel eksternal di metastore Apache Hive, Anda dapat menggunakan CREATE EXTERNAL SCHEMA untuk mendaftarkan tabel tersebut di Redshift Spectrum.

Dalam pernyataan CREATE EXTERNAL SCHEMA, tentukan klausa FROM HIVE METASTORE dan berikan URI Hive metastore dan nomor port. Peran IAM harus menyertakan izin untuk mengakses Amazon S3 tetapi tidak memerlukan izin Athena apa pun. Contoh berikut mendaftarkan metastor Hive.

```
create external schema if not exists hive_schema
from hive metastore
database 'hive_database'
uri 'ip-10-0-111-111.us-west-2.compute.internal' port 9083
iam_role 'arn:aws:iam::123456789012:role/mySpectrumRole';
```

Mengaktifkan cluster Amazon Redshift Anda untuk mengakses cluster Amazon EMR Anda

Jika metastore Hive Anda ada di Amazon EMR, Anda harus memberikan akses cluster Amazon Redshift ke cluster Amazon EMR Anda. Untuk melakukannya, Anda membuat grup keamanan Amazon EC2. Anda kemudian mengizinkan semua lalu lintas masuk ke grup keamanan EC2 dari grup keamanan cluster Amazon Redshift Anda dan grup keamanan cluster Amazon EMR Anda. Kemudian Anda menambahkan keamanan EC2 ke cluster Amazon Redshift dan cluster Amazon EMR Anda.

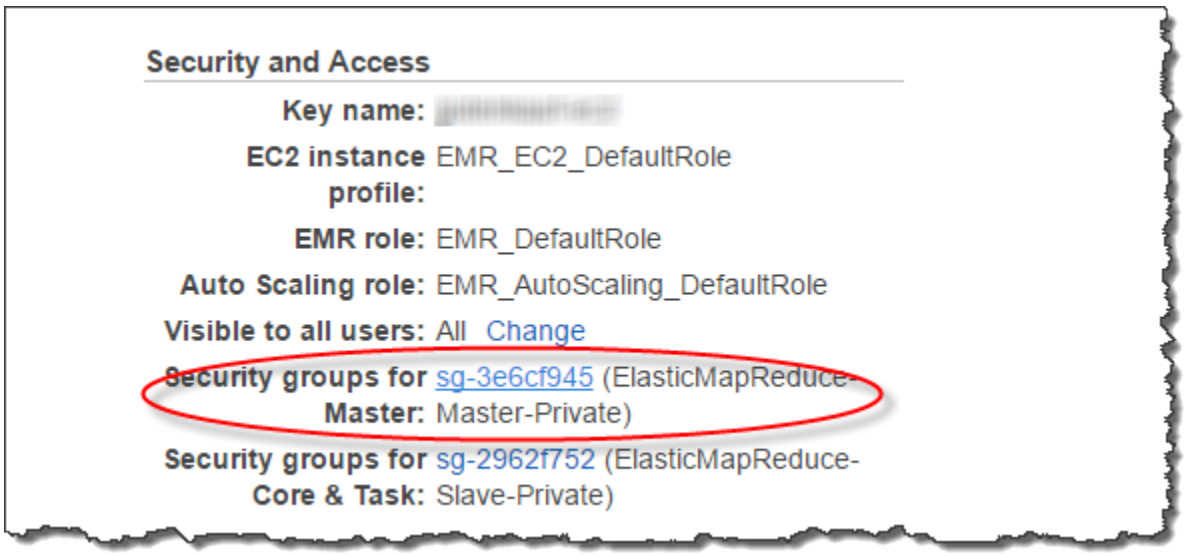
Melihat nama grup keamanan cluster Amazon Redshift Anda

Untuk menampilkan grup keamanan, lakukan hal berikut:

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster dari daftar untuk membuka detailnya.
3. Pilih Properti dan lihat bagian Pengaturan Jaringan dan keamanan.
4. Temukan grup keamanan Anda di grup keamanan VPC dan catat.

Melihat nama grup keamanan node master Amazon EMR

1. Buka cluster Amazon EMR Anda. Untuk informasi selengkapnya, lihat Menggunakan konfigurasi keamanan untuk mengatur keamanan cluster di Panduan Manajemen Amazon EMR.
2. Di bawah Keamanan dan akses, catat nama grup keamanan node master Amazon EMR.



Untuk membuat atau memodifikasi grup keamanan Amazon EC2 untuk mengizinkan koneksi antara Amazon Redshift dan Amazon EMR

1. Di dasbor Amazon EC2, pilih Grup keamanan. Untuk informasi selengkapnya, lihat Aturan grup keamanan di Panduan Pengguna Amazon EC2
2. Pilih Buat grup keamanan.
3. Jika Anda menggunakan VPC, pilih VPC tempat cluster Amazon Redshift dan Amazon EMR Anda berada.
4. Tambahkan aturan inbound.

    1. Untuk Jenis, pilih TCP kustom.

2. Untuk Sumber, pilih K ustom.
3. Masukkan nama grup keamanan Amazon Redshift Anda.

5. Tambahkan aturan inbound lainnya.

1. Untuk Jenis, pilih TCP.
2. Untuk Port Range, masukkan 9083.

> **Note**
>
> Port default untuk EMR HMS adalah 9083. Jika HMS Anda menggunakan port yang berbeda, tentukan port itu dalam aturan masuk dan dalam definisi skema eksternal.

3. Untuk Sumber, pilih K ustom.

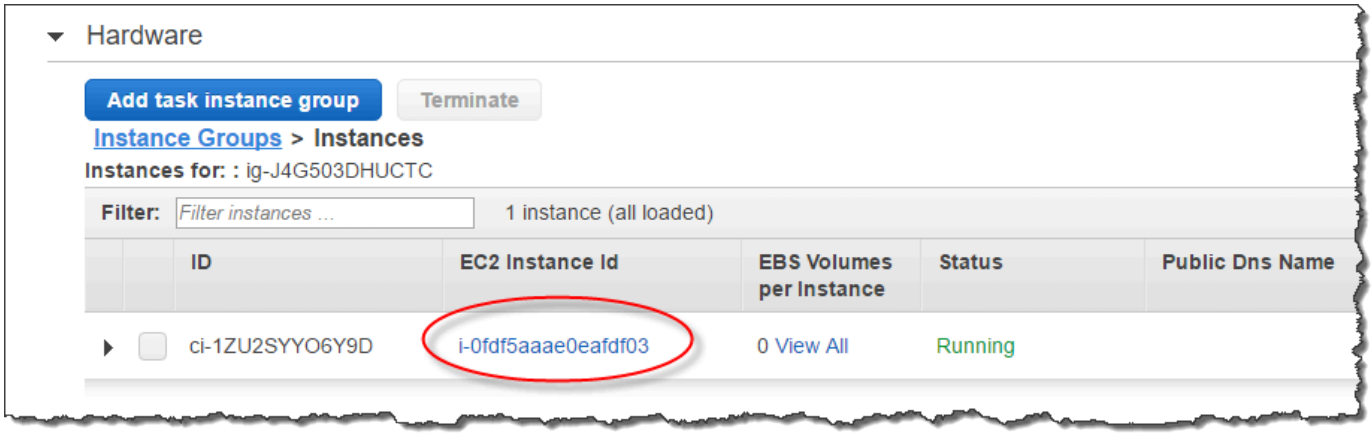
6. Masukkan nama dan deskripsi grup keamanan.
7. Pilih Buat grup keamanan.

Untuk menambahkan grup keamanan Amazon EC2 yang Anda buat dalam prosedur sebelumnya ke cluster Amazon Redshift Anda

1. Di Amazon Redshift, pilih cluster Anda.
2. Pilih Properti.
3. Lihat pengaturan Jaringan dan keamanan dan pilih Edit.
4. Di grup keamanan VPC, pilih nama grup keamanan baru.
5. Pilih Simpan perubahan.

Untuk menambahkan grup keamanan Amazon EC2 ke cluster Amazon EMR Anda

1. Di Amazon EMR, pilih cluster Anda. Untuk informasi selengkapnya, lihat Menggunakan konfigurasi keamanan untuk mengatur keamanan cluster di Panduan Manajemen Amazon EMR.
2. Di bawah Perangkat keras, pilih tautan untuk node Master.
3. Pilih tautan di kolom ID instans EC2.



4. Untuk Tind akan, pilih Keamanan, Ubah grup keamanan.
5. Di grup keamanan terkait, pilih grup keamanan baru, dan pilih Tambahkan grup keamanan.
6. Pilih Simpan.

Tabel eksternal untuk Redshift Spectrum

Topik ini menjelaskan cara membuat dan menggunakan tabel eksternal dengan Redshift Spectrum. Tabel eksternal adalah tabel yang Anda gunakan sebagai referensi untuk mengakses data di luar cluster Amazon Redshift Anda. Tabel ini berisi metadata tentang data eksternal yang dibaca Redshift Spectrum.

Anda membuat tabel eksternal dalam skema eksternal. Untuk membuat tabel eksternal, Anda harus menjadi pemilik skema eksternal atau superuser. Untuk mentransfer kepemilikan skema eksternal, gunakan ALTER SCHEMA untuk mengubah pemilik. Contoh berikut mengubah pemilik `spectrum_schema` skema menjadi`newowner`.

```
alter schema spectrum_schema owner to newowner;
```

Untuk menjalankan kueri Redshift Spectrum, Anda memerlukan izin berikut:

- Izin penggunaan pada skema
- Izin untuk membuat tabel sementara di database saat ini

Contoh berikut memberikan izin penggunaan pada skema `spectrum_schema` ke grup `spectrumusers` pengguna.

```
grant usage on schema spectrum_schema to group spectrumusers;
```

Contoh berikut memberikan izin sementara pada database `spectrumdb` ke grup `spectrumusers` pengguna.

```
grant temp on database spectrumdb to group spectrumusers;
```

Anda dapat membuat tabel eksternal di Amazon Redshift,, Amazon Athena AWS Glue, atau metastore Apache Hive. Untuk informasi selengkapnya, lihat Mem ulai Menggunakan AWS Glue di Panduan Peng AWS Glue embang, Memulai di Panduan Pengguna Amazon Athena, atau Apache Hive di Panduan Pengembang Amazon EMR.

Jika tabel eksternal Anda didefinisikan dalam AWS Glue, Athena, atau metastore Hive, pertama-tama Anda membuat skema eksternal yang mereferensikan database eksternal. Kemudian Anda dapat mereferensikan tabel eksternal dalam pernyataan SELECT Anda dengan mengawali nama tabel dengan nama skema, tanpa perlu membuat tabel di Amazon Redshift. Untuk informasi selengkapnya, lihat Skema eksternal di Amazon Redshift Spectrum.

Untuk mengizinkan Amazon Redshift melihat tabel di AWS Glue Data Catalog, tambahkan `glue:GetTable` ke peran IAM Amazon Redshift. Jika tidak, Anda mungkin mendapatkan kesalahan yang mirip dengan berikut ini.

```
RedshiftIamRoleSession is not authorized to perform: glue:GetTable on resource: *;
```

Misalnya, Anda memiliki tabel eksternal bernama `lineitem_athena` didefinisikan dalam katalog eksternal Athena. Dalam hal ini, Anda dapat menentukan skema eksternal bernama`athena_schema`, lalu kueri tabel menggunakan pernyataan SELECT berikut.

```
select count(*) from athena_schema.lineitem_athena;
```

Untuk menentukan tabel eksternal di Amazon Redshift, gunakan CREATE EXTERNAL TABLE perintah. Pernyataan tabel eksternal mendefinisikan kolom tabel, format file data Anda, dan lokasi data Anda di Amazon S3. Redshift Spectrum memindai file di folder yang ditentukan dan subfolder apa pun. Redshift Spectrum mengabaikan file dan file tersembunyi yang dimulai dengan titik, garis bawah, atau tanda hash (., _, atau #) atau diakhiri dengan tilde (~).

Contoh berikut membuat tabel bernama SALES dalam skema eksternal Amazon Redshift bernama`spectrum`. Data ada dalam file teks yang dibatasi tab.

```
create external table spectrum.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/'
table properties ('numRows'='172000');
```

Untuk melihat tabel eksternal, kueri SVV_EXTERNAL_TABEL tampilan sistem.

Pseudokolom

Secara default, Amazon Redshift membuat tabel eksternal dengan pseudocolumn`$path`,`$size`, dan. `$spectrum_oid` Pilih `$path` kolom untuk melihat jalur ke file data di Amazon S3, dan pilih `$size` kolom untuk melihat ukuran file data untuk setiap baris yang dikembalikan oleh kueri. `$spectrum_oid`Kolom menyediakan kemampuan untuk melakukan kueri berkorelasi dengan Redshift Spectrum. Sebagai contoh, lihat Contoh: Melakukan subquery berkorelasi di Redshift Spectrum. Anda harus membatasi nama `$spectrum_oid` kolom`$path`,`$size`, dan dengan tanda kutip ganda. Klausa SELECT * tidak mengembalikan pseudocolumn. Anda harus secara eksplisit menyertakan nama `$path``$size`,, dan `$spectrum_oid` kolom dalam kueri Anda, seperti yang ditunjukkan contoh berikut.

```
select "$path", "$size", "$spectrum_oid"
from spectrum.sales_part where saledate = '2008-12-01';
```

Anda dapat menonaktifkan pembuatan pseudocolumn untuk sesi dengan mengatur parameter `spectrum_enable_pseudo_columns` konfigurasi `kefalse`. Untuk informasi selengkapnya, lihat Spektrum_enable_pseudo_column. Anda juga dapat menonaktifkan hanya `$spectrum_oid` kolom semu dengan menyetel `enable_spectrum_oid` `kefalse`. Untuk informasi selengkapnya, lihat enable_spectrum_oid. Namun, menonaktifkan `$spectrum_oid` pseudocolumn juga menonaktifkan dukungan untuk kueri berkorelasi dengan Redshift Spectrum.

> **Important**
>
> Mem `$size` ilih`$path`,, `$spectrum_oid` atau dikenakan biaya karena Redshift Spectrum memindai file data di Amazon S3 untuk menentukan ukuran set hasil. Untuk informasi selengkapnya, lihat Harga Amazon Redshift.

Contoh Pseudocolumn

Contoh berikut mengembalikan ukuran total file data terkait untuk tabel eksternal.

```
select distinct "$path", "$size"
from spectrum.sales_part;

 $path                                                                  | $size
------------------------------------------------------------------------+-------
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/ |  1616
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/ |  1444
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/ |  1644
```

Partisi tabel eksternal Redshift Spectrum

Saat Anda mempartisi data Anda, Anda dapat membatasi jumlah data yang dipindai Redshift Spectrum dengan memfilter pada kunci partisi. Anda dapat mempartisi data Anda dengan kunci apa pun.

Praktik umum adalah mempartisi data berdasarkan waktu. Misalnya, Anda dapat memilih untuk mempartisi berdasarkan tahun, bulan, tanggal, dan jam. Jika Anda memiliki data yang berasal dari beberapa sumber, Anda dapat mempartisi berdasarkan pengidentifikasi sumber data dan tanggal.

Prosedur berikut menjelaskan cara mempartisi data Anda.

Untuk mempartisi data Anda

1. Simpan data Anda dalam folder di Amazon S3 sesuai dengan kunci partisi Anda.

   Buat satu folder untuk setiap nilai partisi dan beri nama folder dengan kunci partisi dan nilai. Misalnya, jika Anda mempartisi berdasarkan tanggal, Anda mungkin memiliki folder bernama`saledate=2017-04-01`,`saledate=2017-04-02`, dan seterusnya. Redshift Spectrum memindai file di folder partisi dan subfolder apa pun. Redshift Spectrum mengabaikan

file dan file tersembunyi yang dimulai dengan titik, garis bawah, atau tanda hash (., _, atau #) atau diakhiri dengan tilde (~).

2. Buat tabel eksternal dan tentukan kunci partisi dalam klausa PARTITIONED BY.

   Kunci partisi tidak bisa menjadi nama kolom tabel. Tipe data dapat berupa SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, atau tipe data TIMESTAMP.

3. Tambahkan partisi.

   MenggunakanALTER TABLE... TAMBAHKAN PARTISI, tambahkan setiap partisi, tentukan kolom partisi dan nilai kunci, dan lokasi folder partisi di Amazon S3. Anda dapat menambahkan beberapa partisi dalam satu pernyataan ALTER TABLE... ADD. Contoh berikut menambahkan partisi untuk `'2008-01'` dan`'2008-03'`.

```
alter table spectrum.sales_part add
partition(saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/
saledate=2008-01/'
partition(saledate='2008-03-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/
saledate=2008-03/';
```

> **Note**
>
> Jika Anda menggunakan AWS Glue katalog, Anda dapat menambahkan hingga 100 partisi menggunakan pernyataan ALTER TABLE tunggal.

Contoh data partisi

Dalam contoh ini, Anda membuat tabel eksternal yang dipartisi oleh kunci partisi tunggal dan tabel eksternal yang dipartisi oleh dua kunci partisi.

Data sampel untuk contoh ini terletak di bucket Amazon S3 yang memberikan akses baca ke semua AWS pengguna yang diautentikasi. Cluster Anda dan file data eksternal Anda harus sama Wilayah AWS. Bucket data sampel berada di Wilayah AS Timur (Virginia Utara) (us-east-1). Untuk mengakses data menggunakan Redshift Spectrum, cluster Anda juga harus berada di us-east-1. Untuk membuat daftar folder di Amazon S3, jalankan perintah berikut.

```
aws s3 ls s3://redshift-downloads/tickit/spectrum/sales_partition/
```

```
PRE saledate=2008-01/
PRE saledate=2008-03/
PRE saledate=2008-04/
PRE saledate=2008-05/
PRE saledate=2008-06/
PRE saledate=2008-12/
```

Jika Anda belum memiliki skema eksternal, jalankan perintah berikut. Ganti Nama Sumber Daya Amazon (ARN) untuk peran AWS Identity and Access Management (IAM) Anda.

```
create external schema spectrum
from data catalog
database 'spectrumdb'
iam_role 'arn:aws:iam::123456789012:role/myspectrumrole'
create external database if not exists;
```

Contoh 1: Partisi dengan kunci partisi tunggal

Dalam contoh berikut, Anda membuat tabel eksternal yang dipartisi berdasarkan bulan.

Untuk membuat tabel eksternal yang dipartisi berdasarkan bulan, jalankan perintah berikut.

```
create external table spectrum.sales_part(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
partitioned by (saledate char(10))
row format delimited
fields terminated by '|'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales_partition/'
table properties ('numRows'='172000');
```

Untuk menambahkan partisi, jalankan perintah ALTER TABLE berikut.

```
alter table spectrum.sales_part add
partition(saledate='2008-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/'

partition(saledate='2008-03')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/'

partition(saledate='2008-04')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-04/';
```

Untuk memilih data dari tabel yang dipartisi, jalankan kueri berikut.

```
select top 5 spectrum.sales_part.eventid, sum(spectrum.sales_part.pricepaid)
from spectrum.sales_part, event
where spectrum.sales_part.eventid = event.eventid
  and spectrum.sales_part.pricepaid > 30
  and saledate = '2008-01'
group by spectrum.sales_part.eventid
order by 2 desc;
```

```
eventid | sum
--------+---------
   4124 | 21179.00
   1924 | 20569.00
   2294 | 18830.00
   2260 | 17669.00
   6032 | 17265.00
```

Untuk melihat partisi tabel eksternal, kueri SVV_PARTISI_EKSTERNAL tampilan sistem.

```
select schemaname, tablename, values, location from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename  | values      | location

-----------+------------+-------------
+-------------------------------------------------------------------------
spectrum   | sales_part | ["2008-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-01
```

```
spectrum   | sales_part | ["2008-03"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-04
```

Contoh 2: Partisi dengan beberapa kunci partisi

Untuk membuat tabel eksternal yang dipartisi oleh `date` dan`eventid`, jalankan perintah berikut.

```
create external table spectrum.sales_event(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
partitioned by (salesmonth char(10), event integer)
row format delimited
fields terminated by '|'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/salesevent/'
table properties ('numRows'='172000');
```

Untuk menambahkan partisi, jalankan perintah ALTER TABLE berikut.

```
alter table spectrum.sales_event add
partition(salesmonth='2008-01', event='101')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-01/
event=101/'

partition(salesmonth='2008-01', event='102')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-01/
event=102/'

partition(salesmonth='2008-01', event='103')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-01/
event=103/'

partition(salesmonth='2008-02', event='101')
```

```
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-02/
event=101/'

partition(salesmonth='2008-02', event='102')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-02/
event=102/'

partition(salesmonth='2008-02', event='103')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-02/
event=103/'

partition(salesmonth='2008-03', event='101')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-03/
event=101/'

partition(salesmonth='2008-03', event='102')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-03/
event=102/'

partition(salesmonth='2008-03', event='103')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-03/
event=103/';
```

Jalankan kueri berikut untuk memilih data dari tabel yang dipartisi.

```
select spectrum.sales_event.salesmonth, event.eventname,
 sum(spectrum.sales_event.pricepaid)
from spectrum.sales_event, event
where spectrum.sales_event.eventid = event.eventid
  and salesmonth = '2008-02'
 and (event = '101'
 or event = '102'
 or event = '103')
group by event.eventname, spectrum.sales_event.salesmonth
order by 3 desc;
```

```
salesmonth | eventname       | sum
-----------+-----------------+--------
2008-02    | The Magic Flute | 5062.00
2008-02    | La Sonnambula   | 3498.00
2008-02    | Die Walkure     |  534.00
```

Memetakan kolom tabel eksternal ke kolom ORC

Anda menggunakan tabel eksternal Amazon Redshift Spectrum untuk menanyakan data dari file dalam format ORC. Format kolom baris yang dioptimalkan (ORC) adalah format file penyimpanan kolumnar yang mendukung struktur data bersarang. Untuk informasi selengkapnya tentang menanyakan data bersarang, lihat Mengkueri Data Bersarang dengan Amazon Redshift Spectrum.

Saat Anda membuat tabel eksternal yang mereferensikan data dalam file ORC, Anda memetakan setiap kolom di tabel eksternal ke kolom dalam data ORC. Untuk melakukannya, Anda menggunakan salah satu metode berikut:

- Pemetaan berdasarkan posisi
- Pemetaan berdasarkan nama kolom

Pemetaan berdasarkan nama kolom adalah default.

Pemetaan berdasarkan posisi

Dengan pemetaan posisi, kolom pertama yang ditentukan dalam tabel eksternal memetakan ke kolom pertama dalam file data ORC, kolom kedua ke yang kedua, dan seterusnya. Pemetaan berdasarkan posisi mengharuskan urutan kolom di tabel eksternal dan dalam file ORC cocok. Jika urutan kolom tidak cocok, maka Anda dapat memetakan kolom berdasarkan nama.

Important

Dalam rilis sebelumnya, Redshift Spectrum menggunakan pemetaan posisi secara default. Jika Anda perlu terus menggunakan pemetaan posisi untuk tabel yang ada, atur properti tabel `orc.schema.resolution` ke`position`, seperti yang ditunjukkan contoh berikut.

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='position');
```

Misalnya, tabel `SPECTRUM.ORC_EXAMPLE` didefinisikan sebagai berikut.

```
create external table spectrum.orc_example(
int_col int,
float_col float,
```

```
nested_col struct<
  "int_col" : int,
  "map_col" : map<int, array<float >>
   >
) stored as orc
location 's3://example/orc/files/';
```

Struktur tabel dapat diabstraksikan sebagai berikut.

```
• 'int_col' : int
• 'float_col' : float
• 'nested_col' : struct
   o 'int_col' : int
   o 'map_col' : map
      - key : int
      - value : array
         - value : float
```

File ORC yang mendasarinya memiliki struktur file berikut.

```
• ORC file root(id = 0)
   o 'int_col' : int (id = 1)
   o 'float_col' : float (id = 2)
   o 'nested_col' : struct (id = 3)
      - 'int_col' : int (id = 4)
      - 'map_col' : map (id = 5)
         - key : int (id = 6)
         - value : array (id = 7)
            - value : float (id = 8)
```

Dalam contoh ini, Anda dapat memetakan setiap kolom di tabel eksternal ke kolom dalam file ORC secara ketat berdasarkan posisi. Berikut ini menunjukkan pemetaan.

Nama kolom tabel eksternal	ID kolom ORC	Nama kolom ORC
int_col	1	int_col
float_col	2	float_col
bersarang_col	3	bersarang_col

Nama kolom tabel eksternal	ID kolom ORC	Nama kolom ORC
bersarang_col.int_col	4	int_col
bersarang_col.map_col	5	map_col
bersarang_col.map_col.key	6	TA
bersarang_col.map_col.value	7	TA
bersarang_col.map_col.value .item	8	TA

Pemetaan berdasarkan nama kolom

Menggunakan pemetaan nama, Anda memetakan kolom dalam tabel eksternal ke kolom bernama dalam file ORC pada tingkat yang sama, dengan nama yang sama.

Misalnya, Anda ingin memetakan tabel dari contoh sebelumnya,`SPECTRUM.ORC_EXAMPLE`, dengan file ORC yang menggunakan struktur file berikut.

```
• ORC file root(id = 0)
   o 'nested_col' : struct (id = 1)
      - 'map_col' : map (id = 2)
         - key : int (id = 3)
         - value : array (id = 4)
            - value : float (id = 5)
      - 'int_col' : int (id = 6)
   o 'int_col' : int (id = 7)
   o 'float_col' : float (id = 8)
```

Menggunakan pemetaan posisi, Redshift Spectrum mencoba pemetaan berikut.

Nama kolom tabel eksternal	ID kolom ORC	Nama kolom ORC
int_col	1	struct
float_col	7	int_col
bersarang_col	8	float_col

Ketika Anda menanyakan tabel dengan pemetaan posisi sebelumnya, perintah SELECT gagal pada validasi tipe karena strukturnya berbeda.

Anda dapat memetakan tabel eksternal yang sama ke kedua struktur file yang ditunjukkan pada contoh sebelumnya dengan menggunakan pemetaan nama kolom. Kolom tabel`int_col`,`float_col`, dan `nested_col` peta dengan nama kolom ke kolom dengan nama yang sama dalam file ORC. Kolom yang `nested_col` diberi nama dalam tabel eksternal adalah `struct` kolom dengan subkolom bernama `map_col` dan`int_col`. Subkolom juga dipetakan dengan benar ke kolom yang sesuai dalam file ORC dengan nama kolom.

Membuat tabel eksternal untuk data yang dikelola di Apache Hudi

Untuk menanyakan data dalam format Apache Hudi Copy On Write (CoW), Anda dapat menggunakan tabel eksternal Amazon Redshift Spectrum. Tabel Hudi Copy On Write adalah kumpulan file Apache Parquet yang disimpan di Amazon S3. Anda dapat membaca tabel Copy On Write (CoW) di Apache Hudi versi 0.5.2, 0.6.0, 0.7.0, 0.8.0, 0.9.0, 0.10.0, 0.10.1, 0.11.0, dan 0.11.1 yang dibuat dan dimodifikasi dengan operasi insert, delete, dan upsert write. Misalnya, tabel bootstrap tidak didukung. Untuk informasi selengkapnya, lihat Copy On Write Table dalam dokumentasi Apache Hudi sumber terbuka.

Saat Anda membuat tabel eksternal yang mereferensikan data dalam format Hudi CoW, Anda memetakan setiap kolom di tabel eksternal ke kolom dalam data Hudi. Pemetaan dilakukan dengan kolom.

Pernyataan bahasa definisi data (DDL) untuk tabel Hudi yang dipartisi dan tidak dipartisi mirip dengan format file Apache Parquet lainnya. Untuk tabel Hudi, Anda mendefinisikan `INPUTFORMAT` sebagai`org.apache.hudi.hadoop.HoodieParquetInputFormat`. `LOCATION`Parameter harus menunjuk ke folder dasar tabel Hudi yang berisi `.hoodie` folder, yang diperlukan untuk menetapkan garis waktu komit Hudi. Dalam beberapa kasus, operasi SELECT pada tabel Hudi mungkin gagal dengan pesan Tidak ditemukan garis waktu komit Hudi yang valid. Jika demikian, periksa apakah `.hoodie` folder berada di lokasi yang benar dan berisi timeline komit Hudi yang valid.

> **Note**
>
> Format Apache Hudi hanya didukung saat Anda menggunakan. AWS Glue Data Catalog Ini tidak didukung saat Anda menggunakan metastore Apache Hive sebagai katalog eksternal.

DDL untuk mendefinisikan tabel yang tidak dipartisi memiliki format berikut.

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

DDL untuk mendefinisikan tabel yang dipartisi memiliki format berikut.

```
CREATE EXTERNAL TABLE tbl_name (columns)
PARTITIONED BY(pcolumn1 pcolumn1-type[,...])
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

Untuk menambahkan partisi ke tabel Hudi yang dipartisi, jalankan perintah ALTER TABLE ADD PARTITION di mana LOCATION parameter menunjuk ke subfolder Amazon S3 dengan file milik partisi.

DDL untuk menambahkan partisi memiliki format berikut.

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION 's3://s3-bucket/prefix/partition-path'
```

Membuat tabel eksternal untuk data yang dikelola di Delta Lake

Untuk menanyakan data di tabel Delta Lake, Anda dapat menggunakan tabel eksternal Amazon Redshift Spectrum.

Untuk mengakses tabel Delta Lake dari Redshift Spectrum, buat manifes sebelum kueri. Manifes Danau Delta berisi daftar file yang membentuk snapshot yang konsisten dari tabel Danau Delta. Dalam tabel yang dipartisi, ada satu manifes per partisi. Tabel Delta Lake adalah kumpulan file Parquet Apache yang disimpan di Amazon S3. Untuk informasi lebih lanjut, lihat Delta Lake dalam dokumentasi Delta Lake sumber terbuka.

Saat Anda membuat tabel eksternal yang mereferensikan data dalam tabel Delta Lake, Anda memetakan setiap kolom di tabel eksternal ke kolom di tabel Delta Lake. Pemetaan dilakukan dengan nama kolom.

DDL untuk tabel Delta Lake yang dipartisi dan tidak dipartisi mirip dengan format file Apache Parquet lainnya. Untuk tabel Delta Lake, Anda mendefinisikan `INPUTFORMAT` sebagai `org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat` dan `OUTPUTFORMAT` sebagai`org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat`. `LOCATION`Parameter harus menunjuk ke folder manifes di folder dasar tabel. Jika operasi SELECT pada meja Danau Delta gagal, untuk alasan yang mungkin lihatBatasan dan pemecahan masalah untuk tabel Delta Lake.

DDL untuk mendefinisikan tabel yang tidak dipartisi memiliki format berikut.

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://s3-bucket/prefix/_symlink_format_manifest'
```

DDL untuk mendefinisikan tabel yang dipartisi memiliki format berikut.

```
CREATE EXTERNAL TABLE tbl_name (columns)
PARTITIONED BY(pcolumn1 pcolumn1-type[,...])
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://s3-bucket>/prefix/_symlink_format_manifest'
```

Untuk menambahkan partisi ke tabel Delta Lake yang dipartisi, jalankan perintah ALTER TABLE ADD PARTITION di mana `LOCATION` parameter menunjuk ke subfolder Amazon S3 yang berisi manifes untuk partisi.

DDL untuk menambahkan partisi memiliki format berikut.

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION
```

```
's3://s3-bucket/prefix/_symlink_format_manifest/partition-path'
```

Atau jalankan DDL yang mengarah langsung ke file manifes Delta Lake.

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION
's3://s3-bucket/prefix/_symlink_format_manifest/partition-path/manifest'
```

Batasan dan pemecahan masalah untuk tabel Delta Lake

Pertimbangkan hal berikut saat menanyakan tabel Delta Lake dari Redshift Spectrum:

- Jika manifes menunjuk ke snapshot atau partisi yang tidak ada lagi, kueri gagal sampai manifes baru yang valid telah dibuat. Misalnya, ini mungkin hasil dari operasi VACUUM pada tabel yang mendasarinya,
- Manifest Danau Delta hanya memberikan konsistensi tingkat partisi.

Tabel berikut menjelaskan beberapa alasan potensial untuk kesalahan tertentu saat Anda menanyakan tabel Delta Lake.

Pesan kesalahan	Kemungkinan alasan
Manifes Delta Lake di bucket s3-bucket-1 tidak dapat berisi entri di bucket s3-bucket-2.	Entri manifes menunjuk ke file dalam bucket Amazon S3 yang berbeda dari yang ditentukan.
File Delta Lake diharapkan berada di folder yang sama.	Entri manifes mengarah ke file yang memiliki awalan Amazon S3 berbeda dari yang ditentukan.
Nama file yang terdaftar di jalur manifest-manifest Delta Lake tidak ditemukan.	File yang tercantum dalam manifes tidak ditemukan di Amazon S3.
Kesalahan saat mengambil manifes Delta Lake.	Manifes tidak ditemukan di Amazon S3.
Jalur S3 tidak valid.	Entri dalam file manifes bukanlah jalur Amazon S3 yang valid, atau file manifes telah rusak.

Menggunakan tabel Apache Iceberg dengan Amazon Redshift

> **Note**
>
> Untuk mencapai kinerja terbaik saat menggunakan tabel Apache Iceberg dengan Amazon Redshift, Anda harus membuat statistik kolom untuk tabel yang digunakan. AWS Glue Untuk informasi selengkapnya, lihat Menghasilkan statistik kolom untuk tabel Iceberg di Panduan Peng AWS Glue embang.

Topik ini menjelaskan cara menggunakan tabel dalam format Apache Iceberg dengan Amazon Redshift. Apache Iceberg adalah format tabel sumber terbuka berkinerja tinggi untuk danau data. Untuk informasi selengkapnya, lihat Apache Iceberg dalam dokumentasi Apache Iceberg.

Anda dapat menanyakan tabel Apache Iceberg yang dikatalogkan AWS Glue Data Catalog dengan Amazon Redshift. Jenis instance RG dan Redshift Serverless menggunakan komputasi mereka sendiri untuk memproses kueri data lake, sedangkan tipe instance RA3 menggunakan Redshift Spectrum. Untuk informasi selengkapnya, lihat Meng kueri Danau Data Anda.

Amazon Redshift menyediakan konsistensi transaksional untuk menanyakan tabel Apache Iceberg. Anda dapat memanipulasi data dalam tabel menggunakan layanan yang sesuai dengan ACID (atomisitas, konsistensi, isolasi, ketahanan) seperti Amazon Athena dan Amazon EMR saat menjalankan kueri menggunakan Amazon Redshift. Amazon Redshift dapat menggunakan statistik tabel yang disimpan dalam metadata Apache Iceberg untuk mengoptimalkan rencana kueri dan mengurangi pemindaian file selama pemrosesan kueri. Dengan Amazon Redshift SQL, Anda dapat menggabungkan tabel Redshift dengan tabel data lake.

Untuk mulai menggunakan tabel Iceberg dengan Amazon Redshift:

1. Buat tabel Apache Iceberg pada AWS Glue Data Catalog database menggunakan layanan yang kompatibel seperti Amazon Athena atau Amazon EMR. Untuk membuat tabel Iceberg menggunakan Athena, lihat Menggunakan tabel Apache Iceberg di Panduan Pengguna Amazon Athena.
2. Buat cluster Amazon Redshift atau kelompok kerja Redshift Serverless dengan peran IAM terkait yang memungkinkan akses ke danau data Anda. Untuk informasi tentang cara membuat cluster atau grup kerja, lihat Mem ulai dengan gudang data yang disediakan Amazon Redshift dan Mem ulai dengan gudang data Redshift Serverless di Panduan Memulai Amazon Redshift.

3. Hubungkan ke cluster atau workgroup menggunakan query editor v2 atau klien SQL pihak ketiga. Untuk informasi tentang cara menyambung menggunakan editor kueri v2, lihat Menghubungkan ke gudang data Amazon Redshift menggunakan alat klien SQL di Panduan Manajemen Amazon Redshift.
4. Buat skema eksternal di database Amazon Redshift Anda untuk database Katalog Data tertentu yang menyertakan tabel Iceberg Anda. Untuk informasi tentang membuat skema eksternal, lihatSkema eksternal di Amazon Redshift Spectrum.
5. Jalankan kueri SQL untuk mengakses tabel Iceberg dalam skema eksternal yang Anda buat.

Pertimbangan saat menggunakan tabel Apache Iceberg dengan Amazon Redshift

Pertimbangkan hal berikut saat menggunakan Amazon Redshift dengan tabel Iceberg:

- Dukungan versi Iceberg — Amazon Redshift mendukung menjalankan kueri terhadap versi tabel Iceberg berikut:
  - Versi 1 mendefinisikan bagaimana tabel analitik besar dikelola menggunakan file data yang tidak dapat diubah.
  - Versi 2 menambahkan kemampuan untuk mendukung pembaruan dan penghapusan tingkat baris sambil menjaga file data yang ada tidak berubah, dan menangani perubahan data tabel menggunakan file hapus.
  - Versi 3 memperkenalkan nilai kolom default, pelacakan garis keturunan baris, dan vektor penghapusan. Nilai default memungkinkan Anda menentukan nilai awal untuk kolom yang diterapkan ketika nilai tidak disediakan secara eksplisit. Garis keturunan baris menyediakan kolom semu yang melacak identitas baris dan riwayat modifikasi. Vektor penghapusan menawarkan representasi penghapusan tingkat baris yang lebih ringkas, memungkinkan pembacaan dan penulisan yang lebih cepat dibandingkan dengan file hapus Iceberg v2. Untuk informasi selengkapnya, lihat the section called “Fitur Iceberg v3”.

  Untuk perbedaan antara tabel versi 1, versi 2, dan versi 3, lihat Perubahan versi format dalam dokumentasi Apache Iceberg.

  Untuk informasi selengkapnya tentang fitur Iceberg v3 yang didukung Amazon Redshift, dan batasan saat ini, lihat. the section called “Fitur Iceberg v3”
- Menambahkan partisi — Anda tidak perlu menambahkan partisi secara manual untuk tabel Apache Iceberg Anda. Partisi baru di tabel Apache Iceberg secara otomatis terdeteksi oleh

Amazon Redshift dan tidak diperlukan operasi manual untuk memperbarui partisi dalam definisi tabel. Setiap perubahan dalam spesifikasi partisi juga secara otomatis diterapkan ke kueri Anda tanpa campur tangan pengguna.

- Menelan data Iceberg ke Amazon Redshift — Anda dapat menggunakan perintah INSERT INTO atau CREATE TABLE AS untuk mengimpor data dari tabel Iceberg Anda ke tabel Amazon Redshift lokal. Saat ini Anda tidak dapat menggunakan perintah COPY untuk menelan konten tabel Apache Iceberg ke tabel Amazon Redshift lokal.
- Tampilan terwu jud — Anda dapat membuat tampilan terwujud pada tabel Apache Iceberg seperti tabel eksternal lainnya di Amazon Redshift. Pertimbangan yang sama untuk format tabel danau data lainnya berlaku untuk tabel Apache Iceberg. Penulisan ulang kueri otomatis dan tampilan terwujud otomatis pada tabel data lake saat ini tidak didukung.
- AWS Lake Formation kontrol akses berbutir halus — Amazon Redshift mendukung kontrol akses berbutir AWS Lake Formation halus pada tabel Apache Iceberg.
- User-defined parameter penanganan data — Amazon Redshift mendukung parameter penanganan data yang ditentukan pengguna pada tabel Apache Iceberg. Anda menggunakan parameter penanganan data yang ditentukan pengguna pada file yang ada untuk menyesuaikan data yang ditanyakan dalam tabel eksternal untuk menghindari kesalahan pemindaian. Parameter ini menyediakan kemampuan untuk menangani ketidakcocokan antara skema tabel dan data aktual pada file. Anda juga dapat menggunakan parameter penanganan data yang ditentukan pengguna pada tabel Apache Iceberg.
- Kueri perjalanan waktu — Kueri perjalanan waktu saat ini tidak didukung dengan tabel Apache Iceberg.
- Harga — Saat Anda mengakses tabel Iceberg dari cluster RG atau kelompok kerja Redshift Serverless, kueri data lake berjalan pada sumber daya komputasi cluster atau workgroup sendiri, sehingga tidak ada biaya terpisah untuk kueri data lake. Saat Anda mengakses tabel Iceberg dari cluster DC2 atau RA3, Anda akan dikenakan harga Redshift Spectrum. Untuk informasi tentang harga, lihat harga https://aws.amazon.com/redshift/pricing/ Amazon Redshift.
- Caching metadata — Caching metadata mengasumsikan file metadata tidak dapat diubah berdasarkan spesifikasi Iceberg. https://iceberg.apache.org/spec/#file-system-operations Kekekalan file metadata adalah persyaratan untuk integritas data di Amazon Redshift.
- Identitas federasi — Identitas federasi tidak didukung saat menulis ke tabel Apache Iceberg. Ini termasuk menggunakan kata kunci SESSION untuk parameter IAM_ROLE saat membuat skema eksternal. Untuk informasi selengkapnya tentang parameter IAM_ROLE, lihat MEMBUAT SKEMA EKSTERNAL.

Jenis data yang didukung dengan tabel Apache Iceberg

Topik ini menjelaskan jenis data yang didukung yang dapat dibaca Redshift Spectrum dari tabel dalam format Apache Iceberg.

Amazon Redshift dapat menanyakan tabel Iceberg yang berisi tipe data berikut:

```
binary
boolean
date
decimal
double
float
int
list
long
map
string
struct
timestamp without time zone
```

Saat Anda membuat dan menentukan tabel Iceberg, gunakan nama tipe data Amazon Redshift dalam pernyataan SQL. Redshift secara otomatis memetakan mereka ke jenis Gunung Es yang sesuai. Untuk informasi selengkapnya tentang tipe data gunung es, lihat Skema untuk Gunung Es dalam dokumentasi Apache Iceberg.

Saat membaca dari tabel Iceberg, tipe data gunung es dipetakan ke dalam tipe data pergeseran merah seperti yang ditunjukkan pada tabel di bawah ini:

Jenis gunung es	Jenis Amazon Redshift	Catatan
`boolean`	`boolean`	-
-	`tinyint`	Tidak didukung untuk tabel Iceberg.
-	`smallint`	Tidak didukung untuk tabel Iceberg.
`int`	`int`	-

Jenis gunung es	Jenis Amazon Redshift	Catatan
long	bigint	-
double	double precision	-
float	real	-
decimal(P, S)	decimal(P, S)	Padalah presisi, S adalah skala.
-	char	Tidak didukung untuk tabel Iceberg.
string	varchar(16384)	String yang lebih 16384 besar dari yang dipotong menjadi. 16384
binary	varbyte(64000)	-
date	date	-
time	-	-
timestamp	timestamp	-
timestamptz	timestampz	-
list<E>	SUPER	-
map<K,V>	SUPER	-
struct<...>	SUPER	-
fixed(L)	-	Jenis fixed(L) saat ini tidak didukung di Redshift Spectrum.
uuid	-	Jenis uuid saat ini tidak didukung di Redshift Spectrum.

Jenis gunung es	Jenis Amazon Redshift	Catatan
variant	-	Amazon Redshift tidak mendukung Iceberg V3.
geometry	-	Amazon Redshift tidak mendukung Iceberg V3.
geography	-	Amazon Redshift tidak mendukung Iceberg V3.
timestamp_ns	-	Amazon Redshift tidak mendukung Iceberg V3.
timestamptz_ns	-	Amazon Redshift tidak mendukung Iceberg V3.
Unknown	-	Amazon Redshift tidak mendukung Iceberg V3.

Jenis data berikut didukung saat membuat tabel Iceberg dari Redshift. Tipe data pergeseran merah dipetakan ke dalam tipe data gunung es seperti yang ditunjukkan pada tabel berikut.

Jenis Amazon Redshift	Alias Amazon Redshift	Jenis gunung es	Catatan
integer	int, int4	int	-
bigint	int8	long	-
decimal	numeric	decimal(p,S)	-
real	float4	float	-
double precision	float8, float	double	-

Jenis Amazon Redshift	Alias Amazon Redshift	Jenis gunung es	Catatan
varchar	characte r varying,n varchar, text	string	Tipe varchar(n) data tidak didukung saat membuat tabel Iceberg.
date	-	date	-
timestamp	-	timestamp	-
timestamptz	-	timestamptz	-
boolean	-	boolean	-

Saat menulis ke tabel Iceberg, selain tipe data yang disebutkan dalam tabel sebelumnya, beberapa tipe data sumber dipromosikan ke jenis Gunung Es yang kompatibel seperti yang ditunjukkan pada tabel berikut.

Jenis Amazon Redshift	Jenis gunung es
tinyint	int
smallint	int
varchar(n)	string

Mencoba menggunakan tipe data yang tidak didukung akan mengakibatkan kesalahan sintaks. Saat Anda membuat tabel Iceberg dengan CREATE TABLE AS SELECT klausa, Anda dapat menambahkan cast eksplisit untuk mengatasi perbedaan tipe.

Misalnya, Anda memiliki tabel Redshift RMS dengan skema berikut:

```
CREATE TABLE rms_t (c1 int, c2 char(20));
```

Jika Anda ingin membuat tabel Iceberg menggunakan rms_t sebagai sumber, Anda memerlukan cast eksplisit untuk c2 kolom, karena jen varchar(n) isnya tidak didukung:

```
CREATE TABLE ext_schema.iceberg_t AS SELECT c1, c2::varchar FROM rms_t;
```

Untuk informasi selengkapnya tentang tipe data di Amazon Redshift, lihatJenis Data.

Mereferensikan tabel Iceberg di Amazon Redshift

Amazon Redshift menyediakan beberapa cara untuk mereferensikan tabel Apache Iceberg yang disimpan di danau data Anda. Anda dapat menggunakan skema eksternal untuk membuat referensi ke database Katalog Data yang berisi tabel Iceberg, atau menggunakan notasi tiga bagian untuk akses langsung ke katalog yang dipasang secara otomatis.

Menggunakan skema eksternal untuk mereferensikan tabel Iceberg

Skema eksternal menyediakan cara untuk mereferensikan tabel di Katalog Data Anda dari dalam Amazon Redshift. Saat membuat skema eksternal, Anda membuat koneksi antara database Amazon Redshift dan database Katalog Data tertentu yang berisi tabel Iceberg Anda.

Untuk membuat skema eksternal untuk tabel Iceberg:

```
CREATE EXTERNAL SCHEMA schema_name
FROM DATA CATALOG
DATABASE 'glue_database_name'
IAM_ROLE 'arn:aws:iam::account-id:role/role-name';
```

Setelah membuat skema eksternal, Anda dapat menanyakan tabel Iceberg menggunakan notasi dua bagian:

```
SELECT * FROM schema_name.iceberg_table_name;
```

Anda juga dapat bergabung dengan tabel Iceberg dengan tabel Amazon Redshift lokal:

```
SELECT r.customer_id, i.order_date, r.customer_name
FROM local_customers r
JOIN schema_name.iceberg_orders i
ON r.customer_id = i.customer_id;
```

Menggunakan notasi tiga bagian dengan katalog yang dipasang secara otomatis

Dengan notasi tiga bagian, Anda dapat langsung mereferensikan tabel dalam katalog yang dipasang secara otomatis tanpa membuat skema eksternal. Metode ini sangat berguna saat bekerja dengan

bucket tabel Amazon S3 yang digabungkan dengan AWS Lake Formation. Untuk informasi tentang menyiapkan pemasangan otomatis Katalog Data, lihat Meny ederhanakan akses objek eksternal di Amazon Redshift menggunakan pemasangan otomatis. AWS Glue Data Catalog

Sintaks untuk notasi tiga bagian adalah:

```
"catalog_name".database_name.table_name
```

Misalnya, untuk menanyakan tabel Iceberg di katalog tabel Amazon S3 yang dipasang secara otomatis:

```
SELECT * FROM "my_table_bucket@s3tablescatalog".my_database.my_iceberg_table;
```

Untuk informasi selengkapnya tentang mengintegrasikan bucket tabel Amazon S3 dengan Amazon Redshift, lihat Mengintegrasikan Tabel S3 dengan Amazon Redshift di Panduan Pengguna Amazon S3.

Anda juga dapat mereferensikan tabel di bawah katalog root yang dipasang secara otomatis`awsdatacatalog`, yang menyediakan akses langsung ke database dan tabel yang terdaftar di: AWS Glue Data Catalog

```
SELECT * FROM awsdatacatalog.my_database.my_iceberg_table;
```

Untuk informasi selengkapnya tentang menggunakan katalog `awsdatacatalog` root, lihat Meng kueri AWS Glue Data Catalog di Panduan Manajemen Amazon Redshift dan Mengelola ruang nama Katalog Data di Panduan Pengembang. AWS Lake Formation

Anda juga dapat menggunakan `USE` pernyataan untuk menyetel katalog dan database default untuk bucket tabel Amazon S3:

```
USE "my_table_bucket@s3tablescatalog".my_database;
SELECT * FROM my_iceberg_table;
```

Untuk mengatur jalur pencarian untuk resolusi skema dengan bucket tabel Amazon S3:

```
USE "my_table_bucket@s3tablescatalog";
SET search_path TO my_database;
SELECT * FROM my_iceberg_table;
```

> **Note**
>
> Per USE nyataan dan hanya `search_path` didukung untuk`s3tablescatalog`. Mereka tidak bisa digunakan dengan`awsdatacatalog`. Untuk mereferensikan tabel`awsdatacatalog`, gunakan notasi tiga bagian lengkap.

Praktik terbaik untuk mereferensikan tabel Iceberg

Tabel Apache Iceberg adalah entitas logis tunggal yang terdiri dari beberapa file: file metadata root (metadata.json), daftar manifes, file manifes, dan file data (biasanya.). File metadata root berfungsi sebagai titik masuk dan berisi referensi ke semua file lain yang membentuk tabel. Saat Anda memberikan akses Amazon Redshift ke tabel Iceberg, Amazon Redshift menggunakan file metadata root untuk menemukan dan membaca semua file data yang direferensikan. Jika Amazon Redshift memiliki akses ke file metadata root, itu mengasumsikan dan memerlukan akses ke semua file data yang mendasarinya juga. Ini konsisten dengan desain Iceberg, di mana akses tingkat tabel adalah unit otorisasi yang dimaksudkan.

Untuk meningkatkan kinerja kueri, Amazon Redshift menyimpan file metadata Iceberg (termasuk file metadata root, daftar manifes, dan file manifes) dalam memori. File metadata root (metadata.json) divalidasi ulang terhadap Amazon S3 pada interval yang dapat dikonfigurasi (TTL). Setelah TTL berakhir, Amazon Redshift melakukan permintaan Amazon S3 HEAD pada file metadata root untuk memverifikasi bahwa peran IAM masih memiliki akses dan file tersebut belum dimodifikasi. Jika pemeriksaan izin gagal atau file telah berubah, entri yang di-cache diusir dan metadata diambil kembali dari Amazon S3. Karena file metadata root adalah titik masuk untuk semua akses tabel, validasi ulang ini berfungsi sebagai gerbang izin untuk seluruh tabel. Daftar manifes dan file manifes di-cache tanpa validasi ulang TTL independen - validitas aksesnya berasal dari pemeriksaan izin metadata root. Ini berarti bahwa jika Anda mencabut izin Amazon S3 pada tabel Iceberg, kueri dapat terus berhasil selama maksimal 2 menit saat menggunakan metadata yang di-cache.

> **Important**
>
> Amazon S3 memungkinkan Anda mengatur izin pada tingkat objek individual, yang berarti secara teknis dimungkinkan untuk memberikan akses ke metadata tabel Iceberg sambil membatasi akses ke beberapa file data yang mendasarinya. Ini menciptakan inkonsistensi izin yang dapat menyebabkan kegagalan kueri atau kesalahan akses tak terduga di Amazon Redshift.

Amazon Redshift memvalidasi akses ke file metadata root yang di-cache secara berkala tetapi tidak memvalidasi atau menegakkan konsistensi antara izin tingkat metadata dan tingkat file data dalam bucket Amazon S3 Anda. Merupakan tanggung jawab pelanggan untuk memastikan bahwa izin diterapkan secara konsisten di semua file yang membentuk tabel Iceberg.

Untuk menghindari hal ini, pertimbangkan praktik terbaik berikut saat mereferensikan tabel Iceberg di Amazon Redshift:

- Gunakan nama skema deskriptif — Saat membuat skema eksternal, gunakan nama yang dengan jelas menunjukkan sumber dan tujuan data, seperti `sales_data_lake` atau`customer_analytics`.
- Memanfaatkan statistik tabel — Pastikan statistik kolom dibuat untuk tabel Iceberg Anda yang digunakan untuk AWS Glue mengoptimalkan kinerja kueri. Amazon Redshift menggunakan statistik ini untuk perencanaan dan pengoptimalan kueri.
- Pertimbangkan kesegaran data — Tabel gunung es dapat diperbarui oleh layanan lain saat Anda menanyainya. Amazon Redshift menyediakan konsistensi transaksional, memastikan Anda melihat snapshot data yang konsisten selama eksekusi kueri Anda.
- Gunakan izin IAM yang sesuai — Pastikan cluster atau grup kerja Amazon Redshift Anda memiliki izin IAM yang diperlukan untuk mengakses lokasi Amazon S3 tempat tabel Iceberg Anda disimpan, serta metadata Katalog Data.
- Izin tingkat tabel — Berikan izin di tingkat tabel, bukan pada tingkat file individual.
- Izin seragam — Pertahankan akses seragam di seluruh jalur Amazon S3 untuk tabel Iceberg Anda, termasuk semua metadata, manifes, dan file data.
- Hindari kebijakan tingkat objek yang membatasi — Jangan menetapkan kebijakan tingkat objek yang membatasi pada file Parquet individual dalam awalan tabel Iceberg.
- Memahami cache TTL untuk perubahan izin — Saat Anda mencabut izin Amazon S3 pada tabel Iceberg, kueri dapat terus berhasil menggunakan metadata root yang di-cache hingga durasi TTL yang dikonfigurasi (default: 2 menit).
- Memantau kinerja kueri — Gunakan fitur pemantauan kueri Amazon Redshift untuk melacak kinerja kueri terhadap tabel Iceberg dan mengoptimalkan sesuai kebutuhan.

Pola referensi umum

Contoh berikut menunjukkan pola umum untuk mereferensikan tabel Iceberg:

Menggabungkan data di beberapa tabel Iceberg:

```
SELECT
    region,
    SUM(sales_amount) as total_sales,
    COUNT(*) as transaction_count
FROM data_lake.sales_transactions
WHERE transaction_date >= '2024-01-01'
GROUP BY region
ORDER BY total_sales DESC;
```

Menggabungkan tabel Iceberg dengan tabel Amazon Redshift lokal:

```
SELECT
    c.customer_name,
    c.customer_tier,
    SUM(o.order_amount) as total_orders
FROM customers c
JOIN data_lake.order_history o ON c.customer_id = o.customer_id
WHERE o.order_date >= CURRENT_DATE - INTERVAL '30 days'
GROUP BY c.customer_name, c.customer_tier;
```

Menggunakan notasi tiga bagian dengan kueri kompleks:

```
WITH recent_orders AS (
    SELECT customer_id, order_date, order_amount
    FROM "analytics_bucket@s3tablescatalog".ecommerce.orders
    WHERE order_date >= CURRENT_DATE - INTERVAL '7 days'
)
SELECT
    customer_id,
    COUNT(*) as order_count,
    AVG(order_amount) as avg_order_value
FROM recent_orders
GROUP BY customer_id
HAVING COUNT(*) > 1;
```

Menulis ke tabel Apache Iceberg

Dengan Amazon Redshift, Anda dapat membuat dan menulis ke tabel Apache Iceberg yang disimpan di bucket tabel Amazon S3 dan Amazon S3. Menulis data Iceberg langsung dari Amazon Redshift menyederhanakan pengelolaan data Anda dengan menghilangkan alat tambahan. Tabel gunung es harus didaftarkan. AWS Glue Data Catalog

Anda dapat menggunakan arsitektur rumah danau dengan tabel Apache Iceberg sambil secara bersamaan memanfaatkan analisis SQL Amazon Redshift yang kuat di gudang dan danau. Anda juga mendapatkan akses langsung ke fitur-fitur Amazon Redshift canggih seperti tampilan terwujud pada tabel Iceberg Anda, yang secara signifikan meningkatkan kemampuan analitis Anda tanpa menambah kerumitan.

Penulisan Iceberg didukung di cluster yang disediakan Amazon Redshift dan instans Amazon Redshift Serverless.

Topik

- Perintah SQL
- Mengubah definisi tabel
- Semantik transaksi
- Praktik terbaik

Perintah SQL

Tabel Apache Iceberg di Amazon Redshift menyediakan cara ampuh untuk mengelola kumpulan data analitik besar di danau data Anda. Tabel ini mendukung transaksi ACID, evolusi skema, dan kemampuan perjalanan waktu sambil mempertahankan kinerja tinggi untuk beban kerja analitik. Menggunakan tabel Apache Iceberg, Anda dapat mengatur dan mempartisi data secara efisien, mengontrol format file dan kompresi, dan mengintegrasikan dengan mulus dengan AWS layanan lain.

Anda dapat membuat tabel Iceberg yang dipartisi dan tidak dipartisi menggunakan perintah dan. `CREATE TABLE ... USING ICEBERG` `CREATE TABLE ... USING ICEBERG AS SELECT` Anda dapat mereferensikan tabel Iceberg menggunakan notasi skema eksternal (`external_schema.table_name`) atau notasi tiga bagian (`"catalog_name".database_name.table_name`). Contoh-contoh di bagian ini menunjukkan kedua metode tersebut.

Setelah Anda membuat tabel, Anda dapat menambahkan data menggunakan `INSERT` perintah standar. Perlu diingat bahwa meskipun Amazon Redshift bekerja dengan banyak tipe data Iceberg, Anda mungkin perlu mengonversi beberapa format data saat memasukkan informasi.

Anda dapat melihat tabel Iceberg menggunakan `SHOW TABLES` perintah. Jika Anda ingin menghapus tabel dari AWS Glue Data Catalog, Anda dapat menggunakan `DROP TABLE` perintah. Perhatikan bahwa ini hanya menghapus pendaftaran tabel. Data aktual akan tetap di penyimpanan sampai Anda menghapusnya secara terpisah.

Anda juga dapat memodifikasi data yang ada menggunakan DELETEUPDATE,, dan MERGE perintah. Untuk mengubah definisi tabel seperti skema, spesifikasi partisi, dan properti, lihatMengubah definisi tabel. Pernyataan DDL lain yang tidak didokumentasikan di sana tidak didukung untuk tabel Iceberg.

Anda dapat menulis ke tabel Iceberg yang tidak dibuat oleh Amazon Redshift. Namun, ada beberapa batasan:

- Tabel harus berupa tabel Iceberg v2 atau v3.
- Tabel harus menggunakan Parquet sebagai format data default.
- Tabel tidak boleh memiliki kompresi metadata yang disetel ke True.
- Tabel tidak boleh diaktifkan Write-Audit-Publish (WAP).

Bagian berikut menunjukkan sintaks SQL untuk membuat, menyisipkan, memodifikasi, dan mengelola tabel Iceberg di Amazon Redshift.

Daftar Isi

- CREATE TABLE
- BUAT TABEL SEBAGAI PILIHAN
- TAMPILKAN TABEL
- MASUKKAN KE
- DELETE
- UPDATE
- MERGE
- MEJA DROP

CREATE TABLE

```
CREATE TABLE [IF NOT EXISTS] <external_schema>.<table_name> (
  column_name data_type [DEFAULT literal_value] [, ...]
)
USING ICEBERG
[LOCATION 's3://your-bucket-name/prefix/']
[PARTITIONED BY [[column_name | transform_function]], ...]
[TABLE PROPERTIES (['format-version'='<version>'] [,
 'compression_type'='<compression_value>'])]
```

Anda juga dapat menggunakan notasi tiga bagian untuk bucket tabel S3:

```
CREATE TABLE "<table_bucket_name>@s3tablescatalog".<database_name>.<table_name> (
  column_name data_type [DEFAULT literal_value] [, ...]
)
USING ICEBERG
[PARTITIONED BY [[column_name | transform_function]], ...]
[TABLE PROPERTIES (['format-version'='<version>'] [,
 'compression_type'='<compression_value>'])]
```

Untuk katalog `awsdatacatalog` root yang dipasang secara otomatis:

```
CREATE TABLE awsdatacatalog.<database_name>.<table_name> (
  column_name data_type [DEFAULT literal_value] [, ...]
)
USING ICEBERG
LOCATION 's3://your-bucket-name/prefix/'
[PARTITIONED BY [[column_name | transform_function]], ...]
[TABLE PROPERTIES (['format-version'='<version>'] [,
 'compression_type'='<compression_value>'])]
```

Saat menggunakan sintaks skema eksternal, perhatikan bahwa *<external_schema>* harus merupakan nama skema eksternal yang ada di mana tabel eksternal akan dibuat. Untuk informasi selengkapnya tentang cara membuat dan mengelola skema eksternal, lihat MEMBUAT SKEMA EKSTERNAL di dokumentasi Amazon Redshift.

Kla `LOCATION` usa mendefinisikan lokasi tabel untuk tabel Iceberg yang baru dibuat ini. `LOCATION`diperlukan untuk tabel yang dibuat menggunakan skema eksternal atau katalog `awsdatacatalog` root. Seharusnya lokasi kosong, artinya tidak ada objek Amazon S3 yang ada yang berbagi bucket dan awalan yang sama ini. Wilayah bucket Amazon S3 harus berada di Wilayah yang sama dengan cluster Amazon Redshift. Untuk bucket tabel Amazon S3, `LOCATION` tidak dapat ditentukan karena lokasi tabel ditentukan oleh katalog tabel Amazon S3 (`s3tablescatalog`).

Namun, AWS menyediakan metode untuk mereplikasi data dari tabel Iceberg yang disimpan AWS Glue Data Catalog dalam satu Wilayah AWS ke yang berbeda Wilayah AWS, yang memungkinkan Anda untuk mereplikasi penulisan ke Wilayah yang berbeda. Untuk informasi selengkapnya, lihat Mer eplikasi data di seluruh Wilayah AWS.

`PARTITIONED BY`mendefinisikan partisi tabel Iceberg. Amazon Redshift mendukung semua transformasi partisi Iceberg v2 kecuali untuk. `void` Berikut adalah daftar transformasi yang didukung:

- identitas
- keranjang [N]
- memotong [W]
- tahun
- bulan
- hari
- jam

Untuk definisi lengkap transformasi ini dan tipe data yang kompatibel, lihat Transformasi Partisi dalam dokumentasi Apache Iceberg.

Men `PARTITIONED BY` dukung partisi multi-level. Misalnya, Anda dapat menjalankan perintah berikut:

```
CREATE TABLE ...
USING ICEBERG
LOCATION ...
PARTITIONED BY (bucket(16, id), year(ship_date));
```

Namun, Amazon Redshift tidak mendukung penggunaan satu kolom dalam lebih dari satu transformasi. Misalnya, sintaks berikut tidak didukung:

```
CREATE TABLE ...
USING ICEBERG
LOCATION ...
PARTITIONED BY (bucket(16, ship_date), year(ship_date));
```

Kla `TABLE PROPERTIES` usa mendefinisikan properti tabel tambahan untuk tabel Iceberg ini. Properti tabel yang didukung adalah `format-version` dan`compression_type`. Pro `format-version` perti menentukan versi format tabel Iceberg. Nilai yang mungkin adalah `'2'` (default) dan`'3'`. Pro `compression_type` perti mendefinisikan kompresi file data Parquet default. Jika tidak ditentukan, `snappy` digunakan sebagai codec kompresi. Nilai yang mungkin untuk `compression_type` adalah:`zstd`,`brotli`,`gzip`,`snappy`, dan`uncompressed`.

Untuk tabel Iceberg v3, Anda dapat menentukan nilai default untuk kolom menggunakan kata kunci DEFAULT. Nilai kolom default hanya didukung untuk tabel Iceberg v3. Amazon Redshift

mengembalikan kesalahan jika Anda menentukan nilai default pada tabel Iceberg v2. Hanya nilai literal yang didukung sebagai default.

```
CREATE TABLE <external_schema>.<table_name> (
  column_name data_type [DEFAULT literal_value] [, ...]
)
USING ICEBERG
LOCATION 's3://your-bucket-name/prefix/'
[PARTITIONED BY [[column_name | transform_function]], ...]
[TABLE PROPERTIES ('format-version'='3' [, 'compression_type'='<compression_value>'])];
```

Contoh dengan default:

```
-- External schema notation
CREATE TABLE my_external_schema.orders (
  order_id INT,
  status VARCHAR DEFAULT 'pending',
  region VARCHAR DEFAULT 'us-east-1'
)
USING ICEBERG
LOCATION 's3://amzn-s3-demo-bucket/orders/'
TABLE PROPERTIES ('format-version'='3');

-- Three-part notation (awsdatacatalog / Glue)
CREATE TABLE awsdatacatalog.my_glue_db.orders (
  order_id INT,
  status VARCHAR DEFAULT 'pending',
  region VARCHAR DEFAULT 'us-east-1'
)
USING ICEBERG
LOCATION 's3://amzn-s3-demo-bucket/orders/'
TABLE PROPERTIES ('format-version'='3');

-- Three-part notation (S3 Table Buckets)
CREATE TABLE "amzn-s3-demo-bucket@s3tablescatalog".my_namespace.orders (
  order_id INT,
  status VARCHAR DEFAULT 'pending',
  region VARCHAR DEFAULT 'us-east-1'
)
USING ICEBERG
TABLE PROPERTIES ('format-version'='3');
```

> **Note**
>
> `CREATE TABLE ... LIKE ...`tidak didukung untuk tabel Iceberg. Tabel gunung es juga tidak mendukung batasan kolom dan atribut kolom seperti tabel RMS.

Atau, Anda dapat membuat dan mengisi tabel Iceberg dalam satu operasi menggunakan: `CREATE TABLE AS SELECT`

BUAT TABEL SEBAGAI PILIHAN

```
CREATE TABLE <external_schema>.<table_name> [(
  column_name[, ...]
)]
USING ICEBERG
[LOCATION 's3://your-bucket-name/prefix/']
[PARTITIONED BY [[column_name | transform_function]], ...]
[TABLE PROPERTIES ('compression_type'='<compression-value>')]
AS
SELECT query
```

Anda juga dapat menggunakan notasi tiga bagian untuk membuat tabel dalam katalog yang dipasang secara otomatis:

```
CREATE TABLE "<catalog_name>".<database_name>.<table_name> [(
  column_name[, ...]
)]
USING ICEBERG
[LOCATION 's3://your-bucket-name/prefix/']
[PARTITIONED BY [[column_name | transform_function]], ...]
[TABLE PROPERTIES ('compression_type'='<compression-value>')]
AS
SELECT query
```

Ini mirip dengan `CREATE TABLE` pernyataan kecuali `CREATE` yang diikuti oleh `SELECT` pernyataan untuk mengisi tabel dengan hasil `SELECT` kueri.

Anda tidak dapat lagi menentukan tipe data dengan `CREATE TABLE` klausa di sini, karena `SELECT` kueri menentukan tipe data kolom.

Jika ku SELECT eri gagal karena alasan apa pun, kueri ini akan gagal dan tabel Iceberg tidak akan dibuat.

Untuk tabel Iceberg v3, tentukan `'format-version'='3'` dalam klausa TABLE PROPERTIES. Nilai kolom default tidak diwarisi dari tabel sumber.

```
-- External schema notation
CREATE TABLE my_external_schema.orders_backup (
  order_id, status, region
)
USING ICEBERG
LOCATION 's3://amzn-s3-demo-bucket/orders-backup/'
TABLE PROPERTIES ('format-version'='3')
AS
SELECT order_id, status, region
FROM my_external_schema.orders;

-- Three-part notation (awsdatacatalog / Glue)
CREATE TABLE awsdatacatalog.my_glue_db.orders_backup (
  order_id, status, region
)
USING ICEBERG
LOCATION 's3://amzn-s3-demo-bucket/orders-backup/'
TABLE PROPERTIES ('format-version'='3')
AS
SELECT order_id, status, region
FROM awsdatacatalog.my_glue_db.orders;

-- Three-part notation (S3 Table Buckets)
CREATE TABLE "amzn-s3-demo-bucket@s3tablescatalog".my_namespace.orders_backup (
  order_id, status, region
)
USING ICEBERG
TABLE PROPERTIES ('format-version'='3')
AS
SELECT order_id, status, region
FROM "amzn-s3-demo-bucket@s3tablescatalog".my_namespace.orders;
```

Anda dapat melihat struktur tabel Iceberg Anda menggunakan: `SHOW TABLE`

TAMPILKAN TABEL

```
SHOW TABLE <external_schema>.<table_name>
```

Anda juga dapat menggunakan notasi tiga bagian dengan katalog yang dipasang secara otomatis:

```
SHOW TABLE "<catalog_name>".<database_name>.<table_name>
```

`SHOW TABLE` menampilkan `CREATE TABLE` pernyataan untuk tabel Iceberg. Perintah akan menunjukkan hasil yang sesuai berdasarkan jenis tabel. Berikut ini adalah contoh `SHOW TABLE` output untuk tabel Iceberg:

```
CREATE TABLE my_schema.items (id int, price decimal(5, 2))
USING ICEBERG
LOCATION 's3://my_s3_bucket/items/'
PARTITIONED BY (bucket(16, id))
TABLE PROPERTIES ('compression_type'='snappy')
```

> **Note**
>
> Untuk tabel Amazon S3, karena lokasi tabel dikelola oleh katalog tabel Amazon S3, `LOCATION` klausa akan dihilangkan dalam `SHOW TABLE` hasil.

Untuk tabel Iceberg v3, `SHOW TABLE` menampilkan nilai kolom default dalam outputnya.

```
SHOW TABLE my_external_schema.orders;

-- Output includes the default values in the column definitions:
CREATE TABLE my_external_schema.orders (
  id int,
  status varchar DEFAULT 'active',
  priority int DEFAULT 0
)
USING ICEBERG
LOCATION 's3://amzn-s3-demo-bucket/orders/'
TABLE PROPERTIES ('format-version'='3');
```

Setelah membuat tabel, Anda dapat menambahkan data menggunakan`INSERT INTO`:

MASUKKAN KE

```
INSERT INTO <external_schema>.<table_name> [(column_name [, ...])] VALUES (...)
INSERT INTO <external_schema>.<table_name> [(column_name [, ...])] (SELECT query)

-- Using three-part notation for S3 table buckets:
INSERT INTO "<table_bucket_name>@s3tablescatalog".<database_name>.<table_name>
 [(column_name [, ...])] VALUES (...)
INSERT INTO "<table_bucket_name>@s3tablescatalog".<database_name>.<table_name>
 [(column_name [, ...])] (SELECT query)

-- Using three-part notation for the awsdatacatalog root catalog:
INSERT INTO awsdatacatalog.<database_name>.<table_name> [(column_name [, ...])] VALUES
 (...)
INSERT INTO awsdatacatalog.<database_name>.<table_name> [(column_name [, ...])] (SELECT
 query)
```

Anda dapat membuat `INSERT INTO` tabel Iceberg menggunakan sintaks di atas. Jika `VALUES` klausa digunakan, Anda memberikan nilai untuk kolom yang terdaftar oleh`column_name`, atau semua kolom jika `column_name` bagian dihilangkan.

Ketika data dimasukkan ke dalam tabel partisi, baris baru didistribusikan sesuai dengan spesifikasi partisi yang telah ditentukan. Jika karena alasan apa pun ku `SELECT` eri gagal, kueri akan gagal dan tidak ada data yang akan dimasukkan ke dalam tabel Iceberg.

Untuk tabel Iceberg v3 dengan nilai kolom default, jika kolom dengan default dihilangkan dari pernyataan INSERT atau DEFAULT ditentukan sebagai nilai, nilai default ditulis ke file data.

DELETE

Ku `DELETE` eri untuk tabel Iceberg menggunakan `DELETE` sintaks yang ada di tabel RMS:

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
DELETE [ FROM ] iceberg_table
[ { USING } table_name, ... ] [ WHERE condition ]
```

Anda juga dapat menggunakan notasi tiga bagian untuk bucket tabel S3:

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
DELETE [ FROM ] "<table_bucket_name>@s3tablescatalog".<database_name>.<table_name>
[ { USING } table_name, ... ] [ WHERE condition ]
```

Untuk katalog awsdatacatalog root yang dipasang secara otomatis:

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
DELETE [ FROM ] awsdatacatalog.<database_name>.<table_name>
[ { USING } table_name, ... ] [ WHERE condition ]
```

*iceberg_table*Dapat direferensikan menggunakan *<external_schema>.<external_table_name>* formulir, atau menggunakan notasi 3 bagian untuk katalog yang dipasang secara otomatis. Lihat Referensi tabel Iceberg di Amazon Redshift.

Kla *table_name* usa USING dalam akan digunakan untuk bergabung dengan tabel target untuk menghapus baris yang puas dengan WHERE kondisi tersebut. Bisa *table_name* berupa tabel Iceberg atau tabel Amazon Redshift RMS.

Karena Iceberg menggunakan skema partisi tersembunyi, pengguna dapat menggunakan DELETE query untuk menghapus partisi, mencapai efek yang sama seperti ALTER TABLE ... DROP PARTITION ... untuk tabel Hive.

Misalnya, pertimbangkan tabel Iceberg yang dipartisi seperti berikut:

```
CREATE TABLE my_external_schema.lineitem
(l_item_id int,
 l_ship_date varchar,
 ...
)
USING ICEBERG
LOCATION ...
PARTITIONED BY l_ship_date;
```

Kemudian Anda dapat menghapus partisi dengan menggunakan query seperti berikut:

```
DELETE FROM my_external_schema.lineitem WHERE l_ship_date = '20251231';
```

Untuk kueri seperti ini, Amazon Redshift akan mengoptimalkan eksekusi untuk hanya melakukan operasi metadata saja dan korsleting eksekusi. Jadi tidak seperti DELETE kueri normal, kueri penghapusan metadata saja tidak menunjukkan langkah-langkah eksekusi diEXPLAIN:

```
explain DELETE FROM my_external_schema.lineitem WHERE l_ship_date = '20251231';

 QUERY PLAN
------------
```

```
"XN Seq Scan Metadata of my_external_schema.lineitem location: "s3://s3-path//table-
location" format:ICEBERG (cost=0.00..0.01 rows=0 width=0)"
(0 rows)
```

UPDATE

Sintak UPDATE s kueri untuk tabel Iceberg sangat mirip dengan UPDATE sintaks yang ada untuk tabel RMS:

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
UPDATE iceberg_table [ [ AS ] alias ] SET column = { expression } [,...]
[ FROM fromlist ]
[ WHERE condition ]
```

Anda juga dapat menggunakan notasi tiga bagian untuk bucket tabel S3:

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
UPDATE "<table_bucket_name>@s3tablescatalog".<database_name>.<table_name> [ [ AS ]
 alias ] SET column = { expression } [,...]
[ FROM fromlist ]
[ WHERE condition ]
```

Untuk katalog `awsdatacatalog` root yang dipasang secara otomatis:

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
UPDATE awsdatacatalog.<database_name>.<table_name> [ [ AS ] alias ] SET column =
 { expression } [,...]
[ FROM fromlist ]
[ WHERE condition ]
```

*iceberg_table*Dapat direferensikan menggunakan *<external_schema>.<external_table_name>* formulir, atau menggunakan notasi 3 bagian untuk katalog yang dipasang secara otomatis. Lihat Referensi tabel Iceberg di Amazon Redshift.

Anda dapat memperbarui tabel dengan mereferensikan informasi di tabel lain. Buat daftar tabel lain ini dalam klausa FROM atau gunakan subquery sebagai bagian dari kondisi WHERE. Tabel sumber dapat berupa tabel Iceberg atau tabel Amazon Redshift RMS.

UPDATEjuga dapat berjalan pada tabel yang dipartisi. Ketika UPDATE mengubah nilai kolom yang termasuk dalam spesifikasi partisi saat ini, baris baru yang diperbarui akan dimasukkan ke dalam partisi baru berdasarkan nilai yang baru diperbarui.

Misalnya, pertimbangkan tabel Iceberg yang dipartisi seperti berikut:

```
CREATE TABLE my_external_schema.lineitem
(l_item_id int,
 l_ship_date varchar,
 ...
)
USING ICEBERG
LOCATION ...
PARTITIONED BY l_ship_date;

INSERT INTO my_external_schema.lineitem VALUES (10099, '20251231', ...);
```

Dan ketika Anda menjalankan kueri pembaruan berikut:

```
UPDATE my_external_schema.lineitem SET l_ship_date = '20260101'
WHERE l_item_id = 10099;
```

baris dengan `l_item_id` 10099 bergerak dari partisi `20251231` ke partisi `20260101` baru.

Penting juga untuk dicatat bahwa itu mungkin `UPDATE` memiliki beberapa nilai kandidat. Pertimbangkan pertanyaan di bawah ini:

```
CREATE TABLE my_ext_schema.t1(x1 int, y1 int) USING ICEBERG LOCATION ...;
CREATE TABLE my_ext_schema.t2(x2 int, y2 int) USING ICEBERG LOCATION ...;
INSERT INTO my_ext_schema.t1 VALUES (1,10), (2,20), (3,30);
INSERT INTO my_ext_schema.t2 VALUES (2,40), (2,50);
UPDATE my_ext_schema.t1 SET y1=y2 FROM my_ext_schema.t2 WHERE x1=x2;
```

Dalam hal ini, y1 bisa 40 atau 50. Hasilnya adalah nondeterministik. Anda dapat mengatur parameter konfigurasi `error_on_nondeterministic_update` ke true untuk memaksa kesalahan kueri ketika kasus seperti itu terjadi. Ini konsisten dengan `UPDATE` perilaku tabel RMS yang ada. Untuk selengkapnya, lihat error _on_nondeterministic_update.

MERGE

Ku `MERGE` eri secara kondisional menggabungkan baris dari tabel sumber ke tabel target. Ini berbagi sintaks `MERGE` kueri yang sama dengan tabel RMS yang ada:

```
MERGE INTO target_iceberg_table USING source_table [ [ AS ] alias ]
```

```
ON match_condition
[ WHEN MATCHED THEN { UPDATE SET col_name = { expr } [,...] | DELETE }
  WHEN NOT MATCHED THEN INSERT [ ( col_name [,...] ) ]
  VALUES ( { expr } [, ...] )
| REMOVE DUPLICATES ]
```

Anda juga dapat menggunakan notasi tiga bagian untuk bucket tabel S3:

```
MERGE INTO "<table_bucket_name>@s3tablescatalog".<database_name>.<table_name>
 USING source_table [ [ AS ] alias ]
ON match_condition
[ WHEN MATCHED THEN { UPDATE SET col_name = { expr } [,...] | DELETE }
  WHEN NOT MATCHED THEN INSERT [ ( col_name [,...] ) ]
  VALUES ( { expr } [, ...] )
| REMOVE DUPLICATES ]
```

Untuk katalog `awsdatacatalog` root yang dipasang secara otomatis:

```
MERGE INTO awsdatacatalog.<database_name>.<table_name> USING source_table
 [ [ AS ] alias ]
ON match_condition
[ WHEN MATCHED THEN { UPDATE SET col_name = { expr } [,...] | DELETE }
  WHEN NOT MATCHED THEN INSERT [ ( col_name [,...] ) ]
  VALUES ( { expr } [, ...] )
| REMOVE DUPLICATES ]
```

*target_iceberg_table*Dapat direferensikan menggunakan *<external_schema>*.*<external_table_name>* formulir, atau menggunakan notasi 3 bagian untuk katalog yang dipasang secara otomatis. Lihat Referensi tabel Iceberg di Amazon Redshift.

Bisa *source_table* berupa tabel Iceberg atau tabel Amazon Redshift RMS.

Ketika `REMOVE DUPLICATES` digunakan, `MERGE` perintah menggunakan mode yang disederhanakan. Untuk detail lebih lanjut tentang mode yang disederhanakan, silakan merujuk ke dokumen `MERGE` perintah asli.

Saat menjalankan `MERGE` kueri, Amazon Redshift menghasilkan dan menyimpan file data perantara di lokasi tabel target. File-file ini akan dikumpulkan sampah di akhir kueri. Karena itu, `MERGE` query akan memerlukan `DELETE` izin pada bucket Amazon S3 agar berfungsi dengan baik. Kesalahan izin yang tidak mencukupi akan muncul jika operasi pengumpulan sampah gagal. Untuk tabel Amazon

S3 pengumpulan sampah dikelola oleh layanan tabel Amazon S3. Oleh karena itu `DELETE` izin tidak diperlukan untuk menjalankan `MERGE` query.

MEJA DROP

Untuk menghapus tabel Iceberg dari katalog, gunakan `DROP TABLE` perintah:

```
DROP TABLE <external_schema>.<table_name>
```

Anda juga dapat menggunakan notasi tiga bagian dengan katalog yang dipasang secara otomatis:

```
DROP TABLE "<catalog_name>".<database_name>.<table_name>
```

Menjatuhkan tabel Iceberg adalah operasi khusus metadata. Ini menghapus entri tabel dari AWS Glue Data Catalog dan katalog tabel Amazon S3, jika ini adalah tabel Amazon S3. Amazon Redshift tidak membersihkan atau menghapus file data atau file metadata yang ada di bawah lokasi tabel. Anda dapat menggunakan fitur di AWS Glue dan tabel Amazon S3 untuk menghapus file yatim piatu. Untuk AWS Glue, lihat Menghapus file yatim piatu. Untuk tabel Amazon S3, lihat Pem eliharaan tabel.

Mengubah definisi tabel

`ALTER TABLE`pernyataan digunakan untuk mengubah definisi tabel, termasuk skema, spesifikasi partisi, dan properti. Anda dapat menggunakan pernyataan DDL ini untuk mengelola tabel Iceberg Anda secara langsung melalui Amazon Redshift.

Semua `ALTER TABLE` pernyataan mendukung penggunaan referensi skema eksternal atau referensi notasi tiga bagian untuk tabel Iceberg. Dalam contoh sintaks di bawah ini, *`iceberg_table`* dapat berupa salah satu dari berikut ini:

```
-- External schema notation:
<external_schema>.<table_name>

-- Three-part notation for S3 table buckets:
"<table_bucket_name>@s3tablescatalog".<database_name>.<table_name>

-- Three-part notation for the awsdatacatalog root catalog:
awsdatacatalog.<database_name>.<table_name>
```

Untuk informasi selengkapnya, lihat Mereferensikan tabel Iceberg di Amazon Redshift.

Pertimbangkan hal berikut saat menggunakan `ALTER TABLE` pernyataan:

- `ALTER TABLE`pernyataan saat ini hanya mendukung tabel Iceberg v2 atau v3.
- Semua `ALTER TABLE` pernyataan adalah operasi khusus metadata.
- `ALTER TABLE`pernyataan tidak mendukung tabel dengan kolom tipe kompleks.

Daftar Isi

- Meningkatkan ke Iceberg v3
- UBAH TABEL GANTI NAMA KOLOM
- UBAH ADD/DROP KOLOM TABEL
- MENGUBAH TABEL MENGUBAH KOLOM
- UBAH PROPERTI TABEL SET TABEL
- UBAH TABEL TAMBAH, JATUHKAN, dan GANTI BIDANG PARTISI

Meningkatkan ke Iceberg v3

Anda dapat memutakhirkan tabel Iceberg v2 ke v3 menggunakan: `ALTER TABLE SET TABLE PROPERTIES`

```
ALTER TABLE iceberg_table
SET TABLE PROPERTIES ('format-version' = '3');
```

Untuk detail tambahan, lihat the section called “Meningkatkan dari v2 ke v3”.

UBAH TABEL GANTI NAMA KOLOM

```
ALTER TABLE iceberg_table
RENAME [COLUMN] col_name TO new_name;
```

`ALTER TABLE RENAME COLUMN`mengganti nama kolom yang ada. Ini *`col_name`* bisa berupa kolom partisi atau kolom yang tidak dipartisi. Setelah mengganti nama, tipe data kolom dan spesifikasi partisi tidak berubah.

> **Note**
>
> `ALTER TABLE RENAME COLUMN`tidak didukung untuk tabel AWS Lake Formation terdaftar.

UBAH ADD/DROP KOLOM TABEL

```
ALTER TABLE iceberg_table
ADD [COLUMN] col_name col_type;
```

```
ALTER TABLE iceberg_table
DROP [COLUMN] col_name;
```

`ADD COLUMN`menambahkan satu kolom ke tabel Iceberg yang ada. Anda dapat menggunakan jenis data apa pun yang didukung oleh Amazon Redshift Iceberg write. Untuk informasi selengkapnya, lihat Jenis data yang didukung dengan tabel Apache Iceberg.

`ADD COLUMN`adalah operasi khusus metadata. Nilai kolom yang baru ditambahkan pada baris yang ada adalah`NULL`.

`DROP COLUMN`menjatuhkan kolom dari tabel Iceberg yang ada. Untuk tabel yang dipartisi, Anda tidak dapat melepaskan kolom yang termasuk dalam spesifikasi partisi saat ini. Anda harus menghapus bidang partisi yang melibatkan kolom terlebih dahulu sebelum menjatuhkan kolom. Untuk informasi selengkapnya, lihat UBAH TABEL TAMBAH, JATUHKAN, dan GANTI BIDANG PARTISI.

Untuk tabel Iceberg v3, Anda dapat menentukan nilai default saat menambahkan kolom. File data yang ada mengembalikan default yang ditentukan alih-alih NULL untuk kolom yang baru ditambahkan tanpa memerlukan penulisan ulang data. Nilai kolom default hanya didukung untuk tabel Iceberg v3. Amazon Redshift mengembalikan kesalahan jika Anda menentukan nilai default pada tabel Iceberg v2.

```
-- External schema notation
ALTER TABLE my_external_schema.orders
ADD COLUMN priority INT DEFAULT 0;

-- Three-part notation (awsdatacatalog / Glue)
ALTER TABLE awsdatacatalog.my_glue_db.orders
ADD COLUMN priority INT DEFAULT 0;

-- Three-part notation (S3 Table Buckets)
ALTER TABLE "amzn-s3-demo-bucket@s3tablescatalog".my_namespace.orders
ADD COLUMN priority INT DEFAULT 0;
```

Untuk baris yang ada, nilai default diisi untuk kolom yang baru ditambahkan:

```
-- Existing table with rows
SELECT id FROM my_external_schema.orders;
--  1
--  2

-- Add a column with a default
ALTER TABLE my_external_schema.orders
ADD COLUMN status varchar DEFAULT 'active';

-- Existing rows return the default without a data rewrite
SELECT id, status FROM my_external_schema.orders;
--  1 | active
--  2 | active
```

MENGUBAH TABEL MENGUBAH KOLOM

```
ALTER TABLE iceberg_table
    ALTER COLUMN column_name TYPE updated_data_type;
```

`ALTER TABLE ALTER COLUMN`mengubah tipe data kolom yang ada. Hanya pelebaran tipe yang diperbolehkan, bukan penyempitan. Karena data tidak ditulis ulang setelah`ALTER`, penyempitan tipe data dapat menyebabkan luapan saat mengonsumsi data tabel yang ada. Jenis pelebaran berikut diperbolehkan sesuai spesifikasi gunung es:

```
int # bigint
float # double
decimal(P, S) # decimal(P2, S) where P2 > P (scale cannot be changed)
```

Di Amazon Redshift, tipe data dapat memiliki alias. Misalnya, bilangan bulat 4-byte menggunakan nama `int` tipe atau. `int4` Selama mereka dipetakan ke int 4-byte dalam tipe Iceberg, pelebaran tipe ke bilangan bulat 8-byte diperbolehkan.

Misalnya, tabel Iceberg dibuat menggunakan tipe integer 4-byte:

```
CREATE TABLE iceberg_table (cint int) USING ICEBERG LOCATION 's3://your-bucket-name/
prefix/';
```

Hal ini dapat diperluas dengan `ALTER` pernyataan ini:

```
ALTER TABLE iceberg_table ALTER COLUMN cint TYPE int8;
```

Untuk daftar lengkap pemetaan tipe data antara tipe Amazon Redshift dan tipe Iceberg, lihat. Jenis data yang didukung dengan tabel Apache Iceberg

Memperluas jenis kolom yang termasuk dalam spesifikasi partisi yang ada tidak didukung.

Untuk tabel Iceberg v3, Anda dapat mengatur atau menghapus nilai default pada kolom yang ada:

```
ALTER TABLE iceberg_table
ALTER COLUMN column_name SET DEFAULT literal_value;

ALTER TABLE iceberg_table
ALTER COLUMN column_name DROP DEFAULT;
```

SET DEFAULT mengubah nilai default kolom yang ada. Default baru berlaku untuk file data yang ditulis setelah perubahan. File data yang ada terus menggunakan default sebelumnya.

DROP DEFAULT menghapus nilai default yang diterapkan pada penulisan baru, setara dengan menyetel default ke NULL. File data yang tidak berisi kolom terus mengembalikan nilai default awal kolom.

```
-- External schema notation
ALTER TABLE my_external_schema.orders
ALTER COLUMN status SET DEFAULT 'active';
ALTER TABLE my_external_schema.orders
ALTER COLUMN status DROP DEFAULT;

-- Three-part notation (awsdatacatalog / Glue)
ALTER TABLE awsdatacatalog.my_glue_db.orders
ALTER COLUMN status SET DEFAULT 'active';
ALTER TABLE awsdatacatalog.my_glue_db.orders
ALTER COLUMN status DROP DEFAULT;

-- Three-part notation (S3 Table Buckets)
ALTER TABLE "amzn-s3-demo-bucket@s3tablescatalog".my_namespace.orders
ALTER COLUMN status SET DEFAULT 'active';
ALTER TABLE "amzn-s3-demo-bucket@s3tablescatalog".my_namespace.orders
ALTER COLUMN status DROP DEFAULT;
```

UBAH PROPERTI TABEL SET TABEL

```
ALTER TABLE iceberg_table
SET TABLE PROPERTIES (
```

```
 'compression_type' = 'compression_value');
```

Pernyataan ini memungkinkan Anda untuk mengganti properti tabel default. Saat ini satu-satunya properti tabel yang diizinkan untuk pernyataan ini adalah`compression_type`. Anda dapat menulisnya untuk menggunakan jenis kompresi yang berbeda untuk tabel Iceberg file Parquet. Data dimasukkan setelah `ALTER` menggunakan tipe kompresi baru.

Nilai yang mungkin untuk `compression_type` adalah:`brotli`,`gzip`,`snappy`,`uncompressed`, dan`zstd`.

Anda juga dapat memutakhirkan tabel Iceberg v2 yang ada ke v3 dengan mengatur versi format:

```
ALTER TABLE iceberg_table
SET TABLE PROPERTIES ('format-version' = '3');
```

Untuk detail tambahan, lihat the section called “Meningkatkan dari v2 ke v3”.

UBAH TABEL TAMBAH, JATUHKAN, dan GANTI BIDANG PARTISI

```
ALTER TABLE iceberg_table
    ADD PARTITION FIELD column_name | transform_function;
```

```
ALTER TABLE iceberg_table
    DROP PARTITION FIELD column_name | transform_function;
```

```
ALTER TABLE iceberg_table
    REPLACE PARTITION FIELD column_name | transform_function
        WITH column_name | transform_function;
```

Per `[ADD | DROP | REPLACE] PARTITION FIELD` nyataan ini memungkinkan Anda untuk mengubah spesifikasi partisi tabel yang ada, mendukung evolusi partisi melalui Amazon Redshift.

Per `ALTER` nyataan untuk spesifikasi partisi hanya mengubah metadata tabel dan tidak mempartisi ulang data tabel yang ada. Setelah itu`ALTER`, data baru dimasukkan ke dalam tabel mengikuti spesifikasi partisi yang baru ditentukan.

Pertimbangkan batasan berikut untuk pernyataan ini:

- Saat menentukan bidang partisi baru melalui `ADD` atau`REPLACE ... WITH ...`, bidang baru tidak dapat menyertakan kolom yang sudah menjadi bagian dari bidang partisi lainnya. Ini adalah

batasan yang sama seperti ketika Anda menentukan spesifikasi partisi awal di`CREATE TABLE`. Untuk informasi selengkapnya, lihat CREATE TABLE.

Misalnya, ketika Anda memiliki tabel:

```
CREATE TABLE iceberg_table ... PARTITIONED BY year(ship_date) USING ICEBERG ...;
```

Berikut ini gagal karena `ship_date` sudah menjadi bagian dari bidang partisi yang ada:

```
ALTER TABLE iceberg_table ADD PARTITION FIELD bucket(128, ship_date);
```

- Saat menambahkan bidang partisi baru, bidang yang baru ditambahkan selalu diperlakukan sebagai tingkat terakhir partisi tabel. Contoh:

```
CREATE TABLE iceberg_table ... PARTITIONED BY year(ship_date) USING ICEBERG ...;
ALTER TABLE iceberg_table ADD PARTITION FIELD bucket(256, item_id);
```

Spesifikasi partisi tabel sama dengan:

```
CREATE TABLE iceberg_table ... PARTITIONED BY (year(ship_date), bucket(256, item_id))
    USING ICEBERG ...;
```

- Saat menjatuhkan bidang partisi, itu tidak terbatas pada tingkat partisi terakhir. Anda dapat menghapus bidang partisi yang ada. Contoh:

```
CREATE TABLE iceberg_table ... PARTITIONED BY (year(ship_date), bucket(256, item_id))
    USING ICEBERG ...;
ALTER TABLE iceberg_table DROP PARTITION FIELD year(ship_date);
```

Setelah ini`ALTER`, tabel hanya dipartisi oleh`bucket(256, item_id)`.

- Saat memanggil`REPLACE PARTITION FIELD ... WITH ...`, bidang partisi yang akan diganti dapat berupa bidang apa pun dalam spesifikasi dan tidak terbatas pada bidang terakhir. Contoh:

```
CREATE TABLE iceberg_table ... PARTITIONED BY (year(ship_date), bucket(256, item_id))
    USING ICEBERG ...;
ALTER TABLE iceberg_table REPLACE PARTITION FIELD year(ship_date) WITH
 month(ship_date);
```

Setelah ini`ALTER`, partisi tabel menjadi`(month(ship_date), bucket(256, item_id))`.

- `void`Transformasi tidak didukung dalam `ALTER` pernyataan partisi.

Semantik transaksi

Kueri tulis Redshift Iceberg mendukung ACID dan isolasi snapshot. Transaksi tulis memiliki jaminan atomisitas dan tidak menghasilkan pembaruan parSIAL ketika kueri gagal secara tak terduga.

Beberapa transaksi Iceberg dapat berjalan secara bersamaan, dan jika dua transaksi mencoba memodifikasi tabel atau partisi yang sama secara bersamaan, komit transaksi gagal. Ini memastikan integritas data. Ketika ini terjadi, Anda harus menyelesaikan konflik secara manual dan menjalankan kembali kueri yang gagal. Amazon Redshift tidak secara otomatis mencoba lagi dan menyelesaikan konflik.

Kueri penulisan Iceberg tunggal selalu diperlakukan sebagai transaksi komit otomatis tunggal. Ketika kueri penulisan Iceberg, seperti kueri CREATE atau INSERT, disertakan dalam blok transaksi eksplisit, tidak ada kueri lain yang dapat dijalankan dalam blok transaksi yang sama. Transaksi akan gagal.

Berikut adalah beberapa contoh. Contoh pertama menunjukkan bahwa kueri pernyataan tunggal selalu otomatis commit setelah kueri selesai. Dalam skenario ini, Anda membuat tabel pesanan penjualan baru:

```
CREATE TABLE sales_schema.orders (
    order_id int,
    customer_id int,
    order_date date,
    total_amount decimal(10,2)
) USING ICEBERG LOCATION 's3://my-data-lake/sales/orders/';
```

Contoh ini adalah blok transaksi eksplisit untuk memasukkan pesanan pelanggan menggunakan notasi tiga bagian untuk bucket tabel S3. Transaksi tidak melakukan komit otomatis setelah kueri INSERT, melainkan melakukan commit dan menyisipkan data pesanan dengan perintah COMMIT:

```
BEGIN;
INSERT INTO "analytics_bucket@s3tablescatalog".sales_db.orders VALUES (12345, 9876,
 '2024-10-30', 299.99);
COMMIT;
```

Contoh ini adalah skenario rollback blok transaksi eksplisit di mana Anda menguji penyisipan pesanan tetapi memutuskan untuk membatalkannya. Transaksi tidak melakukan komit otomatis

setelah kueri INSERT, melainkan memutar kembali dengan perintah ROLLBACK tanpa memasukkan urutan pengujian.

```
BEGIN;
INSERT INTO sales_schema.orders VALUES (12346, 5432, '2024-10-30', 150.75);
ROLLBACK;
```

Contoh terakhir ini menunjukkan bagaimana, ketika Anda mencoba menjalankan pernyataan lain dalam blok transaksi yang sama dengan kueri INSERT, transaksi gagal tanpa memasukkan data pesanan. Dalam skenario ini, Anda mencoba memasukkan pesanan dan segera menanyakan tabel:

```
BEGIN;
INSERT INTO sales_schema.orders VALUES (12347, 7890, '2024-10-30', 425.50);
SELECT * FROM sales_schema.orders WHERE order_id = 12347;
```

Satu-satunya pengecualian untuk ini adalah `DROP TABLE` pernyataan, yang selalu berperilaku sebagai pernyataan komit otomatis dan tidak dapat berjalan dalam blok transaksi eksplisit. Ini untuk mempertahankan perilaku yang sama seperti `DROP TABLE` pada tabel eksternal. Untuk informasi selengkapnya, lihat DROP TABLE.

> **Note**
>
> SQL tulis iceberg tidak dapat dieksekusi dari dalam prosedur yang disimpan.

Praktik terbaik

Pertimbangkan praktik terbaik berikut saat Anda menulis ke tabel Apache Iceberg:

- Untuk beban kerja penulisan atau streaming yang kecil dan sering, pertimbangkan untuk menggunakan fitur pemadatan yang disediakan oleh AWS Glue Data Catalog atau tabel Amazon S3 untuk mengoptimalkan ukuran file untuk pembacaan.

- Per `DROP TABLE` intah membatalkan pendaftaran tabel dari AWS Glue Data Catalog atau katalog tabel Amazon S3, tetapi file Anda tetap ada. Anda dapat menggunakan fitur di AWS Glue dan tabel Amazon S3 untuk menghapus file yatim piatu. Untuk AWS Glue, lihat Menghapus file yatim piatu. Untuk tabel Amazon S3, lihat Pem eliharaan tabel.

Fitur Apache Iceberg v3 di Amazon Redshift

Amazon Redshift mendukung tabel Apache Iceberg v3. Iceberg v3 memperkenalkan nilai kolom default, pelacakan garis keturunan baris, dan vektor penghapusan. Anda dapat membuat tabel v3 baru atau memutakhirkan tabel v2 yang ada ke v3.

Tabel Iceberg v3 menggunakan sintaks SQL yang sama untuk kueri dan operasi DML (INSERT, DELETE, UPDATE, MERGE) sebagai tabel v2. Fitur yang dijelaskan pada halaman ini khusus untuk Iceberg v3.

Membuat tabel Iceberg v3

Untuk membuat tabel Iceberg v3, tentukan `'format-version'='3'` dalam klausa TABLE PROPERTIES:

```
CREATE TABLE external_schema.table_name (
  column_name data_type [DEFAULT literal_value] [, ...]
)
USING ICEBERG
LOCATION 's3://your-bucket-name/prefix/'
[PARTITIONED BY [[column_name | transform_function]], ...]
TABLE PROPERTIES ('format-version'='3'
    [, 'compression_type'='compression_value']);
```

Contoh:

```
CREATE TABLE my_schema.orders (
  id int,
  status varchar DEFAULT 'pending',
  priority int DEFAULT 0
)
USING ICEBERG
LOCATION 's3://amzn-s3-demo-bucket/orders/'
TABLE PROPERTIES ('format-version'='3');
```

Jika Anda tidak menentukan versi format, Amazon Redshift membuat tabel sebagai Iceberg v2.

Meningkatkan dari v2 ke v3

Anda dapat memutakhirkan tabel Iceberg v2 yang ada ke v3:

```
ALTER TABLE iceberg_table
```

```
SET TABLE PROPERTIES ('format-version' = '3');
```

Memutakhirkan versi format adalah operasi khusus metadata. File data yang ada tidak ditulis ulang. File penghapusan posisi v2 yang ada tetap valid dan diterapkan selama pembacaan. Operasi penulisan pertama setelah peningkatan menghasilkan nilai garis keturunan baris untuk seluruh tabel. Pada operasi DELETE, UPDATE, atau MERGE berikutnya, Amazon Redshift menggabungkan penghapusan posisi v2 ke dalam vektor penghapusan untuk file data yang terpengaruh oleh operasi.

> **Important**
>
> Sebelum memutakhirkan ke Iceberg v3, tin the section called “Batasan” jau bagian ini, karena beberapa fitur belum didukung untuk tabel v3. Karena menurunkan versi dari v3 ke v2 tidak didukung, pastikan bahwa fitur yang tidak didukung tidak akan memengaruhi beban kerja Anda sebelum melanjutkan dengan peningkatan.

Batasan

- Anda dapat menggunakan Iceberg v3 hanya di cluster Amazon Redshift Serverless (kecuali 4 RPU) dan pada cluster yang disediakan yang menggunakan tipe instans RG.
- Anda tidak dapat membaca atau menulis tipe kompleks (struct, list, map, varian) di tabel Iceberg v3.
- Anda tidak dapat menggunakan tipe data berikut dalam tabel Iceberg v3: struct, list, map, varian, geometri, geografi, biner, uuid, waktu, timestamp_ns, timestamptz_ns, dan unknown.
- Anda tidak dapat menggunakan tabel Iceberg v3 yang berisi penghapusan kesetaraan.
- Anda tidak dapat membuat tampilan terwujud pada tabel Iceberg v3.
- Setelah memutakhirkan tabel ke v3, tipe Iceberg memetakan ke `timestamptz` tipe Amazon Redshift TIMESTAMPTZ alih-alih TIMESTAMP (digunakan di v2). Akibatnya, kueri Anda mengembalikan stempel waktu yang menyertakan informasi zona waktu.

Nilai kolom default

Nilai kolom default hanya didukung untuk tabel Iceberg v3. Amazon Redshift mengembalikan kesalahan jika Anda menentukan nilai default pada tabel Iceberg v2.

Nilai default menentukan nilai literal yang dikembalikan ke kolom ketika tidak ada nilai eksplisit yang ada. Dengan ini, Anda dapat menambahkan kolom baru ke tabel yang ada tanpa menulis ulang file

data. Pembacaan file data yang ditulis sebelumnya secara otomatis mengembalikan nilai default untuk kolom baru. Default juga ditulis ketika pernyataan DML menghilangkan kolom atau menentukan DEFAULT sebagai nilainya.

Hanya nilai literal yang didukung sebagai default. Jenis data bersarang tidak mendukung nilai default.

CREATE TABLE AS SELECT tidak mewarisi nilai kolom default dari tabel sumber. Untuk menentukan default pada tabel baru, gunakan ALTER TABLE ALTER COLUMN SET DEFAULT setelah pembuatan.

Mendefinisikan default pada pembuatan tabel

```
CREATE TABLE external_schema.table_name (
  column_name data_type DEFAULT literal_value [, ...]
)
USING ICEBERG
LOCATION '...'
TABLE PROPERTIES ('format-version'='3');
```

Menambahkan kolom dengan default

```
ALTER TABLE iceberg_table
ADD COLUMN column_name data_type DEFAULT literal_value;
```

File data yang ada mengembalikan nilai default untuk kolom yang baru ditambahkan tanpa memerlukan penulisan ulang data.

Mengubah atau menghapus default

```
ALTER TABLE iceberg_table
ALTER COLUMN column_name SET DEFAULT literal_value;

ALTER TABLE iceberg_table
ALTER COLUMN column_name DROP DEFAULT;
```

SET DEFAULT mengubah nilai default kolom yang ada. Default baru berlaku untuk file data yang ditulis setelah perubahan. File data yang ada terus menggunakan default sebelumnya.

DROP DEFAULT menghapus nilai default dari kolom. Setelah operasi ini, penulisan baru tidak lagi menerapkan default untuk kolom. File data yang tidak berisi kolom terus mengembalikan nilai default awal kolom.

Perilaku INSERT

Saat memasukkan baris tanpa menentukan kolom yang memiliki nilai default, nilai default ditulis ke file data.

TAMPILKAN TABEL

SHOW TABLE menampilkan nilai default dalam outputnya.

Garis garis keturunan

Garis keturunan baris hanya didukung untuk tabel Iceberg v3. Ini menyediakan dua kolom semu yang melacak identitas baris dan riwayat modifikasi. Kolom ini dikelola secara otomatis oleh Amazon Redshift. Anda tidak menetapkan nilai-nilai mereka.

Pseudo-columns

Kolom	Jenis data	Deskripsi
`_row_id`	BIGINT	Secara unik mengidentifikasi setiap baris dalam tabel. Ditugaskan secara otomatis pada operasi penulisan.
`_last_updated_sequence_number`	BIGINT	Nomor urutan snapshot dari operasi penulisan terakhir yang mengubah baris.

Menanyakan garis keturunan baris

Kolom garis keturunan baris harus dinamai secara eksplisit dalam daftar SELECT. Mereka tidak termasuk dalam SELECT *.

```
SELECT _row_id, _last_updated_sequence_number, *
FROM my_schema.my_iceberg_v3_table;
```

Anda dapat menggunakan kolom garis keturunan baris dalam klausa WHERE, ORDER BY, GROUP BY, dan JOIN:

```
SELECT *
```

```
FROM my_schema.my_iceberg_v3_table
WHERE _last_updated_sequence_number >= 3
ORDER BY _row_id;
```

Perilaku pada tabel non-v3

Saat menanyakan tabel non-Iceberg v3, `_row_id` dan `_last_updated_sequence_number` mengembalikan NULL.

Perilaku setelah peningkatan v2-ke-v3

Untuk tabel yang ditingkatkan dari Iceberg v2 ke v3, nilai garis keturunan baris tidak segera tersedia untuk data pra-peningkatan. K `_row_id` edu `_last_updated_sequence_number` anya dan mengembalikan NULL hingga operasi penulisan pertama setelah peningkatan, yang menghasilkan nilai garis keturunan baris untuk seluruh tabel. Ini adalah operasi khusus metadata dan tidak menulis ulang file data yang ada.

Tulis perilaku

Garis garis keturunan secara otomatis diisi oleh Amazon Redshift pada semua operasi penulisan (INSERT, CTAS, UPDATE, MERGE). Setiap baris menerima yang unik`_row_id`, dan `_last_updated_sequence_number` mencerminkan nomor urutan snapshot dari komit yang menulis baris.

Batasan

- Kolom garis keturunan baris tidak termasuk dalam SELECT *.
- Garis keturunan baris tidak didukung untuk Iceberg v2 atau tabel sebelumnya.
- Pre-upgrade data dalam tabel yang ditingkatkan v2-ke-v3 mengembalikan NULL untuk kedua kolom semu hingga operasi penulisan pertama setelah peningkatan.

Vektor penghapusan

Vektor penghapusan adalah mekanisme yang digunakan Iceberg v3 untuk melacak penghapusan tingkat baris. Mereka menggantikan file hapus posisi yang digunakan oleh Iceberg v2.

Saat Anda menjalankan DELETE, UPDATE, atau MERGE pada tabel Iceberg v3, Amazon Redshift mencatat posisi baris yang dihapus dalam vektor penghapusan daripada menulis file penghapusan posisi terpisah. Ini ditangani secara otomatis. Tidak ada perubahan pada sintaks SQL yang diperlukan.

Cara kerja vektor penghapusan

Vektor penghapusan adalah bitmap terkompresi yang mencatat posisi baris mana dalam file data yang telah dihapus. Vektor penghapusan disimpan dalam file Puffin di lokasi S3 yang sama dengan data tabel. Setiap file data memiliki paling banyak satu vektor penghapusan.

Saat membaca tabel Iceberg v3, Amazon Redshift secara otomatis menerapkan vektor penghapusan untuk mengecualikan baris yang dihapus dari hasil kueri.

Manfaat dibandingkan file hapus posisi

- Vektor penghapusan lebih kompak daripada file hapus posisi.
- Vektor penghapusan tunggal per file data menghindari akumulasi banyak file hapus kecil.
- Penghapusan berikutnya pada file data yang sama menghasilkan vektor penghapusan baru yang menggabungkan penghapusan sebelumnya dengan posisi yang baru dihapus, mempertahankan vektor penghapusan tunggal per file data.
- Karena vektor penghapusan lebih kompak dan menghindari akumulasi banyak file kecil, membaca dan menulis lebih cepat dibandingkan dengan file hapus posisi Iceberg v2.

Perilaku setelah peningkatan v2-ke-v3

Setelah memutakhirkan tabel dari Iceberg v2 ke v3, file penghapusan posisi v2 yang ada tetap valid dan diterapkan selama pembacaan. Pada operasi penulisan berikutnya (DELETE, UPDATE, atau MERGE), Amazon Redshift menggabungkan penghapusan posisi v2 yang ada ke dalam vektor penghapusan, seperti yang didefinisikan oleh spesifikasi Iceberg.

Batasan

- Vektor penghapusan tidak didukung untuk Iceberg v2 atau tabel sebelumnya.
- Tabel Iceberg v3 tidak dapat kembali ke file hapus posisi untuk operasi penulisan baru. Ini berarti tabel Iceberg v3 dapat membaca file penghapusan posisi yang ada, tetapi tidak ada file penghapusan posisi baru yang dapat ditambahkan. Sebaliknya, tabel Iceberg v3 hanya dapat menambahkan penghapusan baru menggunakan vektor penghapusan.

Performa kueri Amazon Redshift Spectrum

Topik ini menjelaskan cara meningkatkan kinerja kueri Redshift Spectrum.

Lihatlah rencana kueri untuk menemukan langkah-langkah apa yang telah didorong ke lapisan Amazon Redshift Spectrum.

Langkah-langkah berikut terkait dengan kueri Redshift Spectrum:

- Pemindaian S3 Seq
- S3 HashAggregate
- Pemindaian Kueri S3
- Seq Scan PartitionInfo
- Lingkaran Partisi

Contoh berikut menunjukkan rencana kueri untuk kueri yang menggabungkan tabel eksternal dengan tabel lokal. Perhatikan HashAggregate langkah S3 Seq Scan dan S3 yang dijalankan terhadap data di Amazon S3.

```
explain
select top 10 spectrum.sales.eventid, sum(spectrum.sales.pricepaid)
from spectrum.sales, event
where spectrum.sales.eventid = event.eventid
and spectrum.sales.pricepaid > 30
group by spectrum.sales.eventid
order by 2 desc;
```

```
QUERY PLAN


------------------------------------------------------------------------------
XN Limit  (cost=1001055770628.63..1001055770628.65 rows=10 width=31)


  ->  XN Merge  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


        Merge Key: sum(sales.derived_col2)


        ->  XN Network  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)
```

```
              Send to leader

              ->  XN Sort  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

                    Sort Key: sum(sales.derived_col2)

                    ->  XN HashAggregate  (cost=1055770620.49..1055770620.99 rows=200
width=31)

                          ->  XN Hash Join DS_BCAST_INNER  (cost=3119.97..1055769620.49
rows=200000 width=31)

                                Hash Cond: ("outer".derived_col1 = "inner".eventid)

                                ->  XN S3 Query Scan sales  (cost=3010.00..5010.50
rows=200000 width=31)

                                      ->  S3 HashAggregate  (cost=3010.00..3010.50
rows=200000 width=16)

                                            ->  S3 Seq Scan spectrum.sales
location:"s3://redshift-downloads/tickit/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)
                                                  Filter: (pricepaid > 30.00)

                                ->  XN Hash  (cost=87.98..87.98 rows=8798 width=4)

                                      ->  XN Seq Scan on event  (cost=0.00..87.98
rows=8798 width=4)
```

Perhatikan elemen-elemen berikut dalam rencana kueri:

- N `S3 Seq Scan` ode menunjukkan filter `pricepaid > 30.00` diproses di lapisan Redshift Spectrum.

  Node filter di bawah `XN S3 Query Scan` node menunjukkan pemrosesan predikat di Amazon Redshift di atas data yang dikembalikan dari lapisan Redshift Spectrum.

- N `S3 HashAggregate` ode menunjukkan agregasi di lapisan Redshift Spectrum untuk grup dengan klausa (`group by spectrum.sales.eventid`).

Berikut ini adalah cara untuk meningkatkan kinerja Redshift Spectrum:

- Gunakan file data berformat Apache Parquet. Parquet menyimpan data dalam format kolom, sehingga Redshift Spectrum dapat menghilangkan kolom yang tidak diperlukan dari pemindaian. Ketika data dalam format file teks, Redshift Spectrum perlu memindai seluruh file.
- Gunakan beberapa file untuk mengoptimalkan pemrosesan paralel. Jaga ukuran file Anda lebih besar dari 64 MB. Hindari kemiringan ukuran data dengan menjaga file dengan ukuran yang sama. Untuk informasi tentang file dan rekomendasi konfigurasi Apache Parquet, lihat Format File: Konfigurasi dalam Dokumentasi Parquet Apache.
- Gunakan kolom sesedikit mungkin dalam kueri Anda.
- Letakkan tabel fakta besar Anda di Amazon S3 dan simpan tabel dimensi kecil yang sering digunakan di database Amazon Redshift lokal Anda.
- Perbarui statistik tabel eksternal dengan mengatur parameter TABLE PROPERTIES numRows. Gunakan CREATE EXTERNAL TABLE atau ALTER TABLE untuk mengatur parameter TABLE PROPERTIES numRows untuk mencerminkan jumlah baris dalam tabel. Amazon Redshift tidak menganalisis tabel eksternal untuk menghasilkan statistik tabel yang digunakan pengoptimal kueri untuk menghasilkan rencana kueri. Jika statistik tabel tidak disetel untuk tabel eksternal, Amazon Redshift akan menghasilkan rencana eksekusi kueri. Amazon Redshift menghasilkan rencana ini berdasarkan asumsi bahwa tabel eksternal adalah tabel yang lebih besar dan tabel lokal adalah tabel yang lebih kecil.
- Perencana kueri Amazon Redshift mendorong predikat dan agregasi ke lapisan kueri Redshift Spectrum bila memungkinkan. Ketika sejumlah besar data dikembalikan dari Amazon S3, pemrosesan dibatasi oleh sumber daya cluster Anda. Redshift Spectrum diskalakan secara otomatis untuk memproses permintaan besar. Dengan demikian, kinerja Anda secara keseluruhan meningkat setiap kali Anda dapat mendorong pemrosesan ke lapisan Redshift Spectrum.
- Tulis kueri Anda untuk menggunakan filter dan agregasi yang memenuhi syarat untuk didorong ke lapisan Redshift Spectrum.

  Berikut ini adalah contoh dari beberapa operasi yang dapat didorong ke lapisan Redshift Spectrum:

  - KLAUSUL GROUP BY
  - Kondisi perbandingan dan kondisi pencocokan pola, seperti LIKE.
  - Fungsi agregat, seperti COUNT, SUM, AVG, MIN, dan MAX.

- Fungsi string.

  Operasi yang tidak dapat didorong ke lapisan Redshift Spectrum termasuk DISTINCT dan ORDER BY.

- Gunakan partisi untuk membatasi data yang dipindai. Partisi data Anda berdasarkan predikat kueri yang paling umum, lalu pangkas partisi dengan memfilter kolom partisi. Untuk informasi selengkapnya, lihat Partisi tabel eksternal Redshift Spectrum.

  Kueri SVL_S3PARTISI untuk melihat partisi total dan partisi yang memenuhi syarat.

- Gunakan AWS Glue generator statistik untuk menghitung statistik tingkat kolom untuk tabel. AWS Glue Data Catalog Setelah AWS Glue menghasilkan statistik untuk tabel di Katalog Data, Amazon Redshift Spectrum secara otomatis menggunakan statistik tersebut untuk mengoptimalkan rencana kueri. Untuk informasi selengkapnya tentang menghitung statistik tingkat kolom menggunakan AWS Glue, lihat Bek erja dengan statistik kolom di Panduan Peng AWS Glue embang.

Opsi penanganan data

Topik ini menjelaskan cara mengonfigurasi cara Redshift Spectrum menangani data dalam format yang tidak terduga.

Anda dapat mengatur parameter tabel saat Anda membuat tabel eksternal untuk menyesuaikan data yang ditanyakan di tabel eksternal. Jika tidak, kesalahan pemindaian dapat terjadi. Untuk informasi selengkapnya, lihat PROPERTI TABEL diCREATE EXTERNAL TABLE. Sebagai contoh, lihat Contoh penanganan data. Untuk daftar kesalahan, lihatSVL_SPECTRUM_SCAN_ERROR.

Anda dapat mengatur PROPERTI TABEL berikut ketika Anda membuat tabel eksternal untuk menentukan penanganan input untuk data yang ditanyakan di tabel eksternal.

- `column_count_mismatch_handling`untuk mengidentifikasi apakah file berisi nilai lebih sedikit atau lebih untuk baris daripada jumlah kolom yang ditentukan dalam definisi tabel eksternal.
- `invalid_char_handling`untuk menentukan penanganan input untuk karakter yang tidak valid dalam kolom yang berisi VARCHAR, CHAR, dan data string. Saat Anda menentukan REPLACE untuk`invalid_char_handling`, Anda dapat menentukan karakter pengganti yang akan digunakan.
- `numeric_overflow_handling`untuk menentukan penanganan cast overflow dalam kolom yang berisi data integer dan desimal.

- `surplus_bytes_handling`untuk menentukan penanganan input untuk surplus byte dalam kolom yang berisi data VARBYTE.
- `surplus_char_handling`untuk menentukan penanganan input untuk kelebihan karakter dalam kolom yang berisi VARCHAR, CHAR, dan data string.

Anda dapat mengatur opsi konfigurasi untuk membatalkan kueri yang melebihi jumlah kesalahan maksimum. Untuk informasi selengkapnya, lihat spectrum_query_maxerror.

Contoh: Melakukan subquery berkorelasi di Redshift Spectrum

Topik ini menjelaskan cara melakukan subquery berkorelasi di Redshift Spectrum. Subquery berkorelasi adalah kueri yang menggunakan nilai dari kueri luar.

Anda dapat melakukan subquery berkorelasi di Redshift Spectrum. `$spectrum_oid`Pseudocolumn menyediakan kemampuan untuk melakukan kueri berkorelasi dengan Redshift Spectrum. Untuk melakukan subquery yang berkorelasi, pseudocolumn `$spectrum_oid` harus diaktifkan tetapi tidak muncul dalam pernyataan SQL. Untuk informasi selengkapnya, lihat Pseudokolom.

Untuk membuat skema eksternal dan tabel eksternal untuk contoh ini, lihatMemulai dengan Amazon Redshift Spectrum.

Berikut ini adalah contoh subkueri berkorelasi di Redshift Spectrum.

```
select *
from myspectrum_schema.sales s
where exists
( select *
from myspectrum_schema.listing l
where l.listid = s.listid )
order by salesid
limit 5;
```

```
salesid  listid   sellerid   buyerid   eventid   dateid  qtysold  pricepaid
 commission   saletime
1        1        36861      21191     7872      1875    4        728         109.2
    2008-02-18 02:36:48
2        4        8117       11498     4337      1983    2        76          11.4
    2008-06-06 05:00:16
```

```
3       5       1616      17433     8647      1983    2       350         52.5
    2008-06-06 08:26:17
4       5       1616      19715     8647      1986    1       175         26.25
    2008-06-09 08:38:52
5       6       47402     14115     8240      2069    2       154         23.1
    2008-08-31 09:17:02
```

Metrik di Amazon Redshift Spectrum

Topik ini menjelaskan tampilan sistem yang dapat Anda gunakan untuk memantau kueri data lake.

Anda dapat memantau kueri data lake menggunakan tampilan sistem berikut:

- SVL_S3QUERY

  Gunakan tampilan SVL_S3QUERY untuk mendapatkan detail tentang kueri data lake di segmen dan tingkat potongan node.
- SVL_S3QUERY_SUMMARY

  Gunakan tampilan SVL_S3QUERY_SUMMARY untuk mendapatkan ringkasan dari semua kueri data lake yang telah dijalankan pada sistem.

Berikut ini adalah beberapa hal yang harus dicari di SVL_S3QUERY_SUMMARY:

- Jumlah file yang diproses oleh kueri Redshift Spectrum.
- Jumlah byte yang dipindai dari Amazon S3. Biaya kueri Redshift Spectrum tercermin dalam jumlah data yang dipindai dari Amazon S3.
- Jumlah byte yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Sejumlah besar data yang dikembalikan dapat mempengaruhi kinerja sistem.
- Durasi maksimum dan durasi rata-rata permintaan Redshift Spectrum. Long-running permintaan mungkin menunjukkan hambatan.

Catatan tentang cluster yang disediakan RG

Pada cluster yang disediakan RG, kueri Redshift Spectrum berjalan pada komputasi cluster sendiri daripada pada armada Spectrum khusus yang digunakan oleh cluster yang disediakan RA3 dan DC2. SVL_S3QUERYdan SVL_S3QUERY_SUMMARY terus dihuni pada cluster RG. Kolom berikut memiliki semantik yang berbeda atau tidak digunakan lagi.

Kolom diisi dengan semantik berbeda pada cluster yang disediakan RG

Pada klaster yang disediakan RA3 dan DC2, kolom ini menggambarkan baris, byte, dan unit kerja yang bergerak antara cluster dan armada Spectrum. Pada cluster yang disediakan RG, mereka menggambarkan pekerjaan setara yang dilakukan oleh pembaca asli cluster:

- s3_scanned_rows — baris dibaca langsung dari Amazon S3 oleh pembaca asli cluster (pra-filter).
- s3_scanned_bytes — ukuran rentang pemindaian total dalam byte yang diproses oleh pembaca asli cluster.
- s3query_returned_rows — baris yang dihasilkan setelah pengosongan filter oleh pembaca asli cluster.
- s3query_returned_bytes — byte yang dihasilkan setelah penolakan filter oleh pembaca asli cluster.
- split — jumlah rentang pemindaian yang dikonsumsi oleh pembaca asli cluster.
- total_split_size — ukuran total semua rentang pemindaian yang dikonsumsi, dalam byte.
- max_split_size — ukuran rentang pemindaian terbesar yang dikonsumsi, dalam byte.

Kolom tidak digunakan lagi pada cluster yang disediakan RG

Kolom ini menggambarkan Spectrum-fleet konsep yang tidak ada di RG. Pada cluster RG mereka dicatat sebagai -1 di STL_S3QUERY, dan karena itu muncul sebagai -1 atau tidak bermakna di SVL_S3QUERY dan SVL_S3QUERY_SUMMARY:

- total_retries, max_retries — Pada RG, percobaan ulang terjadi di tingkat klien Amazon S3. Gunakan STL_S3CLIENT dan STL_S3CLIENT_ERROR untuk detail coba lagi.
- max_request_duration, avg_request_duration — RG tidak menggunakan model permintaan Spec trum.
- max_request_parallelism, avg_request_parallelism — RG tidak menggunakan token permintaan Spectrum.
- slowdown_count, max_concurrent_slowdown_count — Di RG, perlambatan Amazon S3 dilacak di STL_S3CLIENT.

Untuk metrik per kueri agregat pada cluster RG (partisi, file yang dipindai, baris dan byte yang dikembalikan, format file, lokasi file, daftar, dan waktu pengambilan partisi), Anda juga dapat menggunakan tampilan pemantauan. SYS_EXTERNAL_QUERY_DETAIL

Pemecahan masalah kueri di Amazon Redshift Spectrum

Topik ini adalah referensi untuk masalah umum yang mungkin Anda temui dengan kueri Amazon Redshift Spectrum.

Untuk melihat kesalahan yang dihasilkan oleh kueri Redshift Spectrum, kueri tabel SVL_S3LOG sistem.

Topik

- Percobaan ulang terlampaui
- Akses dibatasi
- Batas sumber daya terlampaui
- Tidak ada baris yang dikembalikan untuk tabel yang dipartisi
- Tidak ada kesalahan resmi
- Format data yang tidak kompatibel
- Kesalahan sintaks saat menggunakan Hive DDL di Amazon Redshift
- Izin untuk membuat tabel sementara
- Rentang tidak valid
- Nomor versi Parket tidak valid
- Bidang wajib hilang dari informasi partisi eksternal

Percobaan ulang terlampaui

Jika permintaan Amazon Redshift Spectrum habis, permintaan tersebut dibatalkan dan dikirim kembali. Setelah lima percobaan ulang gagal, kueri gagal dengan kesalahan berikut.

```
error:  Spectrum Scan Error: Retries exceeded
```

Kemungkinan penyebabnya meliputi:

- Ukuran file besar (lebih besar dari 1 GB). Periksa ukuran file Anda di Amazon S3 dan cari file besar dan ukuran file miring. Pisahkan file besar menjadi file yang lebih kecil, antara 100 MB dan 1 GB. Cobalah untuk membuat file dengan ukuran yang sama.
- Throughput jaringan lambat. Coba kueri Anda nanti.

Akses dibatasi

Amazon Redshift Spectrum tunduk pada kuota AWS layanan lain. Di bawah penggunaan tinggi, permintaan Redshift Spectrum mungkin diperlukan untuk melambat, mengakibatkan kesalahan berikut.

```
error:  Spectrum Scan Error: Access throttled
```

Dua jenis pelambatan dapat terjadi:

- Akses dibatasi oleh Amazon S3.
- Akses dibatasi oleh AWS KMS.

Konteks kesalahan memberikan detail lebih lanjut tentang jenis pelambatan. Berikut ini, Anda dapat menemukan penyebab dan kemungkinan resolusi untuk pelambatan ini.

Akses dibatasi oleh Amazon S3

Amazon S3 mungkin membatasi permintaan Redshift Spectrum jika tingkat permintaan baca pada awalan terlalu tinggi. Untuk informasi tentang tingkat GET/HEAD permintaan yang dapat Anda capai di Amazon S3, lihat Meng optimalkan Kinerja Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tingkat GET/HEAD permintaan Amazon S3 memperhitungkan semua GET/HEAD permintaan pada awalan sehingga aplikasi berbeda yang mengakses awalan yang sama berbagi tingkat permintaan total.

Jika permintaan Redshift Spectrum Anda sering dibatasi oleh Amazon S3, kurangi jumlah GET/HEAD permintaan Amazon S3 yang dibuat Redshift Spectrum ke Amazon S3. Untuk melakukan ini, coba gabungkan file kecil menjadi file yang lebih besar. Sebaiknya gunakan ukuran file 64 MB atau lebih besar.

Juga pertimbangkan untuk mempartisi tabel Redshift Spectrum Anda untuk mendapatkan manfaat dari pemfilteran awal dan untuk mengurangi jumlah file yang diakses di Amazon S3. Untuk informasi selengkapnya, lihat Partisi tabel eksternal Redshift Spectrum.

Akses dibatasi oleh AWS KMS

Jika Anda menyimpan data di Amazon S3 menggunakan enkripsi sisi server (SSE-S3 atau SSE-KMS), Amazon S3 memanggil operasi API AWS KMS untuk setiap file yang diakses Redshift

Spectrum. Permintaan ini diperhitungkan dalam kuota operasi kriptografi Anda; untuk informasi selengkapnya, lihat Kuota AWS KMS Permintaan. Untuk informasi selengkapnya tentang SSE-S3 dan SSE-KMS, lihat Mel indungi Data Menggunakan Server-Side Enkripsi dan Mel indungi Data Menggunakan Server-Side Enkripsi dengan kunci KMS yang Disimpan AWS KMS di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Langkah pertama untuk mengurangi jumlah permintaan yang dibuat oleh Redshift Spectrum AWS KMS adalah mengurangi jumlah file yang diakses. Untuk melakukan ini, coba gabungkan file kecil menjadi file yang lebih besar. Sebaiknya gunakan ukuran file 64 MB atau lebih besar.

Jika permintaan Redshift Spectrum Anda sering dibatasi AWS KMS, pertimbangkan untuk meminta kenaikan kuota untuk tingkat AWS KMS permintaan Anda untuk operasi kriptografi. Untuk meminta kenaikan kuota, lihat AWS Batas Layanan di Referensi Umum Amazon Web Services.

Batas sumber daya terlampaui

Redshift Spectrum memberlakukan batas atas pada jumlah memori yang dapat digunakan permintaan. Permintaan Redshift Spectrum yang membutuhkan lebih banyak memori gagal, mengakibatkan kesalahan berikut.

```
error:  Spectrum Scan Error: Resource limit exceeded
```

Ada dua alasan umum yang dapat menyebabkan permintaan Redshift Spectrum melebihi tunjangan memorinya:

- Redshift Spectrum memproses sejumlah besar data yang tidak dapat dibagi menjadi potongan yang lebih kecil.
- Langkah agregasi besar diproses oleh Redshift Spectrum.

Sebaiknya gunakan format file yang mendukung pembacaan paralel dengan ukuran split 128 MB atau kurang. Lihat File data untuk kueri di Amazon Redshift Spectrum format file yang didukung dan pedoman umum untuk pembuatan file data. Saat menggunakan format file atau algoritma kompresi yang tidak mendukung pembacaan paralel, sebaiknya jaga ukuran file antara 64 MB dan 128 MB.

Tidak ada baris yang dikembalikan untuk tabel yang dipartisi

Jika kueri Anda mengembalikan nol baris dari tabel eksternal yang dipartisi, periksa apakah partisi telah ditambahkan untuk tabel eksternal ini. Redshift Spectrum hanya memindai file di lokasi Amazon

S3 yang telah ditambahkan secara eksplisit menggunakan`ALTER TABLE … ADD PARTITION`. Kueri SVV_PARTISI_EKSTERNAL tampilan untuk menemukan partisi yang ada. Jalan `ALTER TABLE … ADD PARTITION` kan untuk setiap partisi yang hilang.

Tidak ada kesalahan resmi

Verifikasi bahwa peran IAM untuk cluster memungkinkan akses ke objek file Amazon S3. Jika database eksternal Anda ada di Amazon Athena, verifikasi bahwa peran IAM memungkinkan akses ke sumber daya Athena. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk Amazon Redshift Spectrum.

Format data yang tidak kompatibel

Untuk format file kolumnar, seperti Apache Parquet, tipe kolom disematkan dengan data. Jenis kolom dalam definisi CREATE EXTERNAL TABLE harus sesuai dengan jenis kolom file data. Jika ada ketidakcocokan, Anda menerima kesalahan yang mirip dengan berikut ini:

```
File 'https://s3bucket/location/file has an incompatible Parquet schema
for column ‘s3://s3bucket/location.col1'. Column type: VARCHAR, Par
```

Pesan kesalahan mungkin terpotong karena batas panjang pesan. Untuk mengambil pesan kesalahan lengkap, termasuk nama kolom dan jenis kolom, kueri SVL_S3LOG tampilan sistem.

Contoh berikut menanyakan SVL_S3LOG untuk kueri terakhir yang diselesaikan.

```
select message
from svl_s3log
where query = pg_last_query_id()
order by query,segment,slice;
```

Berikut ini adalah contoh hasil yang menunjukkan pesan kesalahan lengkap.

```
                            message
-----------------------------------------------------------------------------------------
Spectrum Scan Error. File 'https://s3bucket/location/file has an incompatible
Parquet schema for column ' s3bucket/location.col1'.
Column type: VARCHAR, Parquet schema:\noptional int64 l_orderkey [i:0 d:1 r:0]\n
```

Untuk memperbaiki kesalahan, ubah tabel eksternal agar sesuai dengan jenis kolom file Parquet.

Kesalahan sintaks saat menggunakan Hive DDL di Amazon Redshift

Amazon Redshift mendukung bahasa definisi data (DDL) untuk CREATE EXTERNAL TABLE yang mirip dengan Hive DDL. Namun, kedua jenis DDL tidak selalu persis sama. Jika Anda menyalin Hive DDL untuk membuat atau mengubah tabel eksternal Amazon Redshift, Anda mungkin mengalami kesalahan sintaks. Berikut ini adalah contoh perbedaan antara Amazon Redshift dan Hive DDL:

- Amazon Redshift memerlukan tanda kutip tunggal (') di mana Hive DDL mendukung tanda kutip ganda (“).
- Amazon Redshift tidak mendukung tipe data STRING. Gunakan VARCHAR sebagai gantinya.

Izin untuk membuat tabel sementara

Untuk menjalankan kueri Redshift Spectrum, pengguna database harus memiliki izin untuk membuat tabel sementara di database. Contoh berikut memberikan izin sementara pada database `spectrumdb` ke grup `spectrumusers` pengguna.

```
grant temp on database spectrumdb  to group spectrumusers;
```

Untuk informasi selengkapnya, lihat HIBAH.

Rentang tidak valid

Redshift Spectrum mengharapkan bahwa file di Amazon S3 yang termasuk dalam tabel eksternal tidak diganti selama kueri. Jika ini terjadi, itu dapat mengakibatkan kesalahan berikut.

```
Error: HTTP response error code: 416 Message: InvalidRange The requested range is not
 satisfiable
```

Untuk menghindari kesalahan, pastikan file Amazon S3 tidak diganti saat ditanya dengan Redshift Spectrum.

Nomor versi Parket tidak valid

Redshift Spectrum memeriksa metadata dari setiap file Apache Parquet yang diaksesnya. Jika pemeriksaan gagal, itu dapat mengakibatkan kesalahan yang mirip dengan berikut ini:

```
File 'https://s3.region.amazonaws.com/s3bucket/location/file has an invalid version
 number
```

Ada dua alasan umum yang dapat menyebabkan pemeriksaan gagal:

- File Parquet telah diganti selama kueri (lihatRentang tidak valid).
- File Parquet rusak.

Bidang wajib hilang dari informasi partisi eksternal

Saat Anda mencoba menambahkan partisi ke tabel eksternal di katalog eksternal, Anda mungkin mendapatkan kesalahan berikut:

```
Error: The required field (<field_name>) is missing from the external partition
 information. Add missing field in partition and retry. Partition location:
 <partition_path>
```

Kesalahan ini berarti bahwa salah satu partisi di tabel eksternal yang digunakan dalam kueri Anda memiliki informasi metadata partisi yang hilang. Ini dapat terjadi dalam kasus-kasus berikut:

- Anda menambahkan partisi ke tabel eksternal di katalog eksternal, seperti AWS Glue Data Catalog, dengan informasi paruh.
- Anda menanyakan tabel yang dipartisi di Amazon Redshift saat menambahkan atau memperbarui partisi untuk tabel yang sesuai di katalog eksternal, seperti. AWS Glue Data Catalog

Berikut ini adalah bidang yang harus diisi saat mengambil partisi dari: AWS Glue Data Catalog

- StorageDescriptor
  - InputFormat
  - OutputFormat
  - SerDeInfo
- Nilai

Anda dapat menanyakan SVV_PARTISI_EKSTERNAL untuk menemukan partisi yang ada dan melihat detail di bidangnya.

Untuk daftar lengkap bidang AWS Glue Data Catalog partisi, lihat Partisi di Refer AWS Glue ensi API Web.

Tutorial: Menanyakan data bersarang dengan Amazon Redshift Spectrum

Tutorial ini menunjukkan bagaimana untuk query data bersarang dengan Redshift Spectrum. Data bersarang adalah data yang berisi bidang bersarang. Bidang bersarang adalah bidang yang disatukan sebagai entitas tunggal, seperti array, struct, atau objek.

Topik

- Ikhtisar
- Langkah 1: Buat tabel eksternal yang berisi data bersarang
- Langkah 2: Kueri data bersarang Anda di Amazon S3 dengan ekstensi SQL
- Kasus penggunaan data bersarang
- Batasan data bersarang (pratinjau)
- Serialisasi JSON bersarang kompleks

Ikhtisar

Amazon Redshift Spectrum mendukung kueri data bersarang dalam format file Parket, ORC, JSON, dan Ion. Redshift Spectrum mengakses data menggunakan tabel eksternal. Anda dapat membuat tabel eksternal yang menggunakan tipe data yang kompleks`struct`,`array`, dan`map`.

Misalnya, misalkan file data Anda berisi data berikut di Amazon S3 dalam folder bernama. `customers` Meskipun tidak ada elemen root tunggal, setiap objek JSON dalam data sampel ini mewakili baris dalam tabel.

```
{"id": 1,
 "name": {"given": "John", "family": "Smith"},
 "phones": ["123-457789"],
 "orders": [{"shipdate": "2018-03-01T11:59:59.000Z", "price": 100.50},
            {"shipdate": "2018-03-01T09:10:00.000Z", "price": 99.12}]
}
{"id": 2,
 "name": {"given": "Jenny", "family": "Doe"},
 "phones": ["858-8675309", "415-9876543"],
```

```
 "orders": []
}
{"id": 3,
 "name": {"given": "Andy", "family": "Jones"},
 "phones": [],
 "orders": [{"shipdate": "2018-03-02T08:02:15.000Z", "price": 13.50}]
}
```

Anda dapat menggunakan Amazon Redshift Spectrum untuk menanyakan data bersarang dalam file. Tutorial berikut menunjukkan cara melakukannya dengan data Apache Parquet.

Prasyarat

Jika Anda belum menggunakan Redshift Spectrum, ikuti langkah-langkah di Memulai dengan Amazon Redshift Spectrum sebelum melanjutkan.

Untuk membuat skema eksternal, ganti ARN peran IAM dalam perintah berikut dengan peran ARN yang Anda buat di Buat peran IAM. Kemudian jalankan perintah di klien SQL Anda.

```
create external schema spectrum
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
create external database if not exists;
```

Langkah 1: Buat tabel eksternal yang berisi data bersarang

Anda dapat melihat data sumber dengan mengunduhnya dari Amazon S3.

Untuk membuat tabel eksternal untuk tutorial ini, jalankan perintah berikut.

```
CREATE EXTERNAL TABLE spectrum.customers (
  id     int,
  name   struct<given:varchar(20), family:varchar(20)>,
  phones array<varchar(20)>,
  orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/tickit/spectrum/customers/';
```

Pada contoh sebelumnya, tabel eksternal `spectrum.customers` menggunakan tipe `array` data `struct` dan untuk menentukan kolom dengan data bersarang. Amazon Redshift Spectrum

mendukung kueri data bersarang dalam format file Parket, ORC, JSON, dan Ion. `STORED AS`Parameternya `PARQUET` untuk file Apache Parquet. `LOCATION`Parameter harus merujuk ke folder Amazon S3 yang berisi data atau file bersarang. Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.

Anda dapat bersarang `array` dan `struct` mengetik di tingkat mana pun. Misalnya, Anda dapat menentukan kolom bernama `toparray` seperti yang ditunjukkan pada contoh berikut.

```
toparray array<struct<nestedarray:
         array<struct<morenestedarray:
         array<string>>>>>
```

Anda juga dapat `struct` jenis sarang seperti yang ditunjukkan untuk kolom `x` dalam contoh berikut.

```
x struct<a: string,
         b: struct<c: integer,
                   d: struct<e: string>
                  >
        >
```

Langkah 2: Kueri data bersarang Anda di Amazon S3 dengan ekstensi SQL

Redshift Spectrum mendukung kueri `array`map, dan jenis `struct` kompleks melalui ekstensi ke sintaks Amazon Redshift SQL.

Ekstensi 1: Akses ke kolom struct

Anda dapat mengekstrak data dari `struct` kolom menggunakan notasi titik yang menggabungkan nama bidang menjadi jalur. Misalnya, permintaan berikut mengembalikan nama yang diberikan dan keluarga untuk pelanggan. Nama yang diberikan diakses oleh jalur panjang`c.name.given`. Nama keluarga diakses oleh jalan panjang`c.name.family`.

```
SELECT c.id, c.name.given, c.name.family
FROM   spectrum.customers c;
```

Query sebelumnya mengembalikan data berikut.

```
id | given | family
---|-------|-------
```

```
1  | John  | Smith
2  | Jenny | Doe
3  | Andy  | Jones
(3 rows)
```

A `struct` bisa berupa kolom lain`struct`, yang bisa berupa kolom lain`struct`, di tingkat mana pun. Jalur yang mengakses kolom dalam `struct` s bersarang sedemikian dalam bisa sangat panjang. Misalnya, lihat definisi untuk kolom `x` dalam contoh berikut.

```
x struct<a: string,
         b: struct<c: integer,
                   d: struct<e: string>
                  >
        >
```

Anda dapat mengakses data di `e` as`x.b.d.e`.

Ekstensi 2: Mulai dari array dalam klausa FROM

Anda dapat mengekstrak data dari `array` kolom (dan, dengan ekstensi, `map` kolom) dengan menentukan `array` kolom dalam `FROM` klausa sebagai pengganti nama tabel. Ekstensi berlaku untuk `FROM` klausa kueri utama, dan juga `FROM` klausa subquery.

Anda dapat mereferensikan `array` elemen berdasarkan posisi, seperti`c.orders[0]`. (pratinjau)

Dengan menggabungkan mulai `arrays` dengan gabungan, Anda dapat mencapai berbagai jenis unnesting, seperti yang dijelaskan dalam kasus penggunaan berikut.

Menghapus sarang menggunakan sambungan batin

Kueri berikut memilih ID pelanggan dan tanggal pengiriman pesanan untuk pelanggan yang memiliki pesanan. Ekstensi SQL dalam klausa FROM `c.orders o` tergantung pada alias. `c`

```
SELECT c.id, o.shipdate
FROM   spectrum.customers c, c.orders o
```

Untuk setiap pelanggan `c` yang memiliki pesanan, `FROM` klausa mengembalikan satu baris untuk setiap pesanan `o` pelanggan`c`. Baris itu menggabungkan baris pelanggan `c` dan baris pesanan`o`. Kemudian `SELECT` klausa hanya menyimpan `c.id` dan`o.shipdate`. Hasilnya adalah sebagai berikut.

```
id|      shipdate
--|---------------------
1 |2018-03-01  11:59:59
1 |2018-03-01  09:10:00
3 |2018-03-02  08:02:15
(3 rows)
```

Alias c menyediakan akses ke bidang pelanggan, dan alias o menyediakan akses ke bidang pesanan.

Semantiknya mirip dengan SQL standar. Anda dapat menganggap FROM klausa sebagai menjalankan loop bersarang berikut, yang diikuti dengan SELECT memilih bidang yang akan dikeluarkan.

```
for each customer c in spectrum.customers
  for each order o in c.orders
     output c.id and o.shipdate
```

Oleh karena itu, jika pelanggan tidak memiliki pesanan, pelanggan tidak muncul di hasilnya.

Anda juga dapat menganggap ini sebagai FROM klausa yang melakukan a JOIN dengan customers tabel dan orders array. Bahkan, Anda juga dapat menulis query seperti yang ditunjukkan pada contoh berikut.

```
SELECT c.id, o.shipdate
FROM   spectrum.customers c INNER JOIN c.orders o ON true
```

> **Note**
>
> Jika skema bernama c ada dengan tabel bernamaorders, maka c.orders mengacu pada tabelorders, dan bukan kolom array. customers

Menghapus sarang menggunakan gabungan kiri

Kueri berikut menghasilkan semua nama pelanggan dan pesanan mereka. Jika pelanggan belum melakukan pemesanan, nama pelanggan masih dikembalikan. Namun, dalam kasus ini, kolom urutan adalah NULL, seperti yang ditunjukkan pada contoh berikut untuk Jenny Doe.

```
SELECT c.id, c.name.given, c.name.family, o.shipdate, o.price
FROM   spectrum.customers c LEFT JOIN c.orders o ON true
```

Query sebelumnya mengembalikan data berikut.

```
id  |  given  | family  |    shipdate          | price
----|---------|---------|----------------------|--------
 1  |  John   | Smith   | 2018-03-01  11:59:59 | 100.5
 1  |  John   | Smith   | 2018-03-01  09:10:00 |  99.12
 2  |  Jenny  | Doe     |                      |
 3  |  Andy   | Jones   | 2018-03-02  08:02:15 |  13.5
 (4 rows)
```

Ekstensi 3: Mengakses array skalar secara langsung menggunakan alias

Ketika alias p dalam `FROM` klausa berkisar pada array skalar, kueri mengacu pada nilai as. p p Misalnya, kueri berikut menghasilkan pasangan nama pelanggan dan nomor telepon.

```
SELECT c.name.given, c.name.family, p AS phone
FROM   spectrum.customers c LEFT JOIN c.phones p ON true
```

Query sebelumnya mengembalikan data berikut.

```
given  |  family  |  phone
-------|----------|-----------
John   | Smith    | 123-4577891
Jenny  | Doe      | 858-8675309
Jenny  | Doe      | 415-9876543
Andy   | Jones    |
(4 rows)
```

Ekstensi 4: Mengakses elemen peta

Redshift Spectrum memperlakukan tipe `map` data sebagai `array` tipe yang berisi `struct` tipe dengan `key` kolom dan kolom. `value` `key`Harus `ascalar`; nilainya bisa berupa tipe data apa pun.

Misalnya, kode berikut membuat tabel eksternal dengan `map` untuk menyimpan nomor telepon.

```
CREATE EXTERNAL TABLE spectrum.customers2 (
```

```
  id     int,
  name   struct<given:varchar(20), family:varchar(20)>,
  phones map<varchar(20), varchar(20)>,
  orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/tickit/spectrum/customers/';
```

Karena `map` tipe berperilaku seperti `array` tipe dengan kolom `key` dan`value`, Anda dapat memikirkan skema sebelumnya seolah-olah mereka adalah sebagai berikut.

```
CREATE EXTERNAL TABLE spectrum.customers3 (
  id     int,
  name   struct<given:varchar(20), family:varchar(20)>,
  phones array<struct<key:varchar(20), value:varchar(20)>>,
  orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/tickit/spectrum/customers/';
```

Kueri berikut mengembalikan nama pelanggan dengan nomor ponsel dan mengembalikan nomor untuk setiap nama. Kueri peta diperlakukan sebagai setara dengan menanyakan tipe bersarang`array`. `struct` Query berikut hanya mengembalikan data jika Anda telah membuat tabel eksternal seperti yang dijelaskan sebelumnya.

```
SELECT c.name.given, c.name.family, p.value
FROM   spectrum.customers c, c.phones p
WHERE  p.key = 'mobile';
```

> **Note**
>
> The `key` for a `map` adalah `string` untuk jenis file Ion dan JSON.

Kasus penggunaan data bersarang

Topik ini menjelaskan kasus penggunaan untuk data bersarang. Data bersarang adalah data yang berisi bidang bersarang. Bidang bersarang adalah bidang yang disatukan sebagai entitas tunggal, seperti array, struct, atau objek.

Anda dapat menggabungkan ekstensi yang dijelaskan sebelumnya dengan fitur SQL biasa. Kasus penggunaan berikut menggambarkan beberapa kombinasi umum. Contoh-contoh ini membantu menunjukkan bagaimana Anda dapat menggunakan data bersarang. Mereka bukan bagian dari tutorial.

Topik

- Menelan data bersarang
- Menggabungkan data bersarang dengan subkueri
- Bergabung dengan Amazon Redshift dan data bersarang

Menelan data bersarang

Anda dapat menggunakan `CREATE TABLE AS` pernyataan untuk menyerap data dari tabel eksternal yang berisi tipe data kompleks. Kueri berikut mengekstrak semua pelanggan dan nomor telepon mereka dari tabel eksternal, menggunakan`LEFT JOIN`, dan menyimpannya di tabel Amazon Redshift. `CustomerPhones`

```
CREATE TABLE CustomerPhones AS
SELECT  c.name.given, c.name.family, p AS phone
FROM    spectrum.customers c LEFT JOIN c.phones p ON true;
```

Menggabungkan data bersarang dengan subkueri

Anda dapat menggunakan subquery untuk mengumpulkan data bersarang. Contoh berikut menggambarkan pendekatan ini.

```
SELECT c.name.given, c.name.family, (SELECT COUNT(*) FROM c.orders o) AS ordercount
FROM   spectrum.customers c;
```

Data berikut dikembalikan.

```
given   |  family  |  ordercount
--------|----------|-------------
 Jenny  |  Doe     |       0
 John   |  Smith   |       2
 Andy   |  Jones   |       1
 (3 rows)
```

> **Note**
>
> Saat Anda menggabungkan data bersarang dengan mengelompokkan berdasarkan baris induk, cara yang paling efisien adalah yang ditunjukkan pada contoh sebelumnya. Dalam contoh itu, baris bersarang `c.orders` dikelompokkan berdasarkan baris induknya. `c` Atau, jika Anda tahu bahwa `id` unik untuk masing-masing `customer` dan `o.shipdate` tidak pernah nol, Anda dapat menggabungkan seperti yang ditunjukkan pada contoh berikut. Namun, pendekatan ini umumnya tidak seefisien contoh sebelumnya.

```
SELECT    c.name.given, c.name.family, COUNT(o.shipdate) AS ordercount
FROM      spectrum.customers c LEFT JOIN c.orders o ON true
GROUP BY  c.id, c.name.given, c.name.family;
```

Anda juga dapat menulis kueri dengan menggunakan subquery dalam `FROM` klausa yang mengacu pada alias (`c`) dari kueri leluhur dan mengekstrak data array. Contoh berikut menunjukkan pendekatan ini.

```
SELECT c.name.given, c.name.family, s.count AS ordercount
FROM   spectrum.customers c, (SELECT count(*) AS count FROM c.orders o) s;
```

Bergabung dengan Amazon Redshift dan data bersarang

Anda juga dapat menggabungkan data Amazon Redshift dengan data bersarang di tabel eksternal. Misalnya, misalkan Anda memiliki data bersarang berikut di Amazon S3.

```
CREATE EXTERNAL TABLE spectrum.customers2 (
  id      int,
  name    struct<given:varchar(20), family:varchar(20)>,
  phones  array<varchar(20)>,
  orders  array<struct<shipdate:timestamp, item:int>>
);
```

Misalkan juga Anda memiliki tabel berikut di Amazon Redshift.

```
CREATE TABLE prices (
  id int,
  price double precision
```

```
);
```

Kueri berikut menemukan jumlah total dan jumlah pembelian setiap pelanggan berdasarkan sebelumnya. Contoh berikut hanya ilustrasi. Ini hanya mengembalikan data jika Anda telah membuat tabel yang dijelaskan sebelumnya.

```
SELECT   c.name.given, c.name.family, COUNT(o.date) AS ordercount, SUM(p.price) AS
 ordersum
FROM     spectrum.customers2 c, c.orders o, prices p ON o.item = p.id
GROUP BY c.id, c.name.given, c.name.family;
```

Batasan data bersarang (pratinjau)

Topik ini menjelaskan batasan untuk membaca data bersarang dengan Redshift Spectrum. Data bersarang adalah data yang berisi bidang bersarang. Bidang bersarang adalah bidang yang disatukan sebagai entitas tunggal, seperti array, struct, atau objek.

> **Note**
>
> Batasan yang ditandai (pratinjau) dalam daftar berikut hanya berlaku untuk kluster pratinjau yang dibuat di Wilayah berikut.
>
> - AS Timur (Ohio) (us-east-2)
> - AS Timur (Virginia Utara) (us-east-1)
> - AS Barat (California Utara) (us-west-1)
> - Asia Pacific (Tokyo) (ap-northeast-1)
> - Europe (Ireland) (eu-west-1)
> - Eropa (Stockholm) (eu-north-1)
>
> Untuk informasi tentang menyiapkan klaster Pratinjau, lihat Membuat klaster pratinjau di Panduan Manajemen Pergeseran Merah Amazon.

Batasan berikut berlaku untuk data bersarang:

- `map`Tipe `array` atau dapat berisi `map` jenis lain `array` atau selama kueri pada nilai bersarang `arrays` atau `maps` tidak mengembalikan`scalar`. (pratinjau)

- Amazon Redshift Spectrum mendukung tipe data yang kompleks hanya sebagai tabel eksternal.
- Kolom hasil subquery harus tingkat atas. (pratinjau)
- Jika `OUTER JOIN` ekspresi mengacu pada tabel bersarang, itu hanya dapat merujuk ke tabel itu dan array bersarang (dan peta). Jika `OUTER JOIN` ekspresi tidak merujuk ke tabel bersarang, ekspresi dapat merujuk ke sejumlah tabel yang tidak bersarang.
- Jika `FROM` klausa dalam subquery mengacu pada tabel bersarang, itu tidak dapat merujuk ke tabel lain.
- Jika subquery bergantung pada tabel bersarang yang mengacu pada tabel induk, subquery hanya dapat menggunakan tabel induk dalam klausa. `FROM` Anda tidak dapat menggunakan induk dalam klausa lain, seperti klausa `SELECT` atau`WHERE`. Misalnya, kueri berikut tidak berjalan karena `SELECT` klausa subquery mengacu pada tabel induk. `c`

  ```
  SELECT c.name.given
  FROM   spectrum.customers c
  WHERE (SELECT COUNT(c.id) FROM c.phones p WHERE p LIKE '858%') > 1;
  ```

  Kueri berikut berfungsi karena induk hanya `c` digunakan dalam `FROM` klausa subquery.

  ```
  SELECT c.name.given
  FROM   spectrum.customers c
  WHERE (SELECT COUNT(*) FROM c.phones p WHERE p LIKE '858%') > 1;
  ```
- Subquery yang mengakses data bersarang di mana saja selain `FROM` klausa harus mengembalikan satu nilai. Satu-satunya pengecualian adalah `(NOT) EXISTS` operator dalam `WHERE` klausa.
- `(NOT) IN` tidak didukung.
- Kedalaman bersarang maksimum untuk semua jenis bersarang adalah 100. Pembatasan ini berlaku untuk semua format file (Parket, ORC, Ion, dan JSON).
- Subkueri agregasi yang mengakses data bersarang hanya dapat merujuk ke `arrays` dan `maps` dalam `FROM` klausulnya, bukan ke tabel eksternal.
- Menanyakan pseudocolumns dari data bersarang dalam tabel Redshift Spectrum tidak didukung. Untuk informasi selengkapnya, lihat Pseudokolom.
- Saat mengekstrak data dari kolom array atau peta dengan menentukannya dalam `FROM` klausa, Anda hanya dapat memilih nilai dari kolom tersebut jika nilainya. `scalar` Misalnya, kueri berikut keduanya mencoba `SELECT` elemen dari dalam array. Kueri yang memilih `arr.a` berfungsi karena `arr.a` merupakan `scalar` nilai. Kueri kedua tidak berfungsi karena `array` merupakan array yang diekstrak dari `s3.nested table` dalam `FROM` klausa. (pratinjau)

```
SELECT array_column FROM s3.nested_table;

array_column
-----------------
[{"a":1},{"b":2}]

SELECT arr.a FROM s3.nested_table t, t.array_column arr;

arr.a
-----
1

--This query fails to run.
SELECT array FROM s3.nested_table tab, tab.array_column array;
```

Anda tidak dapat menggunakan array atau peta dalam `FROM` klausa yang berasal dari array atau peta lain. Untuk memilih array atau struktur kompleks lainnya yang bersarang di dalam array lain, pertimbangkan untuk menggunakan indeks dalam pernyataan. `SELECT`

Serialisasi JSON bersarang kompleks

Topik ini menunjukkan cara membuat serial data bersarang dalam format JSON. Data bersarang adalah data yang berisi bidang bersarang. Bidang bersarang adalah bidang yang disatukan sebagai entitas tunggal, seperti array, struct, atau objek.

Alternatif untuk metode yang ditunjukkan dalam tutorial ini adalah untuk menanyakan kolom koleksi bersarang tingkat atas sebagai JSON serial. Anda dapat menggunakan serialisasi untuk memeriksa, mengonversi, dan menyerap data bersarang sebagai JSON dengan Redshift Spectrum. Metode ini didukung untuk format ORC, JSON, Ion, dan Parket. Gunakan parameter konfigurasi sesi `json_serialization_enable` untuk mengonfigurasi perilaku serialisasi. Saat diatur, tipe data JSON kompleks diserialisasikan ke VARCHAR (65535). JSON bersarang dapat diakses dengan. Fungsi JSON Untuk informasi selengkapnya, lihat json_serialization_enable.

Misalnya, tanpa pengaturan`json_serialization_enable`, kueri berikut yang mengakses kolom bersarang langsung gagal.

```
SELECT * FROM spectrum.customers LIMIT 1;

=> ERROR:  Nested tables do not support '*' in the SELECT clause.
```

```
SELECT name FROM spectrum.customers LIMIT 1;

=> ERROR:  column "name" does not exist in customers
```

Pengaturan `json_serialization_enable` memungkinkan kueri koleksi tingkat atas secara langsung.

```
SET json_serialization_enable TO true;

SELECT * FROM spectrum.customers order by id LIMIT 1;

id | name                                | phones         | orders
---+-------------------------------------+----------------
+-----------------------------------------------------------------------------------------------------------
1  | {"given": "John", "family": "Smith"} | ["123-457789"] | [{"shipdate":
 "2018-03-01T11:59:59.000Z", "price": 100.50}, {"shipdate": "2018-03-01T09:10:00.000Z",
 "price": 99.12}]

SELECT name FROM spectrum.customers order by id LIMIT 1;

name
---------
{"given": "John", "family": "Smith"}
```

Pertimbangkan item berikut saat membuat serial JSON bersarang.

- Ketika kolom koleksi diserialisasikan sebagai VARCHAR (65535), subbidang bersarangnya tidak dapat diakses secara langsung sebagai bagian dari sintaks kueri (misalnya, dalam klausa filter). Namun, fungsi JSON dapat digunakan untuk mengakses JSON bersarang.
- Representasi khusus berikut tidak didukung:
  - Serikat ORC
  - Peta ORC dengan tombol tipe kompleks
  - Datagram ion
  - Ion SEXP
- Stempel waktu dikembalikan sebagai string serial ISO.
- Kunci peta primitif dipromosikan ke string (misalnya, 1 ke"1").
- Top-level nilai null diserialisasikan sebagai NULLs.

- Jika serialisasi meluap ukuran VARCHAR maksimum 65535, sel diatur ke NULL.

Serialisasi tipe kompleks yang berisi string JSON

Secara default, nilai string yang terkandung dalam koleksi bersarang diserialisasikan sebagai string JSON yang lolos. Melarikan diri mungkin tidak diinginkan ketika string adalah JSON yang valid. Sebagai gantinya, Anda mungkin ingin menulis subelement bersarang atau bidang yang VARCHAR secara langsung sebagai JSON. Aktifkan perilaku ini dengan `json_serialization_parse_nested_strings` konfigurasi tingkat sesi. Ketika keduanya `json_serialization_enable` dan `json_serialization_parse_nested_strings` disetel, nilai JSON yang valid diserialkan sebaris tanpa karakter escape. Ketika nilainya tidak valid JSON, nilai ini diloloskan seolah-olah nilai `json_serialization_parse_nested_strings` konfigurasi tidak disetel. Untuk informasi selengkapnya, lihat json_serialization_parse_bersarang_string.

Misalnya, asumsikan data dari contoh sebelumnya berisi JSON sebagai tipe `structs` kompleks di bidang `name` VARCHAR (20):

```
name
---------
{"given": "{\"first\":\"John\",\"middle\":\"James\"}", "family": "Smith"}
```

Ketika `json_serialization_parse_nested_strings` diatur, `name` kolom diserialisasikan sebagai berikut:

```
SET json_serialization_enable TO true;
SET json_serialization_parse_nested_strings TO true;
SELECT name FROM spectrum.customers order by id LIMIT 1;

name
---------
{"given": {"first":"John","middle":"James"}, "family": "Smith"}
```

Alih-alih melarikan diri seperti ini:

```
SET json_serialization_enable TO true;
SELECT name FROM spectrum.customers order by id LIMIT 1;

name
---------
```

```
{"given": "{\"first\":\"John\",\"middle\":\"James\"}", "family": "Smith"}
```

HyperLogLog sketsa

Topik ini menjelaskan cara menggunakan HyperLogLog sketsa di Amazon Redshift. HyperLogLog adalah algoritma untuk masalah penghitungan yang berbeda, mendekati jumlah elemen yang berbeda dalam kumpulan data. HyperLogLog sketsa adalah array data keunikan tentang kumpulan data.

HyperLogLogadalah algoritma yang digunakan untuk memperkirakan kardinalitas multiset. Kardinalitas mengacu pada jumlah nilai yang berbeda dalam multiset. Misalnya, dalam himpunan {4,3,6,2,2,6,4,3,6,2,2,3}, kardinalitasnya adalah 4 dengan nilai berbeda 4, 3, 6, dan 2.

Ketepatan HyperLogLog algoritma (juga dikenal sebagai nilai m) dapat mempengaruhi keakuratan perkiraan kardinalitas. Selama estimasi kardinalitas, Amazon Redshift menggunakan nilai presisi default 15. Nilai ini bisa mencapai 26 untuk kumpulan data yang lebih kecil. Dengan demikian, kesalahan relatif rata-rata berkisar antara 0,01-0,6%.

Saat menghitung kardinalitas multiset, HyperLogLog algoritma menghasilkan konstruksi yang disebut sketsa HLL. Sketsa HLL merangkum informasi tentang nilai-nilai yang berbeda dalam multiset. Tipe data Amazon Redshift HLLSKETCH mewakili nilai sketsa tersebut. Tipe data ini dapat digunakan untuk menyimpan sketsa dalam tabel Amazon Redshift. Selain itu, Amazon Redshift mendukung operasi yang dapat diterapkan ke nilai HLLSKETCH sebagai fungsi agregat dan skalar. Anda dapat menggunakan fungsi-fungsi ini untuk mengekstrak kardinalitas HLLSKETCH dan menggabungkan beberapa nilai HLLSKETCH.

Tipe data HLLSKETCH menawarkan manfaat kinerja kueri yang signifikan saat mengekstraksi kardinalitas dari kumpulan data besar. Anda dapat melakukan praagregasi kumpulan data ini menggunakan nilai HLLSKETCH dan menyimpannya dalam tabel. Amazon Redshift dapat mengekstrak kardinalitas langsung dari nilai HLLSKETCH yang disimpan tanpa mengakses kumpulan data yang mendasarinya.

Saat memproses sketsa HLL, Amazon Redshift melakukan pengoptimalan yang meminimalkan jejak memori sketsa dan memaksimalkan ketepatan kardinalitas yang diekstraksi. Amazon Redshift menggunakan dua representasi untuk sketsa HLL, jarang dan padat. HLLSKETCH dimulai dalam format sparse. Saat nilai-nilai baru dimasukkan ke dalamnya, ukurannya meningkat. Setelah ukurannya mencapai ukuran representasi padat, Amazon Redshift secara otomatis mengubah sketsa dari jarang menjadi padat.

Amazon Redshift mengimpor, mengekspor, dan mencetak HLLSKETCH sebagai JSON saat sketsa dalam format jarang. Amazon Redshift mengimpor, mengekspor, dan mencetak HLLSKETCH

sebagai string Base64 saat sketsa dalam format padat. Untuk informasi selengkapnya tentang UNLOAD, lihatMembongkar tipe data HLLSKETCH. Untuk mengimpor teks atau data nilai yang dipisahkan koma (CSV) ke Amazon Redshift, gunakan perintah SALIN. Untuk informasi selengkapnya, lihat Memuat tipe data HLLSKETCH.

Untuk informasi tentang fungsi yang digunakan dengan HyperLogLog, lihatHyperLogLog fungsi.

Topik

- Pertimbangan-pertimbangan
- Batasan
- Contoh

Pertimbangan-pertimbangan

Topik ini menjelaskan detail penggunaan untuk HyperLogLog di Amazon Redshift.

Berikut ini adalah pertimbangan untuk digunakan HyperLogLog di Amazon Redshift:

- Non- HyperLogLog fungsi berikut dapat menerima input tipe HLLSKETCH atau kolom tipe HLLSKETCH:
  - Fungsi agregat COUNT
  - Ekspresi kondisional COALESCE dan NVL
  - Ekspresi CASE
- Pengkodean yang didukung adalah RAW.
- Anda dapat melakukan operasi UNLOAD pada tabel dengan kolom HLLSKETCH menjadi teks atau CSV. Anda dapat menggunakan kolom UNLOAD HLLSKETCH untuk menulis data HLLSKETCH. Amazon Redshift menampilkan data dalam format JSON untuk representasi jarang atau format Base64 untuk representasi padat. Untuk informasi selengkapnya tentang UNLOAD, lihatMembongkar tipe data HLLSKETCH.

  Berikut ini menunjukkan format yang digunakan untuk HyperLogLog sketsa jarang yang diwakili dalam format JSON.

```
{"version":1,"logm":15,"sparse":{"indices":
[15099259,33107846,37891580,50065963],"values":[2,3,2,1]}}
```

- Anda dapat mengimpor teks atau data CSV ke Amazon Redshift menggunakan perintah COPY. Untuk informasi selengkapnya, lihat Memuat tipe data HLLSKETCH.
- Pengkodean default untuk HLLSKETCH adalah RAW. Untuk informasi selengkapnya, lihat Pengkodean kompresi.

Batasan

Topik ini menjelaskan batasan untuk HyperLogLog di Amazon Redshift.

Berikut ini adalah batasan untuk digunakan HyperLogLog di Amazon Redshift:

- Tabel Amazon Redshift tidak mendukung kolom HLLSKETCH sebagai kunci pengurutan atau kunci distribusi untuk tabel Amazon Redshift.
- Amazon Redshift tidak mendukung kolom HLLSKETCH dalam klausa ORDER BY, GROUP BY, atau DISTINCT.
- Anda hanya dapat MEMBONGKAR kolom HLLSKETCH ke teks atau format CSV. Amazon Redshift kemudian menulis data HLLSKETCH baik dalam format JSON atau format Base64. Untuk informasi selengkapnya tentang UNLOAD, lihatMEMBONGKAR.
- Amazon Redshift hanya mendukung HyperLogLog sketsa dengan presisi (nilai logm) 15.
- Driver JDBC dan ODBC tidak mendukung tipe data HLLSKETCH. Oleh karena itu, set hasil menggunakan VARCHAR untuk mewakili nilai HLLSKETCH.
- Amazon Redshift Spectrum tidak mendukung data HLLSKETCH secara asli. Oleh karena itu, Anda tidak dapat membuat atau mengubah tabel eksternal dengan kolom HLLSKETCH.
- Tipe data untuk fungsi yang ditentukan pengguna Python (UDF) tidak mendukung tipe data HLLSKETCH. Untuk informasi selengkapnya tentang Python UDF, lihatUDF Python skalar.

> **Note**
>
> Mulai 1 November 2025, Amazon Redshift tidak akan lagi mendukung pembuatan UDF Python baru. UDF Python yang ada akan terus berfungsi hingga 30 Juni 2026. Mulai 1 Juli 2026, Amazon Redshift tidak akan lagi mendukung Python UDF. Kami menyarankan Anda memigrasikan UDF Python yang ada ke UDF Lambda sebelum 1 November 2025. Untuk informasi tentang membuat dan menggunakan Lambda UDF, lihat. UDF Lambda Skalar Untuk informasi tentang mengonversi UDF Python yang ada ke UDF Lambda, lihat posting

blog. https://aws.amazon.com/blogs/big-data/amazon-redshift-python-user-defined-functions-will-reach-end-of-support-after-june-30-2026/

Contoh

Bagian ini berisi contoh untuk digunakan HyperLogLog dengan Amazon Redshift.

Topik

- Contoh: Mengembalikan kardinalitas dalam subquery
- Contoh: Mengembalikan tipe HLLSKETCH dari sketsa gabungan dalam subquery
- Contoh: Mengembalikan HyperLogLog sketsa dari menggabungkan beberapa sketsa
- Contoh: Menghasilkan HyperLogLog sketsa di atas data S3 menggunakan tabel eksternal

Contoh: Mengembalikan kardinalitas dalam subquery

Contoh berikut mengembalikan kardinalitas untuk setiap sketsa dalam subquery untuk tabel bernama Sales.

```
CREATE TABLE Sales (customer VARCHAR, country VARCHAR, amount BIGINT);
INSERT INTO Sales VALUES ('David Joe', 'Greece', 14.5),  ('David Joe', 'Greece',
 19.95), ('John Doe', 'USA', 29.95), ('John Doe', 'USA', 19.95), ('George Spanos',
 'Greece', 9.95), ('George Spanos', 'Greece', 2.95);
```

Kueri berikut menghasilkan sketsa HLL untuk pelanggan dari setiap negara dan mengekstrak kardinalitas. Ini menunjukkan pelanggan unik dari setiap negara.

```
SELECT hll_cardinality(sketch), country
FROM (SELECT hll_create_sketch(customer) AS sketch, country
        FROM Sales
        GROUP BY country) AS hll_subquery;

hll_cardinality | country
----------------+---------
            1   | USA
            2   | Greece
 ...
```

Contoh: Mengembalikan tipe HLLSKETCH dari sketsa gabungan dalam subquery

Contoh berikut mengembalikan tipe HLLSKETCH tunggal yang mewakili kombinasi sketsa individu dari subquery. Sketsa digabungkan dengan menggunakan fungsi agregat HLL_COMBINE.

```
SELECT hll_combine(sketch)
FROM (SELECT hll_create_sketch(customer) AS sketch
        FROM Sales
        GROUP BY country) AS hll_subquery

                                        hll_combine
-----------------------------------------------------------------------------------------
 {"version":1,"logm":15,"sparse":{"indices":[29808639,35021072,47612452],"values":
[1,1,1]}}
(1 row)
```

Contoh: Mengembalikan HyperLogLog sketsa dari menggabungkan beberapa sketsa

Untuk contoh berikut, misalkan tabel `page-users` menyimpan sketsa praagregasi untuk setiap halaman yang dikunjungi pengguna di situs web tertentu. Setiap baris dalam tabel ini berisi HyperLogLog sketsa yang mewakili semua ID pengguna yang menunjukkan halaman yang dikunjungi.

```
page_users
-- +----------------+-------------+--------------+
-- | _PARTITIONTIME | page         | sketch |
-- +----------------+-------------+--------------+
-- | 2019-07-28     | homepage     | CHAQkAQYA... |
-- | 2019-07-28     | Product A    | CHAQxPnYB... |
-- +----------------+-------------+--------------+
```

Contoh berikut menggabungkan beberapa sketsa yang telah digabungkan sebelumnya dan menghasilkan satu sketsa. Sketsa ini merangkum kardinalitas kolektif yang dienkapsulasi setiap sketsa.

```
SELECT hll_combine(sketch) as sketch
```

```
FROM page_users
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
-- +----------------------------------------+
-- | sketch |
-- +----------------------------------------+
-- | CHAQ3sGoCxgCIAuCB4iAIBgTIBgqgIAgAwY.... |
-- +----------------------------------------+
```

Ketika sketsa baru dibuat, Anda dapat menggunakan fungsi HLL_CARDINALITY untuk mendapatkan nilai berbeda kolektif, seperti yang ditunjukkan berikut.

```
SELECT hll_cardinality(sketch)
FROM (
  SELECT
  hll_combine(sketch) as sketch
  FROM page_users
) AS hll_subquery
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
-- +-------+
-- | count |
-- +-------+
-- | 54356 |
-- +-------+
```

Contoh: Menghasilkan HyperLogLog sketsa di atas data S3 menggunakan tabel eksternal

Contoh berikut menyimpan HyperLogLog sketsa untuk menghindari akses langsung ke Amazon S3 untuk estimasi kardinalitas.

Anda dapat melakukan praagregasi dan menyimpan HyperLogLog sketsa dalam cache di tabel eksternal yang didefinisikan untuk menyimpan data Amazon S3. Dengan melakukan ini, Anda dapat mengekstrak perkiraan kardinalitas tanpa mengakses data dasar yang mendasarinya.

Misalnya, Anda telah membongkar sekumpulan file teks yang dibatasi tab ke Amazon S3. Anda menjalankan kueri berikut untuk menentukan tabel eksternal bernama `sales` dalam skema eksternal

Amazon Redshift bernama`spectrum`. Bucket Amazon S3 untuk contoh ini ada di AS Timur (Virginia Utara) Wilayah AWS.

```
create external table spectrum.sales(
salesid integer,
listid integer,
sellerid smallint,
buyerid smallint,
eventid integer,
dateid integer,
qtysold integer,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t' stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/';
```

Misalkan Anda ingin menghitung pembeli berbeda yang membeli barang pada tanggal yang berubah-ubah. Untuk melakukannya, contoh berikut menghasilkan sketsa untuk ID pembeli untuk setiap hari dalam setahun dan menyimpan hasilnya di tabel `hll_sales` Amazon Redshift.

```
CREATE TABLE hll_sales AS
SELECT saletime, hll_create_sketch(buyerid) AS sketch
FROM spectrum.sales
GROUP BY saletime;

SELECT TOP 5 * FROM hll_sales;
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
-- hll_sales

-- | saletime        | sketch
    |
-- +-----------------
+-------------------------------------------------------------------+
-- | 7/22/2008 8:30  | {"version":1,"logm":15,"sparse":{"indices":[9281416],"values":
[1]}}
-- | 2/19/2008 0:38  | {"version":1,"logm":15,"sparse":{"indices":[48735497],"values":
[3]}}
```

```
-- | 11/5/2008 4:49  | {"version":1,"logm":15,"sparse":{"indices":[27858661],"values":
[1]}}
-- | 10/27/2008 4:08 | {"version":1,"logm":15,"sparse":{"indices":[65295430],"values":
[2]}}
-- | 2/16/2008 9:37  | {"version":1,"logm":15,"sparse":{"indices":[56869618],"values":
[2]}}
-- +----------------
 +-----------------------------------------------------------------------+
```

Kueri berikut menunjukkan perkiraan jumlah pembeli berbeda yang membeli barang selama Jumat setelah Thanksgiving di tahun 2008.

```
SELECT hll_cardinality(hll_combine(sketch)) as distinct_buyers
FROM hll_sales
WHERE trunc(saletime) = '2008-11-28';
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
distinct_buyers
---------------
386
```

Misalkan Anda menginginkan jumlah pengguna berbeda yang membeli item pada rentang tanggal tertentu. Contohnya mungkin dari Jumat setelah Thanksgiving hingga Senin berikutnya. Untuk mendapatkan ini, query berikut menggunakan fungsi `hll_combine` agregat. Dengan fungsi ini, Anda dapat menghindari penghitungan ganda pembeli yang membeli item pada lebih dari satu hari dari kisaran yang dipilih.

```
SELECT hll_cardinality(hll_combine(sketch)) as distinct_buyers
FROM hll_sales
WHERE saletime BETWEEN '2008-11-28' AND '2008-12-01';
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
distinct_buyers
---------------
1166
```

Untuk menjaga `hll_sales` tabel tetap terkini, jalankan kueri berikut di akhir setiap hari. Melakukan hal ini menghasilkan HyperLogLog sketsa berdasarkan ID pembeli yang membeli item hari ini dan menambahkannya ke `hll_sales` tabel.

```
INSERT INTO hll_sales
SELECT saletime, hll_create_sketch(buyerid)
FROM spectrum.sales
WHERE TRUNC(saletime) = to_char(GETDATE(), 'YYYY-MM-DD')
GROUP BY saletime;
```

Cross-database pertanyaan

Topik ini menjelaskan kueri lintas database, yaitu kueri yang beroperasi pada beberapa database Amazon Redshift dalam satu cluster Amazon Redshift.

Dengan menggunakan kueri lintas database di Amazon Redshift, Anda dapat menanyakan dan menulis di seluruh database di cluster Amazon Redshift. Dengan kueri lintas database, Anda dapat menanyakan dan menulis data dari database apa pun di cluster Amazon Redshift, terlepas dari database mana yang terhubung ke Anda. Cross-database query menghilangkan salinan data dan menyederhanakan organisasi data Anda untuk mendukung beberapa grup bisnis dari gudang data yang sama.

Dengan kueri lintas database, Anda dapat melakukan hal berikut:

- Kueri data di seluruh database di cluster Amazon Redshift Anda.

  Anda tidak hanya dapat menanyakan dari database yang terhubung ke Anda, Anda juga dapat membaca dari database lain yang memiliki izin untuk Anda.

  Saat Anda menanyakan objek database pada database lain yang tidak terhubung, Anda memiliki akses baca hanya ke objek database tersebut. Anda dapat menggunakan kueri lintas database untuk mengakses data dari database mana pun di cluster Amazon Redshift Anda tanpa harus terhubung ke database tertentu. Melakukan hal ini dapat membantu Anda menanyakan dan menggabungkan data yang tersebar di beberapa database di cluster Amazon Redshift Anda dengan cepat dan mudah.

  Anda juga dapat menggabungkan kumpulan data dari beberapa database dalam satu kueri dan menganalisis data menggunakan intelijen bisnis (BI) atau alat analitik. Anda dapat terus menyiapkan kontrol akses baca tingkat tabel granular untuk pengguna dengan menggunakan perintah Amazon Redshift SQL standar. Dengan demikian, Anda dapat membantu memastikan bahwa pengguna hanya melihat subset data yang relevan yang memiliki izin untuk mereka.
- Tulis data di seluruh database di cluster Amazon Redshift Anda. .

  Anda dapat menulis dari database yang terhubung ke Anda, dan juga menulis dari database lain yang memiliki izin untuk Anda.

  Jika Anda memiliki izin menulis pada objek database di database lain yang tidak terhubung, Anda dapat menggunakan kueri lintas database untuk menulis data dari database apa pun di cluster

Amazon Redshift Anda tanpa harus terhubung ke database tertentu. Melakukan hal ini dapat membantu Anda dengan operasi penulisan yang kompleks sambil dengan cepat dan mudah menggabungkan data yang tersebar di beberapa database di cluster Amazon Redshift Anda.

Anda juga dapat menggabungkan kumpulan data dari beberapa database dalam satu kueri dan menulis data menggunakan extract-transform-load (ETL) atau alat analitik yang berbeda. Anda dapat terus menyiapkan kontrol akses tulis tingkat tabel granular untuk pengguna dengan menggunakan perintah Amazon Redshift SQL standar. Ini memastikan bahwa pengguna hanya melihat subset data yang relevan yang mereka miliki izin.

- Objek kueri.

  Anda dapat menanyakan objek database lain menggunakan nama objek yang sepenuhnya memenuhi syarat yang dinyatakan dengan notasi tiga bagian. Jalur lengkap ke objek database terdiri dari tiga komponen: nama database, skema, dan nama objek. Anda dapat mengakses objek apa pun dari database lain menggunakan notasi jalur lengkap,*database_name*.*schema_name*.*object_name*. Untuk mengakses kolom tertentu, gunakan*database_name*.*schema_name*.*object_name*.*column_name*.

  Anda juga dapat membuat alias untuk skema di database lain menggunakan notasi skema eksternal. Skema eksternal ini merujuk ke database lain dan pasangan skema. Query dapat mengakses objek database lain menggunakan notasi skema eksternal,*external_schema_name*.*object_name*.

  Dalam kueri read-only yang sama, Anda dapat menanyakan berbagai objek database, seperti tabel pengguna, tampilan reguler, tampilan terwujud, dan tampilan pengikatan terlambat dari database lain.

- Kelola izin.

  Pengguna dengan hak akses untuk objek di database apa pun di cluster Amazon Redshift dapat menanyakan dan menulis data ke objek tersebut. Anda memberikan hak istimewa kepada pengguna dan grup pengguna menggunakan HIBAH perintah. Anda juga dapat mencabut hak istimewa menggunakan MENCABUT perintah ketika pengguna tidak lagi memerlukan akses ke objek database tertentu.

- Bekerja dengan metadata dan alat BI.

  Anda dapat membuat skema eksternal untuk merujuk ke skema di database Amazon Redshift lain dalam cluster Amazon Redshift yang sama. Untuk informasi, lihat BUAT SKEMA EKSTERNAL perintah.

Setelah referensi skema eksternal dibuat, Amazon Redshift menunjukkan tabel di bawah skema database lain di SVV_EXTERNAL_TABEL dan SVV_KOLON_EKSTERNAL_ untuk alat untuk menjelajahi metadata.

Untuk mengintegrasikan kueri lintas database dengan alat BI, Anda dapat menggunakan tampilan sistem berikut. Ini membantu Anda melihat informasi tentang metadata objek di database yang terhubung dan lainnya di cluster Amazon Redshift.

Berikut ini adalah tampilan sistem yang menampilkan semua objek Amazon Redshift dan objek eksternal dari semua database di cluster Amazon Redshift Anda:

- SVV_ALL_KOLOM
- SVV_ALL_SCHEMAS
- SVV_ALL_TABEL

Berikut ini adalah tampilan sistem yang menampilkan semua objek Amazon Redshift dari semua database di cluster Amazon Redshift Anda:

- SVV_REDSHIFT_KOLOM
- SVV_REDSHIFT_DATABASE
- SVV_REDSHIFT_FUNCTION
- SVV_REDSHIFT_SKEMA
- SVV_REDSHIFT_TABEL

Topik

- Pertimbangan-pertimbangan
- Batasan
- Cross-database contoh kueri
- Menggunakan kueri lintas database dengan editor kueri

Pertimbangan-pertimbangan

Topik ini menjelaskan detail penggunaan untuk kueri lintas database di Amazon Redshift.

Saat Anda bekerja dengan fitur kueri lintas database di Amazon Redshift, pertimbangkan hal berikut:

- Amazon Redshift mendukung kueri lintas database pada semua jenis node RG dan RA3 serta ruang nama tanpa server.
- Amazon Redshift mendukung penggabungan data dari tabel atau tampilan di satu atau lebih database dalam cluster Amazon Redshift yang sama.
- Semua kueri dalam transaksi pada database yang terhubung membaca data dalam keadaan yang sama dari database lain seperti data pada awal transaksi. Pendekatan ini membantu memberikan konsistensi transaksional kueri di seluruh database. Amazon Redshift mendukung konsistensi transaksional untuk kueri lintas database.
- Untuk mendapatkan metadata di seluruh database, gunakan tampilan metadata SVV_ALL* dan SVV_REDSHIFT*. Anda tidak dapat menggunakan notasi tiga bagian atau skema eksternal untuk menanyakan tabel metadata lintas-database atau tampilan di bawah information_schema dan pg_catalog.

Batasan

Topik ini menjelaskan batasan untuk kueri lintas database di Amazon Redshift.

Saat Anda bekerja dengan fitur kueri lintas-database di Amazon Redshift, perhatikan batasan berikut:

- Anda tidak dapat menanyakan tampilan yang dibuat pada database lain yang merujuk ke objek database lain.
- Anda tidak dapat menulis ke skema eksternal ke objek database pada database lain yang tidak terhubung.
- Anda hanya dapat membuat tampilan pengikatan terlambat dan terwujud pada objek database lain di cluster. Anda tidak dapat membuat tampilan reguler pada objek database lain di cluster.
- Amazon Redshift tidak mendukung tabel dengan hak istimewa tingkat kolom untuk kueri lintas database.
- Menjalankan kueri lintas database pada tabel dengan kunci pengurutan yang disisipkan tidak didukung.

Cross-database contoh kueri

Topik ini berisi contoh cara menggunakan kueri lintas database. Cross-database query adalah kueri yang beroperasi pada beberapa database dalam satu cluster Amazon Redshift.

Gunakan contoh berikut untuk membantu mempelajari cara menyiapkan kueri lintas-database yang mereferensikan database Amazon Redshift.

Untuk memulai, buat database db1 db2 dan pengguna user1 dan user2 di cluster Amazon Redshift Anda. Untuk informasi selengkapnya, lihat BUAT BASIS DATA dan BUAT PENGGUNA.

```
--As user1 on db1
CREATE DATABASE db1;

CREATE DATABASE db2;

CREATE USER user1 PASSWORD 'Redshift01';

CREATE USER user2 PASSWORD 'Redshift01';
```

Seperti user1 halnyadb1, buat tabel, berikan hak akses keuser2, dan masukkan nilai ketable1. Untuk informasi selengkapnya, lihat HIBAH dan INSERT.

```
--As user1 on db1
CREATE TABLE table1 (c1 int, c2 int, c3 int);

GRANT SELECT ON table1 TO user2;

INSERT INTO table1 VALUES (1,2,3),(4,5,6),(7,8,9);
```

Seperti user2 halnyadb2, jalankan query lintas-database db2 menggunakan notasi tiga bagian.

```
--As user2 on db2
SELECT * from db1.public.table1 ORDER BY c1;

c1 | c2  | c3
---+-----+----
1  |  2  | 3
4  |  5  | 6
7  |  8  | 9
(3 rows)
```

Sekarang berikan hak menulis keuser2, dan masukkan nilai ke table1 dalam db1 asuser2.

```
--As user1 on db1
```

```
GRANT INSERT ON table1 TO user2;
```

Seperti `user2` halnyadb2, jalankan kueri lintas database dengan db2 menggunakan notasi tiga bagian untuk memasukkan data ke dalam`table1`. db1

```
--As user2 on db2
INSERT INTO db1.public.table1 VALUES (10,11,12);
SELECT * from db1.public.table1 ORDER BY c1;

c1  | c2   | c3
----+------+----
1   |  2   | 3
4   |  5   | 6
7   |  8   | 9
10  |  11  | 12
(4 rows)
```

Seperti `user2` halnyadb2, buat skema eksternal dan jalankan kueri lintas database db2 menggunakan notasi skema eksternal.

```
--As user2 on db2
CREATE EXTERNAL SCHEMA db1_public_sch
FROM REDSHIFT DATABASE 'db1' SCHEMA 'public';

SELECT * FROM db1_public_sch.table1 ORDER BY c1;

c1  | c2 | c3
----+----+----
1   | 2  | 3
4   | 5  | 6
7   | 8  | 9
10  | 11 | 12
(4 rows)
```

Untuk membuat tampilan yang berbeda dan memberikan izin ke tampilan tersebut, seperti `user1` padadb1, lakukan hal berikut.

```
--As user1 on db1
CREATE VIEW regular_view AS SELECT c1 FROM table1;

GRANT SELECT ON regular_view TO user2;
```

```
CREATE MATERIALIZED VIEW mat_view AS SELECT c2 FROM table1;

GRANT SELECT ON mat_view TO user2;


CREATE VIEW late_bind_view AS SELECT c3 FROM public.table1 WITH NO SCHEMA BINDING;

GRANT SELECT ON late_bind_view TO user2;
```

Seperti `user2` halnyadb2, jalankan query lintas-database berikut menggunakan notasi tiga bagian untuk melihat tampilan tertentu.

```
--As user2 on db2
SELECT * FROM db1.public.regular_view;

c1
----
1
4
7
10
(4 rows)

SELECT * FROM db1.public.mat_view;

c2
----
2
5
8
11
(4 rows)

SELECT * FROM db1.public.late_bind_view;

c3
----
3
6
9
12
(4 rows)
```

Seperti `user2` halnyadb2, jalankan kueri lintas-database berikut menggunakan notasi skema eksternal untuk menanyakan tampilan pengikatan terlambat.

```
--As user2 on db2
SELECT * FROM db1_public_sch.late_bind_view;

c3
----
3
6
9
12
(4 rows)
```

Seperti `user2` halnyadb2, jalankan perintah berikut menggunakan tabel yang terhubung dalam satu kueri.

```
--As user2 on db2
CREATE TABLE table1 (a int, b int, c int);

INSERT INTO table1 VALUES (1,2,3), (4,5,6), (7,8,9);

SELECT a AS col_1, (db1.public.table1.c2 + b) AS sum_col2, (db1.public.table1.c3 + c)
 AS sum_col3 FROM db1.public.table1, table1 WHERE db1.public.table1.c1 = a;
col_1 | sum_col2 | sum_col3
------+----------+----------
1     | 4        | 6
4     | 10       | 12
7     | 16       | 18
(3 rows)
```

Contoh berikut mencantumkan semua database pada cluster.

```
select database_name, database_owner, database_type
from svv_redshift_databases
where database_name in ('db1', 'db2');

 database_name | database_owner | database_type
---------------+----------------+---------------
 db1           |            100 | local
 db2           |            100 | local
(2 rows)
```

Contoh berikut mencantumkan semua skema Amazon Redshift dari semua database di cluster.

```
select database_name, schema_name, schema_owner, schema_type
from svv_redshift_schemas
where database_name in ('db1', 'db2');

 database_name |    schema_name     | schema_owner | schema_type
---------------+--------------------+--------------+-------------
 db1           | pg_catalog         |            1 | local
 db1           | public             |            1 | local
 db1           | information_schema |            1 | local
 db2           | pg_catalog         |            1 | local
 db2           | public             |            1 | local
 db2           | information_schema |            1 | local
(6 rows)
```

Contoh berikut mencantumkan semua tabel Amazon Redshift atau tampilan semua database di cluster.

```
select database_name, schema_name, table_name, table_type
from svv_redshift_tables
where database_name in ('db1', 'db2') and schema_name in ('public');

 database_name | schema_name |     table_name      | table_type
---------------+-------------+---------------------+------------
 db1           | public      | late_bind_view      | VIEW
 db1           | public      | mat_view            | VIEW
 db1           | public      | mv_tbl__mat_view__0 | TABLE
 db1           | public      | regular_view        | VIEW
 db1           | public      | table1              | TABLE
 db2           | public      | table2              | TABLE
(6 rows)
```

Contoh berikut mencantumkan semua Amazon Redshift dan skema eksternal dari semua database di cluster.

```
select database_name, schema_name, schema_owner, schema_type
from svv_all_schemas where database_name in ('db1', 'db2') ;

 database_name |    schema_name     | schema_owner | schema_type
---------------+--------------------+--------------+-------------
 db1           | pg_catalog         |            1 | local
```

```
 db1           | public             |           1 | local
 db1           | information_schema |           1 | local
 db2           | pg_catalog         |           1 | local
 db2           | public             |           1 | local
 db2           | information_schema |           1 | local
 db2           | db1_public_sch     |           1 | external
(7 rows)
```

Contoh berikut mencantumkan semua Amazon Redshift dan tabel eksternal dari semua database di cluster.

```
select database_name, schema_name, table_name, table_type
from svv_all_tables
where database_name in ('db1', 'db2') and schema_name in ('public');

 database_name | schema_name |     table_name      | table_type
---------------+-------------+---------------------+------------
 db1           | public      | regular_view        | VIEW
 db1           | public      | mv_tbl__mat_view__0 | TABLE
 db1           | public      | mat_view            | VIEW
 db1           | public      | late_bind_view      | VIEW
 db1           | public      | table1              | TABLE
 db2           | public      | table2              | TABLE
(6 rows)
```

Menggunakan kueri lintas database dengan editor kueri

Topik ini menjelaskan cara menggunakan kueri lintas database dengan editor kueri. Cross-databasequery adalah kueri yang beroperasi pada beberapa database dalam satu cluster Amazon Redshift.

Anda dapat menggunakan kueri lintas database untuk mengakses data dari database mana pun di cluster Amazon Redshift Anda tanpa harus terhubung ke database tertentu. Ketika Anda menjalankan kueri lintas database pada database lain yang tidak terhubung, Anda hanya memiliki akses baca dan tulis ke objek database tersebut.

Anda dapat menanyakan dan menulis ke objek database lain menggunakan nama objek yang sepenuhnya memenuhi syarat yang dinyatakan dengan notasi tiga bagian. Jalur lengkap ke objek database terdiri dari tiga komponen: nama database, skema, dan nama objek. Contohnya adalah *database_name.schema_name.object_name*.

Untuk menggunakan kueri lintas database dengan editor kueri v2

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Buat cluster untuk menggunakan kueri lintas database di editor kueri Amazon Redshift v2. Untuk informasi selengkapnya, lihat Membuat cluster di Panduan Manajemen Amazon Redshift.
3. Aktifkan akses ke editor kueri dengan izin yang sesuai. Untuk informasi selengkapnya, lihat Mengkueri database menggunakan editor kueri v2 di Panduan Manajemen Amazon Redshift.
4. Pada menu navigasi, pilih Query editor v2, lalu sambungkan ke database di cluster Anda.

   Saat Anda terhubung ke editor kueri v2 untuk pertama kalinya, Amazon Redshift menunjukkan sumber daya untuk database yang terhubung secara default.
5. Pilih database lain yang Anda miliki akses untuk melihat objek database untuk database lain ini. Untuk melihat objek, pastikan Anda memiliki izin yang sesuai. Setelah Anda memilih database, Amazon Redshift menampilkan daftar skema dari database.

   Pilih skema untuk melihat daftar objek database dalam skema itu.

   > **Note**
   >
   > Amazon Redshift tidak secara langsung mendukung objek katalog kueri yang merupakan bagian dari AWS Glue atau database federasi. Untuk menanyakan ini, pertama-tama buat skema eksternal yang merujuk ke sumber data eksternal di setiap database.
   > Kueri lintas database Amazon Redshift dengan notasi tiga bagian tidak mendukung tabel metadata di bawah skema `information_schema` dan `pg_catalog` karena tampilan metadata ini khusus untuk database.
6. (Opsional) Memfilter daftar tabel atau tampilan untuk skema yang Anda pilih.

Kompatibilitas Apache Iceberg untuk Amazon Redshift

Anda dapat mendaftarkan seluruh cluster atau ruang nama tanpa server yang disediakan Amazon Redshift ke untuk membuat katalog yang berbagi AWS Glue Data Catalog data langsung dengan aman di seluruh akun. AWS Anda dapat mengakses katalog ini dari mesin query SQL apa pun yang mendukung Apache Iceberg REST API. AWS Lake Formation mengelola izin untuk katalog, memungkinkan Anda mengelola satu salinan data dengan satu set izin sambil memanfaatkan fitur Amazon Redshift seperti tampilan terwujud dan integrasi Zero-ETL.

Semua katalog yang dibuat dari cluster yang disediakan Amazon Redshift terdaftar dan ruang nama tanpa server di dalamnya secara otomatis AWS Glue Data Catalog dipasang sebagai database eksternal pada semua cluster yang disediakan dan grup kerja tanpa server di akun yang sama di bawah akun yang sama. Wilayah AWS Katalog yang dibuat AWS Glue Data Catalog untuk menyimpan data di Redshift Managed Storage (RMS) dipasang dengan cara yang sama sebagai database eksternal. Setelah database dipasang, Anda dapat langsung terhubung ke database ini dan menanyakan objek menggunakan not `database@namespace-catalog.schema.table` asi tiga bagian.

Wilayah di mana kompatibilitas Apache Iceberg tersedia

Kompatibilitas Apache Iceberg dengan Amazon Redshift tersedia sebagai berikut: Wilayah AWS

- Timur AS (N. Virginia)
- AS Timur (Ohio)
- AS Barat (California Utara)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Seoul);
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Europe (London)

- Europe (Stockholm)
- Amerika Selatan (Sao Paulo)

Pertimbangan dan batasan saat menggunakan katalog Amazon Redshift di AWS Glue Data Catalog

Saat menggunakan katalog Amazon Redshift di AWS Glue Data Catalog, pertimbangkan hal berikut:

- Gudang data terdaftar AWS Glue Data Catalog mengikuti sintaks tiga bagian untuk mengakses tabel (`database@namespace-catalog.schema.table`). Misalnya, jika Anda mendaftarkan namespace Amazon Redshift bernama a, diisi oleh database bernama b, yang menyimpan skema bernama c, di mana tabel bernama d, Anda akan memilih dari d menggunakan pernyataan berikut:

  ```
  SELECT * FROM b@a.c.d;
  ```

  Perhatikan bahwa panjang gabungan dari `database@namespace-catalog` bagian sintaks harus 127 karakter atau kurang.
- Saat Anda mendaftarkan cluster atau namespace ke AWS Glue Data Catalog, Amazon Redshift mendaftarkan semua database dan relasi di cluster atau namespace tersebut.
- Anda dapat mendaftarkan beberapa cluster Redshift dan ruang nama ke. AWS Glue Data Catalog
- Mendaftarkan cluster atau namespace hanya mendaftarkan skema internal dan relasi di cluster atau namespace itu. Berikut ini tidak terdaftar:
  - Skema eksternal.
  - Tabel eksternal. Perhatikan bahwa tampilan pengikatan terlambat yang dibuat dari tabel eksternal akan didaftarkan.
  - User-created fungsi.
  - Prosedur.
  - Tabel dengan keamanan tingkat baris atau kebijakan penyembunyian data dinamis terlampir.
  - Objek database dengan nama huruf besar atau huruf campuran. Tabel yang berisi kolom huruf besar atau huruf campuran tidak terdaftar. Ini berlaku bahkan ketika din enable_case_sensitive_identifier onaktifkan.
- Izin database Amazon Redshift, seperti peran yang diberikan oleh kontrol akses berbasis peran, tidak ditransfer ke katalog di. AWS Glue Data Catalog Gunakan AWS Lake Formation untuk mengkonfigurasi izin untuk AWS Glue Data Catalog. Untuk informasi selengkapnya tentang

menggunakan Lake Formation untuk mengonfigurasi izin, lihat Menyiapkan izin untuk berbagi data Amazon Redshift di Panduan Peng AWS Lake Formation embang.

- Saat Anda membuat katalog dari cluster terdaftar atau namespace tanpa server, Anda akan membuat grup kerja terkelola Amazon Redshift menggunakan sumber daya komputasi Amazon Redshift untuk menangani persyaratan komputasi saat menanyakan katalog tersebut. AWS Glue Data Catalog Anda dapat melihat grup kerja terkelola di konsol Amazon Redshift Serverless dan mengelolanya di. AWS Glue
- Saat Anda mendaftarkan cluster yang dijeda AWS Glue Data Catalog , cluster tersebut tidak akan di-mount sebagai katalog sampai cluster dilanjutkan.
- Saat Anda mendaftarkan namespace tanpa server yang tidak digunakan secara aktif, namespace AWS Glue Data Catalog itu tidak akan di-mount sebagai katalog sampai namespace digunakan lagi.
- Untuk mengakses tabel di gudang data terdaftar ke AWS Glue Data Catalog, tingkat isolasi database harus SNAPSHOT. Mencoba mengakses tabel dengan tingkat isolasi SERIALIZABLE akan mengakibatkan kesalahan. Untuk informasi selengkapnya tentang isolasi yang dapat diserialisasi, lihat. Tingkat isolasi di Amazon Redshift Untuk informasi tentang mengubah tingkat isolasi database, lihatALTER DATABASE.

  Perhatikan bahwa tingkat isolasi `dev` database default tidak dapat diubah. Akibatnya, ini berarti bahwa tabel dalam `dev` database tidak akan dimasukkan dalam data yang dapat diakses saat mendaftarkan gudang data ke AWS Glue Data Catalog.

Persyaratan kebijakan IAM untuk mengakses cluster dan ruang nama yang terdaftar di Katalog Data

Topik ini menjelaskan izin IAM yang diperlukan untuk mendaftarkan cluster yang disediakan dan ruang nama tanpa server ke Katalog Data dan mengaksesnya dengan Amazon Redshift.

Setelah Anda mendaftarkan cluster yang disediakan atau namespace tanpa server ke AWS Glue Data Catalog, Anda memerlukan izin berikut untuk menemukan pembuatan dan perubahan pada katalog yang dibuat selanjutnya.

- `glue:GetCatalog`
- `glue:GetCatalogs`
- `lakeformation:GetDataAccess`

Izin ini disertakan dalam peran terkait layanan. `AmazonRedshiftServiceLinkedRolePolicy` Untuk informasi selengkapnya tentang peran ini, lihat Menggunakan peran terkait layanan untuk Amazon Redshift di Panduan Manajemen Amazon Redshift.

Mendaftarkan cluster Amazon Redshift dan ruang nama ke AWS Glue Data Catalog

Anda dapat menambahkan cluster yang disediakan Amazon Redshift dan ruang nama tanpa server ke untuk mengaksesnya menggunakan API REST AWS Glue Data Catalog Apache Iceberg. Anda melakukan ini dengan mendaftarkan gudang data Amazon Redshift ke AWS Glue Data Catalog menggunakan konsol Amazon Redshift atau AWS CLI, kemudian membuat katalog federasi Amazon Redshift untuk gudang yang digunakan. AWS Lake Formation

Gudang data Amazon Redshift yang terdaftar di Katalog Data bertindak sebagai pembagian data produsen. Perubahan yang Anda buat pada cluster atau ruang nama tanpa server dari katalognya tercermin dalam cluster atau namespace di Redshift, dan sebaliknya.

Registering using the Amazon Redshift console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Arahkan ke namespace tanpa server atau cluster yang disediakan yang ingin Anda daftarkan dan pilih.
3. Dari halaman detail namespace atau cluster, pilih Daftar ke AWS Glue Data Catalog dari menu tarik-turun T ind akan.
4. Masukkan ID akun tujuan tempat Anda ingin mendaftarkan namespace atau cluster dan pilih Daftar.
5. Jika Anda mendaftar dari akun Anda ke akun yang sama di AWS Glue, Anda akan dibawa langsung ke AWS Lake Formation konsol untuk menyelesaikan prosesnya. Jika Anda mendaftar ke akun lain, tautan muncul yang membawa Anda ke Formasi Danau.

Registering using the AWS CLI

Untuk mendaftarkan cluster atau namespace AWS Glue Data Catalog menggunakan AWS CLI, gunakan `register-namespace` perintah dengan opsi berikut:

- `namespace-identifier`: Objek dengan pengenal unik cluster atau namespace yang Anda daftarkan. Objek ini berbeda tergantung pada apakah Anda mendaftarkan cluster yang disediakan atau namespace tanpa server. Pertimbangkan hal berikut:
  - Untuk cluster yang disediakan, Anda menyediakan `ProvisionedIdentifier` objek, yang berisi `ClusterIdentifier` objek dengan pengenal unik cluster yang Anda daftarkan.
  - atau ruang nama tanpa server, Anda menyediakan `ServerlessIdentifier` objek, yang berisi `NamespaceIdentifier` objek dengan pengenal unik dari namespace yang Anda daftarkan, serta `WorkgroupIdentifier` objek dengan pengidentifikasi unik dari kelompok kerja yang terkait dengan namespace itu.
- `consumer-identifiers`: Array dengan elemen tunggal, berisi pengidentifikasi unik dari akun tempat Anda mendaftarkan cluster atau namespace.

Contoh berikut mendaftarkan namespace `mySampleNamespace` tanpa server ke ID akun. `012345678910`

```
aws redshift register-namespace /
--namespace-identifier {ServerlessIdentifier: {NamespaceIdentifer:
 mySampleNamespace, WorkgroupIdentifier: mySampleWorkgroup}} /
--consumer-identifiers [012345678910]
```

Menghapus pendaftaran cluster dan ruang nama Amazon Redshift dari AWS Glue Data Catalog

Anda dapat membatalkan pendaftaran cluster yang disediakan atau namespace tanpa server dari AWS Glue Data Catalog menggunakan konsol Amazon Redshift atau. AWS CLI Gudang data yang Anda deregistrasi dari hanya AWS Glue Data Catalog dihapus dari itu sendiri. AWS Glue Data Catalog Gudang tetap menjadi bagian dari akun Amazon Redshift Anda. Selain itu, katalog tetap masuk AWS Glue dan harus dihapus secara manual. Saat Anda menghapus katalog dari AWS Glue, kelompok kerja terkelola yang terkait dengannya juga dihapus.

Deregistering using the Amazon Redshift console

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.

2. Arahkan ke namespace tanpa server atau cluster yang disediakan yang ingin Anda deregister dan pilih.
3. Dari halaman detail namespace atau cluster, pilih Deregister AWS Glue Data Catalog dari menu drop-down Ac tions. Opsi ini hanya muncul jika Anda telah memilih gudang data yang sudah terdaftar ke AWS Glue Data Catalog.
4. Masukkan ID akun yang memiliki katalog yang menyimpan cluster atau namespace Anda dan pilih Der egister.

Deregistering using the AWS CLI

Untuk membatalkan pendaftaran cluster atau namespace dari AWS Glue Data Catalog menggunakan AWS CLI, gunakan `deregister-namespace` perintah dengan opsi berikut:

- `namespace-identifier`: Objek dengan pengenal unik cluster atau namespace yang Anda deregistrasi. Objek ini berbeda tergantung pada apakah Anda membatalkan pendaftaran cluster yang disediakan atau namespace tanpa server. Pertimbangkan hal berikut:
  - Untuk cluster yang disediakan, Anda menyediakan `ProvisionedIdentifier` objek, yang berisi `ClusterIdentifier` objek dengan pengidentifikasi unik cluster yang Anda deregistrasi.
  - atau ruang nama tanpa server, Anda menyediakan `ServerlessIdentifier` objek, yang berisi `NamespaceIdentifier` objek dengan pengidentifikasi unik dari namespace yang Anda deregistrasi, serta `WorkgroupIdentifier` objek dengan pengidentifikasi unik dari kelompok kerja yang terkait dengan namespace itu.
- `consumer-identifiers`: Array dengan elemen tunggal, berisi pengidentifikasi unik dari akun tempat Anda membatalkan pendaftaran cluster atau namespace.

Contoh berikut membatalkan pendaftaran namespace `mySampleNamespace` tanpa server dari ID akun. `012345678910`

```
aws redshift deregister-namespace /
--namespace-identifier {ServerlessIdentifier: {NamespaceIdentifer:
 mySampleNamespace, WorkgroupIdentifier: mySampleWorkgroup}} /
--consumer-identifiers [012345678910]
```

Kelompok kerja yang dikelola

Saat Anda mendaftarkan cluster yang disediakan atau namespace tanpa server ke AWS Glue Data Catalog dan membuat katalog darinya, AWS Glue membuat grup kerja terkelola untuk menyediakan sumber daya komputasi untuk mesin kueri SQL apa pun yang mengakses katalog tersebut. Misalnya, Amazon Athena pengguna yang menanyakan tabel dalam katalog akan memenuhi persyaratan komputasinya oleh sumber daya komputasi Amazon Redshift kelompok kerja yang dikelola. Kelompok kerja terkelola menggunakan sumber daya komputasi Amazon Redshift, tetapi dikelola di. AWS Glue

Kelompok kerja terkelola dihitung dalam kuota grup kerja Amazon Redshift Serverless, yang secara default adalah 25. Untuk informasi selengkapnya, lihat Kuota dan batasan di Amazon Redshift dalam Panduan Manajemen Amazon Redshift.

Anda dapat melihat informasi umum, metrik kinerja, dan informasi kueri untuk grup kerja terkelola di konsol Amazon Redshift Serverless. Untuk informasi selengkapnya, lihat Melihat properti untuk grup kerja di Panduan Manajemen Amazon Redshift.

Mengkueri katalog yang terdaftar di AWS Glue Data Catalog

Setelah Anda mendaftarkan gudang data Amazon Redshift ke AWS Glue Data Catalog dan menetapkan izin untuk katalog yang dihasilkan di AWS Lake Formation, katalog secara otomatis dipasang di semua instans Amazon Redshift dengan akses ke gudang data sumber di akun yang sama dan. Wilayah AWS Anda kemudian dapat menanyakan katalog itu seperti yang Anda lakukan pada cluster lokal atau grup kerja. Anda juga dapat menanyakan katalog yang terdaftar AWS Glue Data Catalog menggunakan mesin SQL apa pun yang mendukung Apache Iceberg REST Open API. Untuk informasi selengkapnya tentang membuat kueri katalog dalam AWS Glue Data Catalog menggunakan Apache Iceberg REST API, lihat Mengakses Katalog Data di Panduan Peng AWS Glue embang. Untuk informasi tentang Apache Iceberg REST API, lihat spesifikasi Apache Iceberg REST Open API.

Untuk menanyakan katalog, Anda harus terlebih dahulu mengatur izin untuk katalog menggunakan AWS Lake Formation. Untuk informasi selengkapnya tentang menyetel izin untuk katalog di AWS Lake Formation, lihat Menyiapkan izin untuk berbagi data Amazon Redshift di Panduan Peng AWS Lake Formation embang. Anda juga memerlukan peran IAM dengan kebijakan terkel `AmazonRedshiftServiceLinkedRolePolicy` ola terlampir. Untuk informasi tentang peran terkait layanan, lihat Menggunakan peran terkait layanan untuk Amazon Redshift di Panduan Manajemen Amazon Redshift.

Perhatikan bahwa kueri terhadap katalog harus mengikuti sintaks tiga bagian berikut untuk mengakses tabel:

```
database@namespace.schema.table
```

Untuk informasi umum tentang menanyakan gudang data Amazon Redshift, lihat Ku eri database di Panduan Manajemen Amazon Redshift.

Querying using the query editor v2

Setelah menetapkan izin untuk akun untuk mengakses grup kerja terkelola, kelompok kerja terkelola tersebut muncul di panel tampilan pohon di bawah bagian database eksternal dari database tanpa server Anda. Anda dapat menanyakan grup kerja terkelola dengan cara yang sama seperti menanyakan klaster internal yang disediakan Amazon Redshift atau kelompok kerja tanpa server, menggunakan format sintaks tiga bagian. `database@namespace/cluster.schema.table` Lihat contoh pernyataan berikut:

```
SELECT price FROM sales_db@mynamespace.sales_schema.inventory_table
```

Querying using the Data API

Anda dapat menanyakan grup kerja terkelola menggunakan Amazon Redshift Data API dengan cara yang sama seperti menanyakan klaster internal yang disediakan Amazon Redshift atau grup kerja tanpa server, meneruskan Nama Sumber Daya Amazon (ARN) katalog ke atribut yang relevan. `database` Pertimbangkan contoh berikut yang membuat tabel dalam katalog.

```
aws redshift-data execute-statement –sql 'CREATE TABLE IF NOT EXISTS "dev@test-
rms-catalog".public.t1 (c1 INT, c2 VARCHAR(10));' –database arn:aws:glue:us-
east-1:550022730026:catalog/test-rms-catalog
```

Melakukan kueri AWS Glue Katalog federasi IRC dengan Amazon Redshift

Dengan AWS Glue Iceberg REST Catalog (IRC), Anda dapat menyatukan katalog data eksternal, seperti Databricks Unity Catalog, ke dalam. AWS Glue Data Catalog Setelah katalog disatukan, Amazon Redshift dapat menanyakan tabel eksternal ini secara langsung tanpa memerlukan pergerakan data. Bab ini menjelaskan cara mengakses tabel Databricks Unity Catalog menggunakan federasi dari Amazon Redshift.

Gambaran umum arsitektur

Arsitektur melibatkan tiga komponen yang mungkin berada di akun dan Wilayah yang berbeda:

- Sumber data (bucket Amazon S3) — Bucket Amazon S3 dikelola oleh platform data eksternal (misalnya, Databricks). Di sinilah file data tabel Iceberg yang sebenarnya berada. Untuk membaca tabel Delta, pastikan metadata Iceberg tersedia untuk tabel ini menggunakan UniForm.
- AWS Glue Katalog federasi IRC — Entri Katalog Data yang menyatukan katalog eksternal (misalnya, Databricks Unity Catalog) dan mengekspos metadatanya melalui protokol Iceberg REST.
- Amazon Redshift compute — Cluster yang disediakan atau kelompok kerja tanpa server yang mengeksekusi kueri terhadap tabel federasi.

Anda dapat mengakses katalog federasi dari Amazon Redshift dengan cara berikut:

- Automount — Amazon Redshift secara otomatis memasang `awsdatacatalog` database, yang menyediakan akses ke semua database yang terdaftar di Katalog Data untuk akun dan Wilayah yang sama. Tidak diperlukan pembuatan skema. Kueri menggunakan notasi tiga bagian:`"awsdatacatalog"."database_name"."table_name"`.
- Skema eksternal — Anda secara eksplisit membuat skema eksternal menggunakan skema `CREATE EXTERNAL SCHEMA` yang menunjuk ke database tautan sumber daya di Katalog Data, menentukan peran IAM statis atau `SESSION` untuk otorisasi identitas federasi.

> **Important**
>
> Skema eksternal hanya berfungsi jika semua sumber daya (katalog, komputasi, dan bucket Amazon S3) berada di Wilayah yang sama.

Menyiapkan dan menanyakan katalog federasi di akun yang sama

Menyiapkan federasi katalog

Ikuti petunjuk di Katalog Federate to Databricks Unity di Panduan Peng AWS Lake Formation embang.

Membuat tautan sumber daya di Katalog Data

Masuk sebagai admin danau data. Buat tautan sumber daya database di Katalog Data yang menunjuk ke database federasi.

Memberikan izin untuk mengakses tabel di Amazon Redshift

Sebagai admin danau data, berikan izin berikut ke prinsipal IAM atau peran IAM yang digunakan untuk membuat skema eksternal:

- DESKRIPSI akses pada tautan sumber daya basis data
- DESKRIPSI akses pada database federasi
- PILIH dan JELASKAN pada tabel federasi

Menanyakan tabel federasi dari Amazon Redshift

Hubungkan ke cluster atau grup kerja Amazon Redshift Anda sebagai pengguna admin dan akses tabel menggunakan salah satu opsi berikut.

Opsi 1: Buat skema eksternal menggunakan peran IAM

Buat peran IAM dengan izin ke AWS Glue dan AWS Lake Formation API dan lampirkan peran ke cluster.

```
CREATE EXTERNAL SCHEMA external_schema_name
FROM DATA CATALOG DATABASE 'resource_link_database'
IAM_ROLE 'arn:aws:iam::account_id:role/spectrum_role';
```

Anda dapat memberikan izin pada skema eksternal ini kepada pengguna dan grup lokal Amazon Redshift. Identitas Amazon Redshift lokal dapat mengakses data dengan menanyakan skema menggunakan izin yang diberikan ke peran IAM yang dilampirkan ke skema eksternal.

```
SELECT * FROM "database"."external_schema_name"."table_name" LIMIT 10;
```

Opsi 2: Buat skema eksternal menggunakan IAM SESSION untuk prinsipal federasi

```
CREATE EXTERNAL SCHEMA external_schema_name
FROM DATA CATALOG DATABASE 'resource_link_database'
```

```
IAM_ROLE 'SESSION' CATALOG_ID 'account_id';
```

Masuk sebagai pengguna federasi atau dengan identitas IAM dan jalankan perintah berikut untuk menanyakan tabel:

```
SELECT * FROM "database"."external_schema_name"."table_name" LIMIT 10;
```

Opsi 3: Gunakan automount untuk menanyakan tabel federasi

Pengguna admin harus memberikan penggunaan pada database otomatis ke prinsipal federasi. Kepala sekolah kemudian dapat mengakses tabel federasi dengan menjalankan perintah berikut:

```
SELECT * FROM "awsdatacatalog"."resource_link_database"."table_name" LIMIT 10;
```

Menyiapkan dan menanyakan katalog federasi di seluruh akun

Anda dapat membagikan katalog federasi di seluruh akun menggunakan AWS Lake Formation dan mengakses tabel bersama dari akun yang berbeda menggunakan Amazon Redshift.

Akun pemberi (tempat katalog federasi berada)

Menyiapkan federasi katalog

Ikuti petunjuk di Katalog Federate to Databricks Unity di Panduan Peng AWS Lake Formation embang.

Mengatur versi lintas akun

Atur versi lintas akun ke V4 di AWS Lake Formation pengaturan. Jika izin tingkat katalog sudah ada, pastikan AWS RAM memiliki izin untuk berbagi sumber daya dengan menambahkan kebijakan sumber daya Katalog Data berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:ShareResource"
      ],
```

```
      "Principal": {
        "Service": [
          "ram.amazonaws.com"
        ]
      },
      "Resource": [
        "arn:aws:glue:region:grantor_account_id:catalog",
        "arn:aws:glue:region:grantor_account_id:database/*",
        "arn:aws:glue:region:grantor_account_id:table/*"
      ]
    }
  ]
}
```

Berbagi sumber daya federasi lintas akun

Tindakan ini secara AWS RAM otomatis membuat undangan berbagi untuk berbagi sumber daya dengan akun penerima.

Opsi 1: Bagikan ke akun untuk admin danau data untuk mengelola delegasi

Sebagai admin danau data, berikan izin berikut ke akun penerima:

- DESKRIPSI akses pada database federasi dengan izin yang dapat diberikan
- PILIH dan JELASKAN pada tabel federasi dengan izin yang dapat diberikan

Opsi 2: Bagikan langsung ke prinsipal IAM lintas akun

Sebagai admin danau data, berikan izin berikut ke prinsipal IAM di akun penerima:

- DESKRIPSI akses pada database federasi
- PILIH dan JELASKAN pada tabel federasi

Secara opsional, Anda dapat memberikan izin di atas ke akun tanpa diberikan kepada admin danau data akun penerima untuk melihat sumber daya bersama.

Akun penerima (tempat Amazon Redshift berada)

Menerima AWS RAM berbagi

Masuk sebagai admin danau data. Jika penerimaan otomatis disetel untuk akun anggota, pembagian AWS RAM akan diterima secara otomatis. Jika tidak, terima AWS RAM undangan berbagi. Setelah diterima, sumber daya bersama akan terlihat di AWS RAM konsol di bawah “Dibagikan dengan saya.”

Membuat tautan sumber daya basis data

Buat tautan sumber daya database di Katalog Data lokal yang menunjuk ke database federasi bersama. Tautan sumber daya ini berfungsi sebagai referensi lokal yang digunakan Amazon Redshift untuk mengakses data bersama.

Memberikan izin untuk mengakses tabel di Amazon Redshift

Opsi 1: Dibagikan pada tingkat akun

Sebagai admin danau data, berikan AWS Lake Formation izin berikut ke prinsipal IAM atau peran IAM yang digunakan untuk membuat skema eksternal:

- DESKRIPSIKAN pada database tautan sumber daya
- DESKRIPSIKAN pada database federasi bersama
- PILIH dan DESKRIPSIKAN pada tabel federasi bersama

Opsi 2: Dibagikan langsung ke prinsipal IAM lintas akun

Sebagai admin danau data, berikan AWS Lake Formation izin berikut ke prinsipal IAM atau peran IAM yang digunakan untuk membuat skema eksternal:

- DESKRIPSIKAN pada database tautan sumber daya

Mengkueri tabel federasi dari Amazon Redshift

Hubungkan ke cluster atau grup kerja Amazon Redshift Anda sebagai pengguna admin dan akses tabel menggunakan salah satu opsi berikut.

Opsi 1: Buat skema eksternal menggunakan peran IAM

Buat peran IAM dengan izin ke AWS Glue dan AWS Lake Formation API dan lampirkan peran ke cluster.

```
CREATE EXTERNAL SCHEMA external_schema_name
FROM DATA CATALOG DATABASE 'resource_link_database'
IAM_ROLE 'arn:aws:iam::account_id:role/spectrum_role';
```

Anda dapat memberikan izin pada skema eksternal ini kepada pengguna dan grup lokal Amazon Redshift. Identitas Amazon Redshift lokal dapat mengakses data dengan menanyakan skema menggunakan izin yang diberikan ke peran IAM yang dilampirkan ke skema eksternal.

```
SELECT * FROM "database"."external_schema_name"."table_name" LIMIT 10;
```

Opsi 2: Buat skema eksternal menggunakan IAM SESSION untuk prinsipal federasi

```
CREATE EXTERNAL SCHEMA external_schema_name
FROM DATA CATALOG DATABASE 'resource_link_database'
IAM_ROLE 'SESSION' CATALOG_ID 'account_id';
```

Masuk sebagai pengguna federasi atau dengan identitas IAM dan jalankan perintah berikut untuk menanyakan tabel:

```
SELECT * FROM "database"."external_schema_name"."table_name" LIMIT 10;
```

Opsi 3: Gunakan automount untuk menanyakan tabel federasi

Pengguna admin harus memberikan penggunaan pada database otomatis ke prinsipal federasi. Kepala sekolah kemudian dapat mengakses tabel federasi dengan menjalankan perintah berikut:

```
SELECT * FROM "awsdatacatalog"."resource_link_database"."table_name" LIMIT 10;
```

Praktik dan batasan terbaik

- Lebih suka automount untuk penerapan lintas wilayah. Automount menangani resolusi lintas wilayah secara transparan dan tidak memerlukan overhead manajemen skema.
- Gunakan penyebaran wilayah yang sama saat skema eksternal diperlukan. Jika alur kerja Anda memerlukan skema eksternal (misalnya, untuk mengontrol izin atau penamaan tingkat skema), pastikan katalog AWS Glue federasi, komputasi Amazon Redshift, dan bucket sumber Amazon S3 semuanya berada di Wilayah yang sama.

Berbagi data di Amazon Redshift

Dengan Amazon Redshift, Anda dapat berbagi data dengan aman di seluruh cluster Amazon Redshift atau dengan layanan lain AWS . Dengan berbagi data, Anda dapat berbagi data langsung, tanpa harus membuat salinan atau memindahkannya. Administrator database dan insinyur data dapat menggunakan berbagi data untuk menyediakan akses baca-saja yang aman ke data untuk tujuan analitik, sambil mempertahankan kontrol atas data. Analis data, profesional intelijen bisnis, dan ilmuwan data dapat memanfaatkan data bersama untuk mendapatkan wawasan tanpa menduplikasi atau memindahkan data. Kasus penggunaan umum termasuk berbagi data dengan mitra, memungkinkan analisis lintas fungsi, dan memfasilitasi demokratisasi data dalam suatu organisasi. Bagian berikut mencakup detail konfigurasi dan pengelolaan berbagi data di Amazon Redshift.

Dengan berbagi data Amazon Redshift, Anda dapat dengan aman berbagi akses ke data langsung di seluruh cluster Amazon Redshift, grup kerja Akun AWS, dan Wilayah AWS tanpa memindahkan atau menyalin data secara manual. Karena data aktif, semua pengguna dapat melihat informasi terbaru dan konsisten di Amazon Redshift segera setelah diperbarui.

Anda dapat berbagi data di seluruh cluster yang disediakan, grup kerja tanpa server, Zona Ketersediaan, dan. Akun AWS Wilayah AWS Anda dapat berbagi antar tipe cluster serta antara cluster yang disediakan dan tanpa server.

Anda dapat berbagi objek database untuk membaca dan menulis di berbagai cluster Amazon Redshift atau grup kerja Amazon Redshift Serverless dalam kelompok kerja yang sama Akun AWS, atau dari satu ke yang lain. Akun AWS Anda juga dapat membaca dan menulis data di seluruh Wilayah. Anda dapat memberikan izin seperti SELECT, INSERT, dan UPDATE untuk tabel yang berbeda dan USAGE dan CREATE untuk skema yang berbeda. Data langsung dan tersedia untuk semua gudang segera setelah transaksi penulisan dilakukan.

Kasus penggunaan berbagi data untuk Amazon Redshift

Berbagi data Amazon Redshift sangat berguna untuk kasus penggunaan berikut:

- Mendukung berbagai jenis beban kerja kritis bisnis — Gunakan cluster ekstrak, transformasi, dan muat (ETL) pusat yang berbagi data dengan beberapa business intelligence (BI) atau cluster analitik. Pendekatan ini menyediakan isolasi beban kerja baca dan tagihan balik untuk beban kerja individual. Anda dapat mengukur dan menskalakan komputasi beban kerja individu sesuai dengan persyaratan harga dan kinerja khusus beban kerja.

- Mengaktifkan kolaborasi lintas kelompok — Aktifkan kolaborasi yang mulus di seluruh tim dan grup bisnis untuk analisis yang lebih luas, ilmu data, dan analisis dampak lintas produk.
- Menyampaikan data sebagai layanan — Bagikan data sebagai layanan di seluruh organisasi Anda.
- Berbagi data antar lingkungan — Berbagi data di antara lingkungan pengembangan, pengujian, dan produksi. Anda dapat meningkatkan kelincahan tim dengan berbagi data pada tingkat perincian yang berbeda.
- Lisensi akses ke data di Amazon Redshift — Daftar kumpulan data Amazon Redshift di AWS Data Exchange katalog yang dapat ditemukan, berlangganan, dan kueri pelanggan dalam hitungan menit.

Kasus penggunaan akses tulis berbagi data

Berbagi data untuk menulis memiliki beberapa kasus penggunaan penting:

- Memperbarui data sumber bisnis pada produsen — Anda dapat berbagi data sebagai layanan di seluruh organisasi Anda, tetapi kemudian konsumen juga dapat melakukan tindakan pada data sumber. Misalnya, mereka dapat mengomunikasikan kembali nilai-nilai terbaru atau mengakui penerimaan data. Ini hanya beberapa kemungkinan kasus penggunaan bisnis.
- Masukkan catatan tambahan pada produsen — Konsumen dapat menambahkan catatan ke data sumber asli. Ini dapat ditandai sebagai dari konsumen, jika diperlukan.

Untuk informasi khusus mengenai cara melakukan operasi penulisan pada datashare, lihat Berbagi akses tulis ke data.

Berbagi data di berbagai tingkat di Amazon Redshift

Dengan Amazon Redshift, Anda dapat berbagi data di berbagai tingkat. Level ini mencakup database, skema, tabel, tampilan (termasuk tampilan reguler, pengikatan terlambat, dan terwujud), dan fungsi yang ditentukan pengguna SQL (UDF). Anda dapat membuat beberapa datashares untuk database tertentu. Sebuah datashare dapat berisi objek dari beberapa skema dalam database tempat berbagi dibuat.

Dengan memiliki fleksibilitas dalam berbagi data, Anda mendapatkan kontrol akses yang halus. Anda dapat menyesuaikan kontrol ini untuk pengguna dan bisnis yang berbeda yang memerlukan akses ke data Amazon Redshift.

Manajemen konsistensi untuk berbagi data di Amazon Redshift

Amazon Redshift memberikan konsistensi transaksional pada semua cluster produsen dan konsumen dan berbagi pandangan data terkini dan konsisten dengan semua konsumen.

Anda dapat terus memperbarui data pada cluster produsen. Semua kueri pada cluster konsumen dalam transaksi membaca status yang sama dari data bersama. Amazon Redshift tidak mempertimbangkan data yang diubah oleh transaksi lain pada cluster produsen yang dilakukan setelah dimulainya transaksi pada cluster konsumen. Setelah perubahan data dilakukan pada cluster produsen, transaksi baru pada cluster konsumen dapat segera menanyakan data yang diperbarui.

Konsistensi yang kuat menghilangkan risiko laporan bisnis dengan fidelitas rendah yang mungkin berisi hasil yang tidak valid selama berbagi data. Faktor ini sangat penting untuk analisis keuangan atau di mana hasilnya dapat digunakan untuk menyiapkan kumpulan data yang digunakan untuk melatih model pembelajaran mesin.

Pertimbangan untuk berbagi data di Amazon Redshift

Dengan berbagi data Amazon Redshift, Anda dapat dengan aman berbagi akses ke data langsung di seluruh cluster Amazon Redshift, grup kerja Akun AWS, dan Wilayah AWS tanpa memindahkan atau menyalin data secara manual. Sebelumnya, objek dalam datashares hanya dibaca dalam semua keadaan. Menulis ke objek di datashare adalah fitur baru. Objek dalam datashares hanya diaktifkan penulisan ketika produsen secara khusus memberikan hak penulisan seperti INSERT atau CREATE pada objek ke datashare. Selain itu, untuk berbagi lintas akun, produsen harus mengotorisasi datashare untuk penulisan dan konsumen harus mengaitkan cluster dan kelompok kerja tertentu untuk penulisan.

Bagian ini mencakup pertimbangan saat bekerja dengan berbagi data Amazon Redshift.

Topik

- Pertimbangan umum untuk berbagi data di Amazon Redshift
- Pertimbangan untuk pembacaan dan penulisan berbagi data di Amazon Redshift
- Pertimbangan untuk berbagi data di Amazon Redshift Serverless restore
- Pertimbangan untuk berbagi data dengan tabel data lake di Amazon Redshift
- Pertimbangan untuk berbagi data dengan AWS Lake Formation di Amazon Redshift
- Pertimbangan untuk berbagi data dengan AWS Data Exchange di Amazon Redshift

- Izin yang dapat Anda berikan untuk berbagi data
- Pernyataan SQL yang didukung untuk berbagi data menulis pada konsumen
- Pernyataan SQL yang tidak didukung untuk berbagi data menulis pada konsumen

Pertimbangan umum untuk berbagi data di Amazon Redshift

Berikut ini adalah pertimbangan umum saat bekerja dengan berbagi data di Amazon Redshift:

- Basis data default — Saat Anda membaca data dari datashare, Anda tetap terhubung ke database cluster lokal Anda. Untuk informasi selengkapnya tentang menyiapkan dan membaca dari database yang dibuat dari datashare, lihat Mengkueri objek datash are dan. Tampilan terwujud pada tabel danau data eksternal di Amazon Redshift Spectrum
- Koneksi — Anda harus terhubung langsung ke database datashare atau menjalankan perintah USE untuk menulis ke datashares. Anda juga dapat menggunakan notasi tiga bagian. Perintah USE tidak didukung pada tabel eksternal.
- Kinerja — Kinerja kueri pada data bersama tergantung pada kapasitas komputasi cluster konsumen.
- Biaya transfer data — berbagi Cross-Region data termasuk biaya transfer data lintas wilayah tambahan.
  - Biaya transfer data ini tidak berlaku di Wilayah yang sama, hanya di seluruh Wilayah. Untuk informasi selengkapnya, lihat Mengelola kontrol biaya untuk berbagi data lintas wilayah.
  - Konsumen dikenakan biaya untuk semua biaya transfer data komputasi dan lintas wilayah yang diperlukan untuk menanyakan data produsen. Produsen dikenai biaya untuk penyimpanan data yang mendasari di cluster yang disediakan atau namespace tanpa server.
- Berbagi data di dalam dan di antara cluster — Anda hanya perlu berbagi data saat berbagi data antara berbagai cluster Amazon Redshift yang disediakan atau grup kerja tanpa server. Dalam cluster yang sama, Anda dapat menanyakan database lain menggunakan notasi tiga bagian sederhana `database.schema.table` selama Anda memiliki izin yang diperlukan pada objek di database lain.
- Penemuan Metadata - Ketika Anda adalah konsumen yang terhubung langsung ke database datashare melalui driver Redshift JDBC, ODBC, atau Python, Anda dapat melihat data katalog dengan cara berikut:
  - Per https://docs.aws.amazon.com/redshift/latest/dg/r_SHOW.html intah SQL SHOW.
  - Mengkueri tabel dan tampilan information_schema.

  - Mengkueri tampilan metadata SVV.
- Visibilitas izin — Konsumen dapat melihat izin yang diberikan kepada data share melalui perintah SHOW GRANTS SQL.
- Manajemen enkripsi cluster untuk berbagi data — Untuk berbagi data di seluruh Akun AWS, cluster produsen dan konsumen harus dienkripsi.
  - Jika cluster produsen dan konsumen dan ruang nama tanpa server berada di akun yang sama, mereka harus memiliki jenis enkripsi yang sama (baik keduanya tidak terenkripsi, atau keduanya dienkripsi). Dalam setiap kasus lain, termasuk berbagi data yang dikelola Lake Formation, baik konsumen maupun produsen harus dienkripsi. Ini untuk tujuan keamanan. Namun, mereka tidak perlu berbagi kunci enkripsi yang sama.
  - Untuk melindungi data dalam transit, semua data dienkripsi dalam transit melalui skema enkripsi cluster produsen. Cluster konsumen mengadopsi skema enkripsi ini ketika data dimuat. Cluster konsumen kemudian beroperasi sebagai cluster terenkripsi normal. Komunikasi antara produsen dan konsumen juga dienkripsi menggunakan skema kunci bersama. Untuk informasi lebih lanjut tentang enkripsi dalam transit, Enkripsi dalam transit.

Pertimbangan untuk pembacaan dan penulisan berbagi data di Amazon Redshift

> **Note**
>
> Penulisan multi-gudang Amazon Redshift menggunakan berbagi data hanya didukung di Amazon Redshift patch 186 untuk cluster yang disediakan pada trek versi saat ini 1.0.78881 atau lebih tinggi, dan untuk grup kerja Amazon Redshift Serverless pada versi 1.0.78890 atau lebih tinggi.

Berikut ini adalah pertimbangan saat bekerja dengan membaca dan menulis datashare di Amazon Redshift:

- Anda hanya dapat berbagi SQL UDF melalui datashares. UDF Python dan Lambda tidak didukung.
- Jika database produsen memiliki kolasi tertentu, gunakan pengaturan kolasi yang sama untuk database konsumen.
- Amazon Redshift tidak mendukung fungsi SQL yang ditentukan pengguna bersarang pada cluster produsen.

- Amazon Redshift tidak mendukung berbagi tabel dengan kunci pengurutan yang disisipkan dan tampilan yang merujuk ke tabel dengan kunci pengurutan yang disisipkan.
- Amazon Redshift tidak mendukung akses objek datashare yang memiliki DDL bersamaan terjadi antara Prepare dan Execute akses.
- Amazon Redshift tidak mendukung berbagi prosedur tersimpan melalui berbagi data.
- Amazon Redshift tidak mendukung berbagi tampilan sistem metadata dan tabel sistem.
- Jenis komputasi — Anda harus menggunakan grup kerja tanpa server atau cluster yang disediakan dengan tipe node RG atau RA3 untuk menggunakan fitur ini.
- Tingkat isolasi — Tingkat isolasi database Anda harus berupa isolasi snapshot untuk memungkinkan kelompok kerja Tanpa Server lainnya dan cluster yang disediakan untuk menulis ke sana.
- Multi-statement query dan transaksi — Multi-statement kueri di luar blok transaksi saat ini tidak didukung. Akibatnya, jika Anda menggunakan editor kueri seperti dbeaver dan Anda memiliki beberapa kueri tulis, Anda perlu membungkus kueri Anda dalam pernyataan transaksi BEGIN... END eksplisit.

  Ketika pernyataan multi-perintah digunakan di luar transaksi, jika perintah pertama adalah menulis ke database produsen, perintah tulis berikutnya dalam pernyataan hanya diizinkan untuk database produsen tersebut. Jika perintah pertama adalah baca, perintah tulis berikutnya hanya diperbolehkan ke database yang digunakan, jika disetel, jika tidak ke database lokal. Perhatikan bahwa penulisan dalam transaksi hanya didukung untuk satu database.
- Ukuran konsumen — Cluster konsumen harus memiliki setidaknya 64 irisan atau lebih untuk melakukan penulisan menggunakan berbagi data.
- Tampilan dan tampilan terwu jud — Anda tidak dapat membuat, memperbarui, atau mengubah tampilan atau tampilan terwujud pada database datashare.
- Keamanan — Anda tidak dapat melampirkan atau menghapus kebijakan keamanan seperti tingkat kolom (CLS), tingkat baris (RLS), dan penyamaran data dinamis (DDM) ke objek datashare.
- Kemampuan pengelolaan — Gudang konsumen tidak dapat menambahkan objek atau tampilan datashare yang mereferensikan objek datashare ke datashare lain. Konsumen juga tidak dapat memodifikasi atau menghapus data yang ada.
- Operasi potong — Datashare menulis dukungan pemotongan transaksional untuk tabel jarak jauh. Ini berbeda dari potongan yang Anda jalankan secara lokal di cluster, yang merupakan komit otomatis. Untuk informasi selengkapnya tentang perintah SQL, lihat TRUNCATE.

- Kloning — CREATE TABLE dengan pernyataan klausa LIKE mendukung kloning dari tabel induk tunggal saat Anda menulis dari gudang konsumen ke produsen.

Pertimbangan untuk berbagi data di Amazon Redshift Serverless restore

Pertimbangkan hal berikut saat bekerja dengan berbagi data selama operasi pemulihan tanpa server Amazon Redshift:

Kembalikan snapshot ke namespace produsen Amazon Redshift Serverless:

- Memulihkan snapshot ke namespace produsen Amazon Redshift Serverless menggantikan data share saat ini dengan data share di snapshot. Namun, izin datashare hanya akan dipertahankan jika SEMUA kondisi berikut terpenuhi:
  - Snapshot diambil pada namespace yang sama dengan yang dipulihkan pelanggan, DAN
  - Izin datashare ada di namespace saat ini, DAN
  - Izin datashare ada di namespace saat snapshot diambil.
- Jika datashare diberikan kepada konsumen saat mengambil snapshot, tetapi dihentikan atau hibah dicabut pada namespace terbaru, datashare akan dipulihkan. Namun, izin datashare yang diberikan ke cluster konsumen tidak lagi valid setelah pemulihan. Untuk menghidupkan kembali akses datashare, berikan kembali izin penggunaan data share ke cluster konsumen yang diinginkan.
- Jika datashare diberikan kepada konsumen pada namespace terbaru, tetapi tidak diberikan saat mengambil snapshot, akses datashare tidak akan dipertahankan setelah pemulihan.
- Cak authorization/association upan datashare tidak akan berubah pada namespace setelah pemulihan. Misalnya, sebelum memulihkan jika datashare dikaitkan dengan keseluruhan Akun AWS, setelah memulihkan cakupan asosiasi akan tetap di tingkat akun terlepas dari cakupan asosiasi saat mengambil snapshot.
- Memulihkan dari snapshot akan mengembalikan pengaturan yang dapat diakses publik ke statusnya pada saat snapshot dibuat, terlepas dari perubahan berikutnya.

Kembalikan snapshot ke namespace konsumen Amazon Redshift Serverless:

- Memulihkan namespace konsumen mempertahankan akses datashare tanpa mengharuskan administrator produsen untuk memberikan kembali penggunaan. Namun, jika database yang dibuat dari datashare tidak lagi ada setelah pemulihan, konsumen harus membuatnya kembali dari datashare.

Pertimbangan untuk berbagi data dengan tabel data lake di Amazon Redshift

Berikut ini adalah pertimbangan saat bekerja dengan tabel data lake di Amazon Redshift:

- Berbagi data tabel data lake tidak mendukung enkripsi bucket Amazon S3 yang dikelola AWS KMS keys pelanggan. Anda dapat menggunakan Kunci yang dikelola AWS untuk enkripsi. Untuk informasi selengkapnya, lihat Menggunakan enkripsi sisi server dengan kunci terkelola Amazon S3 ()SSE-S3.
- Untuk berbagi data tabel data lake dari AWS Glue katalog terenkripsi, Anda harus mendelegasikan AWS KMS operasi ke peran IAM dengan mengikuti petunjuk dalam Enkripsi Katalog Data Anda.
- Tabel eksternal yang secara eksplisit menentukan file manifes dalam `LOCATION` klausa tidak didukung untuk berbagi data. Ini termasuk tabel berikut yang didukung Amazon Redshift Spectrum:
  - Danau Delta
  - Hudi

Pertimbangan untuk berbagi data dengan AWS Lake Formation di Amazon Redshift

Berikut ini adalah pertimbangan dan batasan untuk berbagi data Amazon Redshift dengan Lake Formation. Untuk informasi tentang pertimbangan dan batasan berbagi data, lihat Per timbangan saat menggunakan berbagi data di Amazon Redshift. Untuk informasi tentang batasan Formasi Danau, lihat Catatan tentang bekerja dengan berbagi data Amazon Redshift di Lake Formation.

- Berbagi datashare ke Formasi Danau di seluruh Wilayah saat ini tidak didukung.
- Jika filter tingkat kolom ditentukan untuk pengguna pada relasi bersama, melakukan `SELECT *` operasi hanya mengembalikan kolom yang dapat diakses pengguna.
- Cell-level filter dari Formasi Danau tidak didukung.
- Jika Anda membuat dan membagikan tampilan beserta tabelnya ke Lake Formation, Anda dapat mengonfigurasi filter untuk mengelola akses tabel, Amazon Redshift memberlakukan kebijakan yang ditentukan Lake Formation saat pengguna cluster konsumen mengakses objek bersama. Saat pengguna mengakses tampilan yang dibagikan dengan Lake Formation, Redshift hanya memberlakukan kebijakan Formasi Danau yang ditentukan pada tampilan dan bukan tabel yang terkandung dalam tampilan. Namun, ketika pengguna langsung mengakses tabel, Redshift memberlakukan kebijakan Formasi Danau yang ditentukan di atas meja.

- Anda tidak dapat membuat tampilan terwujud pada konsumen berdasarkan tabel bersama jika tabel memiliki filter Lake Formation yang dikonfigurasi.
- Administrator Danau Formation harus memiliki izin administrator danau data dan izin yang diperlukan untuk menerima datashare.
- Cluster konsumen produsen harus berupa cluster RG atau RA3 dengan versi cluster Amazon Redshift terbaru atau kelompok kerja tanpa server untuk berbagi data melalui Lake Formation.
- Cluster produsen dan konsumen harus dienkripsi.
- Kebijakan kontrol akses tingkat baris dan tingkat kolom pergeseran merah yang diterapkan di cluster produsen atau kelompok kerja diabaikan saat datashare dibagikan ke Lake Formation. Administrator Formasi Danau harus mengonfigurasi kebijakan ini di Formasi Danau. Administrator cluster produsen atau workgroup dapat mematikan RLS untuk tabel dengan menggunakan perintah ALTER TA BLE.
- Berbagi data share melalui Lake Formation hanya tersedia untuk pengguna yang memiliki akses ke Redshift dan Lake Formation.

Pertimbangan untuk berbagi data dengan AWS Data Exchange di Amazon Redshift

Saat menggunakan AWS Data Exchange untuk Amazon Redshift, pertimbangkan hal berikut:

- Baik produsen maupun konsumen harus menggunakan tipe instans RG atau RA3 untuk menggunakan data share Amazon Redshift. Produsen harus menggunakan jenis instans RG atau RA3 dengan versi cluster Amazon Redshift terbaru.
- Cluster produsen dan konsumen harus dienkripsi.
- Anda harus terdaftar sebagai AWS Data Exchange penyedia untuk membuat daftar produk AWS Data Exchange, termasuk produk yang berisi AWS Data Exchange data share. Untuk informasi selengkapnya, lihat Mem ulai sebagai penyedia.
- Anda tidak perlu menjadi AWS Data Exchange penyedia terdaftar untuk menemukan, berlangganan, dan menanyakan data Amazon Redshift. AWS Data Exchange
- Untuk mengontrol akses ke data Anda, buat berbagi AWS Data Exchange data dengan pengaturan yang dapat diakses publik diaktifkan. Untuk mengubah AWS Data Exchange datashare untuk mematikan pengaturan yang dapat diakses publik, setel variabel sesi untuk mengizinkan ALTER DATASHARE SET PUBLICACESSIBLE FALSE. Untuk informasi selengkapnya, lihat Catatan penggunaan ALTER DATASHARE.

- Produsen tidak dapat menambahkan atau menghapus konsumen secara AWS Data Exchange manual dari berbagi data karena akses ke data share diberikan berdasarkan langganan aktif ke AWS Data Exchange produk yang berisi datashare. AWS Data Exchange
- Produsen tidak dapat melihat kueri SQL yang dijalankan konsumen. Mereka hanya dapat melihat metadata, seperti jumlah kueri atau objek yang diminta konsumen, melalui tabel Amazon Redshift yang hanya dapat diakses oleh produsen. Untuk informasi selengkapnya, lihat Memantau dan mengaudit berbagi data di Amazon Redshift.
- Kami menyarankan Anda membuat data share Anda dapat diakses publik. Jika tidak, pelanggan AWS Data Exchange dengan cluster konsumen yang dapat diakses publik tidak akan dapat menggunakan datashare Anda.
- Kami menyarankan agar Anda tidak menghapus data yang AWS Data Exchange dibagikan ke orang lain Akun AWS menggunakan pernyataan DROP DATASHARE. Jika Anda melakukannya, Akun AWS yang memiliki akses ke datashare akan kehilangan akses. Tindakan ini tidak dapat diubah. Melakukan perubahan jenis ini dapat melanggar ketentuan produk data AWS Data Exchange. Jika Anda ingin menghapus AWS Data Exchange datashare, lihat. Catatan penggunaan DROP DATASHARE
- Untuk berbagi data lintas wilayah, Anda dapat membuat berbagi data untuk berbagi AWS Data Exchange data berlisensi.
- Saat mengonsumsi data dari Wilayah yang berbeda, konsumen membayar biaya transfer Cross-Region data dari Wilayah produsen ke Wilayah konsumen.

Izin yang dapat Anda berikan untuk berbagi data

Jenis objek yang berbeda dan berbagai izin yang dapat Anda berikan kepada mereka dalam konteks berbagi data.

Basis data:

- CREATE
- PENGGUNAAN
- MENGUBAH
- MENJATUHKAN

Skema:

- CREATE
- PENGGUNAAN
- MENGUBAH
- MENJATUHKAN

Tabel:

- SELECT
- INSERT
- UPDATE
- DELETE
- MEMOTONG
- MENJATUHKAN
- MENGUBAH
- REFERENSI

Fungsi:

- EXECUTE

Pernyataan SQL yang didukung untuk berbagi data menulis pada konsumen

Pernyataan Data Definition Language (DDL) berikut ini didukung untuk berbagi data dengan penulisan:

- ALTER TABLE RENAME TO
- UBAH TABEL GANTI NAMA KOLOM MENJADI
- UBAH ADD/DROP KOLOM TABEL
- UBAH SKEMA GANTI NAMA MENJADI
- {BUAT | JATUHKAN} SKEMA
- {BUAT | JATUHKAN | TAMPILKAN} TABEL
- BUAT TABEL table_name AS

- MULAI | MULAI TRANSAKSI
- AKHIR | KOMIT | KEMBALIKAN
- MEMOTONG

Pernyataan Data Manipulation Language (DML) berikut ini didukung untuk berbagi data dengan penulisan:

- SELECT
- INSERT
- SISIPKAN KE PILIH
- UPDATE
- DELETE
- MERGE
- COPY tanpa COMPUPDATE

Pernyataan analisis berikut didukung untuk berbagi data dengan penulisan:

- MENGANALISIS. Konsumen menjalankan USER ANALYZE dan mengirimkan statistik ke produsen.
- Analisis diaktifkan dengan CTAS/COPY menjalankan pada konsumen. Ini termasuk multiplexing untuk konsumen kecil.
- Auto-analyze jalankan pada produser setelah COPY.

Pernyataan izin berikut didukung untuk berbagi data dengan penulisan:

- {GRANT | CABUT} privilege_name PADA OBJECT_TYPE object_name KE consumer_user
- MENUNJUKKAN HIBAH. Untuk informasi selengkapnya, lihat TUNJUKKAN HIBAH.

Pernyataan SQL yang tidak didukung untuk berbagi data menulis pada konsumen

Berikut ini tidak didukung:

- Multi-statement pertanyaan ke gudang konsumen saat menulis ke produsen.

- Multi-statement kueri ke gudang konsumen dalam database yang berbeda, jika perintah sebelumnya adalah pernyataan baca.
- Referensi objek selain notasi tiga titik, seperti notasi satu.dot atau two.dot, jika tidak terhubung ke database bersama.
- Penulisan kueri penskalaan konkurensi dari konsumen ke produsen.
- Auto-copy pekerjaan menulis dari konsumen ke produsen.
- Streaming pekerjaan menulis dari konsumen ke produsen.
- Konsumen membuat tabel integrasi Zero-ETL pada cluster produsen. Untuk informasi selengkapnya tentang integrasi Zero-ETL, lihat Bek erja dengan integrasi Zero-ETL.
- Menulis ke tabel dengan kunci pengurutan yang disisipkan melalui berbagi data.
- Menulis ke prosedur tersimpan melalui berbagi data.
- Menulis ke fungsi yang ditentukan pengguna SQL (UDF) melalui berbagi data. Ini termasuk UDF bersarang, Python, dan Lambda.
- PERBARUI, SISIPKAN, atau SALIN pernyataan pada kolom identitas ke gudang konsumen dengan irisan komputasi lebih banyak daripada produsen.
- GABUNG pernyataan pada tabel eksternal non-RMS ke gudang konsumen, saat menulis ke produsen.
- BUAT pernyataan TABEL dengan:
  - Ekspresi DEFAULT diatur ke tipe data VARBYTE. Tipe data VARBYTE tidak dapat dikonversi secara implisit ke tipe data lainnya. Untuk informasi selengkapnya, lihat Fungsi CAST.
  - AS OF SELECT pernyataan dengan parameter NULL ke gudang konsumen, saat menulis ke produsen.

Wilayah AWS Di mana berbagi data tersedia

Tabel berikut mencantumkan ketersediaan untuk kemampuan baca dan tulis berbagi data.

Region	Same-region berbagi data	Cross-region berbagi data	AWS Lake Formation berbagi data yang diatur
AS Timur (Virginia Utara) (us-east-1)	Ya	Ya	Ya

Region	Same-region berbagi data	Cross-region berbagi data	AWS Lake Formation berbagi data yang diatur
AS Timur (Ohio) (us-east-2)	Ya	Ya	Ya
AS Barat (California Utara) (us-west-1)	Ya	Ya	Ya
AS Barat (Oregon) (us-west-2)	Ya	Ya	Ya
Africa (Cape Town) (af-south-1)	Ya	Tidak	Ya
Asia Pacific (Hong Kong) (ap-east-1)	Ya	Tidak	Ya
Asia Pasifik (Hyderabad) (ap-south-2)	Ya	Tidak	Ya
Asia Pasifik (Jakarta) (ap-Southeast-3)	Ya	Tidak	Ya
Asia Pasifik (Malaysia) (ap-Southeast-5)	Ya	Tidak	Ya
Asia Pasifik (Melbourne) (ap-Southeast-4)	Ya	Tidak	Ya
Asia Pasifik (Mumbai) (ap-south-1)	Ya	Ya	Ya
Asia Pasifik (Selandia Baru) (ap-south east-6)	Ya	Tidak	Ya

Region	Same-region berbagi data	Cross-region berbagi data	AWS Lake Formation berbagi data yang diatur
Asia Pasifik (Osaka) (ap-northeast-3)	Ya	Ya	Ya
Asia Pasifik (Seoul) (ap-northeast-2)	Ya	Ya	Ya
Asia Pasifik (Singapura) (ap-southeast-1)	Ya	Ya	Ya
Asia Pasifik (Sydney) (ap-southeast-2)	Ya	Ya	Ya
Asia Pasifik (Taipei) (ap-east-2)	Ya	Tidak	Ya
Asia Pasifik (Thailand) (ap-southeast-7)	Ya	Tidak	Ya
Asia Pasifik (Tokyo) (ap-northeast-1)	Ya	Ya	Ya
Kanada (Pusat) (ca-central-1)	Ya	Ya	Ya
Kanada Barat (Calgary) (ca-barat-1)	Ya	Tidak	Ya
Tiongkok (Beijing) (cn-utara-1)	Ya	Ya	Ya
China (Ningxia) (cn-barat laut-1)	Ya	Ya	Ya
Eropa (Frankfurt) (eu-central-1)	Ya	Ya	Ya

Region	Same-region berbagi data	Cross-region berbagi data	AWS Lake Formation berbagi data yang diatur
Eropa (Irlandia) (eu-west-1)	Ya	Ya	Ya
Eropa (London) (eu-west-2)	Ya	Ya	Ya
Europe (Milan) (eu-south-1)	Ya	Tidak	Ya
Eropa (Paris) (eu-west-3)	Ya	Ya	Ya
Eropa (Spanyol) (eu-selatan-2)	Ya	Tidak	Ya
Eropa (Stockholm) (eu-north-1)	Ya	Ya	Ya
Eropa (Zurich) (eu-central-2)	Ya	Tidak	Ya
Israel (Tel Aviv) (pusat-1)	Ya	Tidak	Ya
Meksiko (Tengah) (mx-central-1)	Ya	Tidak	Ya
Middle East (Bahrain) (me-south-1)	Ya	Tidak	Ya
Middle East (UAE) (Timur Tengah (UAE)	Ya	Tidak	Ya
Amerika Selatan (Sao Paulo) (sa-east-1)	Ya	Ya	Ya

Region	Same-region berbagi data	Cross-region berbagi data	AWS Lake Formation berbagi data yang diatur
AWS GovCloud (US-East) (us-gov-east-1)	Ya	Tidak	Ya
AWS GovCloud (US-West) (us-gov-west-1)	Ya	Tidak	Ya

Memulai berbagi data di Amazon Redshift

Anda dapat memulai berbagi data dengan mengikuti salah satu panduan di bagian ini.

Topik

- Memulai berbagi data baca-saja di konsol
- Memulai dengan berbagi data read-only dengan antarmuka SQL
- Memulai penulisan multi-gudang menggunakan berbagi data di Amazon Redshift
- Memulai berbagi data dengan AWS CloudFormation di Amazon Redshift

Memulai berbagi data baca-saja di konsol

Dengan Amazon Redshift, Anda dapat mengelola berbagi data dengan menulis menggunakan konsol untuk mengontrol akses dan mengatur data di seluruh cluster dan AWS akun Amazon Redshift. Bagian berikut memberikan petunjuk langkah demi langkah tentang mengonfigurasi dan mengelola berbagi data dengan penulisan menggunakan konsol Amazon Redshift.

Gunakan konsol Amazon Redshift untuk mengelola berbagi data yang dibuat di akun Anda atau dibagikan dari akun lain.

Anda memerlukan izin untuk membuat, mengedit, atau menghapus data share. Untuk informasi, lihat Mengelola izin untuk berbagi data di Amazon Redshift.

- Jika Anda adalah administrator produsen, Anda dapat membuat datashares, menambahkan konsumen data, menambahkan objek datashare, membuat database dari datashares, mengedit datashares, atau menghapus datashares dari tab CLUSTERS.

Dari menu navigasi, navigasikan tab Cluster, pilih cluster dari daftar cluster. Kemudian lakukan salah satu hal berikut:

- Pilih tab Datashares, pilih datashare dari Datashares yang dibuat di bagian namespace saya. Kemudian lakukan salah satu hal berikut:
  - Buat datashare

    Saat datashare dibuat, Anda dapat menambahkan objek datashare atau konsumen data. Untuk informasi selengkapnya, lihat Tambahkan objek datashare ke datashares dan Tambahkan konsumen data ke datashares.
  - Mengedit data share yang dibuat di akun Anda
  - Menghapus data yang dibuat di akun Anda
- Pilih Datashares dan pilih datashare dari bagian Datashares dari cluster lain. Kemudian lakukan salah satu hal berikut:
  - Buat datashare
  - Membuat database dari datashares
- Pilih Database dan pilih database dari bagian Database. Kemudian pilih Buat datashare. Untuk informasi selengkapnya, lihat Membuat database dari datashares.

> **Note**
>
> Untuk melihat database dan objek dalam database atau untuk melihat berbagi data di cluster, sambungkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.

Menghubungkan ke basis data

Dengan Amazon Redshift, Anda dapat membuat koneksi ke cluster gudang data Anda dan menjalankan kueri SQL, memuat data, atau melakukan tugas administratif. Menghubungkan ke database mengacu pada proses pembuatan saluran aman antara aplikasi atau alat klien dan cluster Amazon Redshift. Bagian berikut akan memberikan petunjuk langkah demi langkah tentang cara menyambung ke database Amazon Redshift.

Hubungkan ke database untuk melihat database dan objek dalam database di cluster ini atau untuk melihat berbagi data.

KredenSIAL pengguna yang digunakan untuk terhubung ke database tertentu harus memiliki izin yang diperlukan untuk melihat semua data share.

Jika tidak ada koneksi lokal, lakukan salah satu hal berikut:

- Di halaman detail cluster, dari tab Database, di bagian Database atau Datashare object, pilih Connect to database untuk melihat objek database di cluster.
- Di halaman detail cluster, dari tab Datashares, lakukan salah satu hal berikut:
  - Di bagian Ber bagi data dari cluster lain, pilih Hubun gkan ke database untuk melihat berbagi data dari cluster lain.
  - Di bagian Datashares yang dibuat di cluster saya, pilih Hubun gkan ke database untuk melihat berbagi data di cluster Anda.
- Pada jendela Connect to database, lakukan salah satu hal berikut:
  - Jika Anda memilih Buat koneksi baru, pilih Manajer AWS Rahasia untuk menggunakan rahasia yang disimpan untuk mengotentikasi akses koneksi.

    Atau, pilih Kreden SIAL sementara untuk menggunakan kredenSIAL database untuk mengotentikasi akses untuk koneksi. Tentukan nilai untuk nama Database dan pengguna Database.

    Pilih Hubungkan.
  - Pilih Gunakan koneksi terbaru untuk menyambung ke database lain yang memiliki izin yang diperlukan.

    Amazon Redshift secara otomatis membuat koneksi.

Setelah koneksi database dibuat, Anda dapat mulai membuat data share, menanyakan data share, atau membuat database dari datashares.

Membuat datashares

Dengan Amazon Redshift, Anda dapat berbagi data langsung di seluruh cluster atau AWS akun Amazon Redshift menggunakan berbagi data. Datashare adalah objek produser-konsumen yang Anda gunakan untuk berbagi data langsung dari cluster Amazon Redshift Anda dengan cluster atau akun lain. AWS Membuat berbagi data memungkinkan berbagi data yang aman sambil mempertahankan kontrol atas akses dan memastikan data tetap mutakhir. Bagian berikut memberikan detail tentang membuat berbagi data dan menambahkan objek database seperti skema, tabel, dan tampilan untuk berbagi data langsung dengan aman.

Buat datashare

Datashare adalah wadah logis objek database, izin, dan konsumen. Konsumen adalah cluster yang disediakan Amazon Redshift atau ruang nama Amazon Redshift Serverless di akun Anda dan lainnya. Akun AWS Setiap datashare dikaitkan dengan database yang dibuat dan hanya objek dari database itu yang dapat ditambahkan. Sebagai administrator produsen, Anda dapat membuat data share di konsol dan dengan SQL dengan mengikuti salah satu prosedur di bawah ini.

Console

Di konsol, Anda dapat membuat data share dari tab Datashares di halaman detail cluster atau namespace. Setelah datashare dibuat, Anda dapat membuat database dari datashare pada konsumen sebagai administrator konsumen.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Di halaman detail cluster atau namespace, dari tab Datashares, di bagian Datashares, sambungkan ke database jika Anda tidak memiliki koneksi database. Di bagian Datashares yang dibuat di akun saya, pilih Buat datashare. Halaman Buat datashare muncul.
4. Pilih Buat datashare. Anda hanya dapat membuat datashare dari database lokal. Jika Anda belum terhubung ke database, halaman Connect to database akan muncul. Ikuti langkah-langkah Menghubungkan ke basis data untuk terhubung ke database. Jika ada koneksi baru-baru ini, halaman Buat datashare muncul.
5. Di bagian informasi Datashare, pilih salah satu dari yang berikut ini:
   - Pilih Datashare untuk membuat berbagi data untuk berbagi data untuk tujuan baca atau tulis di berbagai gudang data Amazon Redshift (cluster yang disediakan atau titik akhir Tanpa Server) atau di gudang data yang sama atau berbeda. Akun AWS Akun AWS
   - Pilih AWS Data Exchange datashare untuk membuat data share untuk melisensikan data Anda. AWS Data Exchange
6. Tentukan nilai untuk nama Datashare, Nama Database, dan Akses publik. Saat Anda mengubah nama database, buat koneksi database baru.
7. Tambahkan objek ke datashare Anda menggunakan bagian Izin yang dic akup atau Izin Lang sung. Untuk menambahkan objek ke datashare, lihat. Membuat datashare di Amazon Redshift
8. Di bagian Konsumen data, Anda dapat memilih untuk mempublikasikan ke Amazon Redshift, atau mempublikasikan ke AWS Glue Data Catalog, yang memulai proses berbagi data dengan

Lake Formation. Menerbitkan datashare Anda ke Amazon Redshift berarti berbagi data Anda dengan namespace lain atau akun Amazon Redshift yang bertindak sebagai konsumen.

> **Note**
>
> Setelah datashare dibuat, Anda tidak dapat mengedit konfigurasi untuk dipublikasikan ke opsi lain.

9. Pilih Buat datashare.

SQL

Perintah berikut membuat datashare:

```
CREATE DATASHARE salesshare;
```

Pada saat pembuatan datashare, setiap datashare dikaitkan dengan database. Hanya objek dari database itu yang dapat dibagikan di datashare itu. Beberapa data share dapat dibuat pada database yang sama dengan perincian objek yang sama atau berbeda. Tidak ada batasan jumlah data share yang dapat dibuat oleh cluster. Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat berbagi data. Untuk informasi selengkapnya, lihat BUAT DATASHARE.

Anda juga dapat mengontrol pembatasan keamanan ke datashare selama pembuatan. Contoh berikut menunjukkan bahwa konsumen dengan akses IP publik diizinkan untuk membaca datashare.

```
CREATE DATASHARE my_datashare [PUBLICACCESSIBLE = TRUE];
```

Menyetel PUBLICACCESSIBLE = TRUE memungkinkan konsumen untuk menanyakan data Anda dari cluster yang dapat diakses publik dan kelompok kerja yang disediakan. Tinggalkan ini atau secara eksplisit atur ke false jika Anda tidak ingin mengizinkannya.

Anda dapat memodifikasi properti tentang jenis konsumen setelah Anda membuat datashare. Misalnya, Anda dapat menentukan bahwa cluster yang ingin menggunakan data dari datashare tertentu tidak dapat diakses publik. Kueri dari cluster konsumen yang tidak memenuhi batasan keamanan yang ditentukan dalam datashare ditolak saat runtime kueri. Untuk informasi selengkapnya, lihat ALTER DATA.

Tambahkan objek datashare ke datashares

Anda dapat menambahkan objek database dari berbagai jenis pada konsol dan dengan SQL dengan mengikuti salah satu prosedur di bawah ini.

Console

Anda dapat menambahkan objek ke datashare Anda menggunakan izin terc akup atau bagian izin langsung. Pilih Berikan izin terlingkup atau Ber ikan izin langsung untuk menambahkan objek. Pilih tombol Tambah untuk menambahkan objek. Dialog muncul. Lakukan langkah-langkah berikut ini:

1. Jika Anda memilih Ber ikan izin bercakupan, halaman Berikan izin cakupan akan muncul di mana Anda dapat memberikan izin cakupan baik di tingkat database atau skema. Berbagi data dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema. Untuk lebih jelasnya lihat,Izin terlingkup.

    1. Selanjutnya, pilih Iz in cakupan database untuk memberikan izin terlingkup di tingkat database. Saat Anda memberikan izin cakupan, izin tersebut berlaku untuk database saat ini saat membuat datashare. Izin ini tidak dapat diberikan ke objek individual dan berlaku untuk objek yang ada dan baru (skema, tabel, tampilan, UDF).
    2. Pilih izin cakupan untuk skema, tabel atau tampilan atau fungsi yang ditentukan pengguna. Ini berarti semua objek dalam database memiliki izin yang dipilih yang diberikan kepada konsumen. Pilih Grant untuk menyelesaikan pemberian izin bercakupan database.
    3. Kemudian, pilih izin cakupan skema untuk memberikan izin cakupan di tingkat skema. Saat Anda memberikan izin cakupan skema, semua objek yang ditambahkan ke skema memiliki izin datashare yang ditentukan.
    4. Pilih skema yang ingin Anda tambahkan ke datashare dari dropdown. Anda hanya dapat memilih satu skema pada satu waktu. Kemudian, pilih izin langsung yang ingin Anda berikan pada skema yang dipilih.
    5. Pilih izin cakupan untuk objek skema Anda seperti tabel, tampilan, dan fungsi yang ditentukan pengguna. Izin diberikan pada semua objek yang cocok dalam skema. Ini bisa berupa objek yang ada atau yang ditambahkan di masa mendatang. Saat diterapkan, Anda tidak dapat menghapus izin dari objek tanpa mencabut izin cakupan.
    6. Pilih Grant untuk menyelesaikan pemberian izin bercakupan skema.
2. Jika Anda memilih Berikan izin langsung, halaman Berikan izin langsung muncul di mana Anda dapat memberikan izin langsung pada setiap tingkat objek seperti skema, tabel,

tampilan, atau fungsi yang ditentukan pengguna. Untuk memberikan izin langsung, Anda harus terlebih dahulu menambahkan skema yang relevan ke datashare.

1. Selanjutnya, pilih Berikan izin langsung ke skema untuk menerapkan izin langsung pada skema tertentu. Kemudian, pilih izin skema untuk objek skema Anda seperti tabel, tampilan, dan fungsi yang ditentukan pengguna dan pilih skema yang ingin Anda tambahkan ke datashare. Pilih Grant untuk menyelesaikan penambahan skema ke datashare.
2. Setelah Anda memiliki skema yang ditambahkan ke datashare Anda, Anda dapat melanjutkan dengan menambahkan izin langsung untuk objek skema Anda. Pilih Berikan izin langsung lagi. Halaman Beri izin langsung muncul. Kemudian, arahkan ke tab izin langsung untuk objek skema.
3. Pilih Berikan izin langsung ke tabel dan tampilan untuk memberikan izin langsung tingkat objek pada objek ini. Pilih izin langsung yang diperlukan dan objek yang diperlukan dari daftar. Gunakan bidang pencarian untuk menemukan objek datashare. Pilih Grant untuk menyelesaikan penambahan tabel dan tampilan ke datashare.
4. Pilih Berikan izin langsung ke fungsi yang ditentukan pengguna untuk memberikan izin langsung tingkat objek pada fungsi yang ditentukan pengguna. Pilih izin langsung yang diperlukan dan objek yang diperlukan dari daftar. Gunakan bidang pencarian untuk menemukan objek datashare. Pilih Grant untuk menyelesaikan penambahan fungsi yang ditentukan pengguna ke datashare.

3. Anda juga dapat memilih apakah Anda ingin menambahkan objek masa depan. Ketika Anda memilih untuk menyertakan objek datashare yang ditambahkan ke skema, itu berarti bahwa objek yang ditambahkan ke skema ditambahkan ke datashare secara otomatis.
4. Pilih Tambah untuk menyelesaikan bagian dan menambahkan objek. Mereka terdaftar di bawah objek Datashare.
5. Setelah menambahkan objek, Anda dapat memilih objek individual dan mengedit izin mereka. Jika Anda memilih skema, dialog muncul yang menanyakan apakah Anda ingin menambahkan izin yang dicakup. Ini membuatnya sehingga setiap objek yang ada atau ditambahkan ke skema memiliki kumpulan izin yang telah dipilih sebelumnya, sesuai untuk jenis objek. Misalnya, administrator dapat mengatur bahwa semua tabel yang ditambahkan memiliki izin SELECT dan UPDATE, misalnya.
6. Semua objek datashare terdaftar di bawah bagian Izin terc akup atau Izin Lang sung.
7. Di bagian Konsumen data, Anda dapat menambahkan ruang nama atau menambahkan AWS akun sebagai konsumen datashare.

8. Pilih Buat datashare untuk menyimpan perubahan Anda.

Setelah Anda membuat datashare, itu muncul dalam daftar di bawah Datashares yang dibuat di namespace saya. Jika Anda memilih datashare dari daftar, Anda dapat melihat konsumen, objeknya, dan properti lainnya.

SQL

Dengan SQL, pemilik datashare harus memberikan USAGE pada skema yang ingin mereka tambahkan ke datashare. GRANT digunakan untuk memberikan berbagai tindakan pada skema, termasuk CREATE dan USAGE. Skema menyimpan objek bersama:

```
CREATE SCHEMA myshared_schema1;
CREATE SCHEMA myshared_schema2;

GRANT USAGE ON SCHEMA myshared_schema1 TO DATASHARE my_datashare;
GRANT CREATE, USAGE ON SCHEMA myshared_schema2 TO DATASHARE my_datashare;
```

Atau, administrator dapat terus menjalankan perintah ALTER untuk menambahkan skema ke datashare. Hanya izin USAGE yang diberikan ketika skema ditambahkan dengan cara ini.

```
ALTER DATASHARE my_datashare ADD SCHEMA myshared_schema1;
```

Setelah administrator menambahkan skema, mereka dapat memberikan izin datashare pada objek dalam skema. Ini bisa berupa izin baca dan tulis. Sampel GRANT ALL menunjukkan cara memberikan semua izin.

```
GRANT SELECT, INSERT ON TABLE myshared_schema1.table1, myshared_schema1.table2,
 myshared_schema2.table1
TO DATASHARE my_datashare;

GRANT ALL ON TABLE myshared_schema1.table4 TO DATASHARE my_datashare;
```

Anda dapat terus menjalankan perintah seperti ALTER DATASHARE untuk menambahkan tabel. Ketika Anda melakukannya, hanya izin SELECT yang diberikan pada objek yang ditambahkan.

```
ALTER DATASHARE my_datashare ADD TABLE myshared_schema1.table1,
 myshared_schema1.table2, myshared_schema2.table1;
```

Anda dapat memberikan izin cakupan ke datashare pada semua objek jenis dalam database atau skema. Berbagi data dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema.

Anda dapat melihat cakupan izin cakupan tingkat database di. SVV_DATABASE_HAK ISTIMEWA Anda dapat melihat cakupan izin cakupan tingkat skema di. SVV_SCHEMA_HAK ISTIMEWA

Berikut ini adalah sintaks untuk memberikan izin cakupan untuk berbagi data. Untuk informasi selengkapnya tentang izin bercakupan, lihatIzin terlingkup.

```
GRANT { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }FOR SCHEMAS IN
DATABASE db_name
TO DATASHARE { datashare_name}

GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [PRIVILEGES] } }FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO DATASHARE { datashare_name}

GRANT { EXECUTE | ALL [ PRIVILEGES ] }FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO DATASHARE { datashare_name}
```

Tambahkan konsumen data ke datashares

Anda dapat menambahkan satu atau lebih konsumen data ke datashares. Konsumen data dapat berupa ruang nama yang secara unik mengidentifikasi cluster Amazon Redshift atau. Akun AWS

Anda harus secara eksplisit memilih untuk mematikan atau mengaktifkan berbagi data Anda ke cluster dengan akses publik.

- Pilih Tambahkan ruang nama ke datashare. Ruang nama adalah pengenal unik global (GUID) untuk cluster Amazon Redshift.
- Pilih Tambahkan Akun AWS ke datashare. Yang ditentukan Akun AWS harus memiliki izin akses ke datashare.

Mengotorisasi atau menghapus otorisasi dari datashare

Sebagai administrator produsen, pilih konsumen data mana yang akan disahkan untuk mengakses data share atau menghapus otorisasi. Konsumen data resmi menerima pemberitahuan untuk

mengambil tindakan pada berbagi data. Jika Anda menambahkan namespace sebagai konsumen data, Anda tidak perlu melakukan otorisasi.

Prasyarat: Untuk mengotorisasi atau menghapus otorisasi untuk datashare, setidaknya harus ada satu konsumen data yang ditambahkan ke datashare.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Datashares. Halaman daftar datashare muncul.
3. Pilih Di akun saya.
4. Di bagian Datashares di akun saya, lakukan salah satu hal berikut:
   - Pilih satu atau lebih cluster konsumen yang ingin Anda otorisasi. Halaman Otorisasi konsumen data muncul. Lalu, pilih Otorisasi.

     Jika Anda memilih Publish to AWS Glue Data Catalog saat membuat datashare, Anda hanya dapat memberikan otorisasi datashare ke akun Lake Formation.

     Untuk AWS Data Exchange datashare, Anda hanya dapat mengotorisasi satu datashare pada satu waktu.

     Ketika Anda mengotorisasi AWS Data Exchange datashare, Anda berbagi datashare dengan AWS Data Exchange layanan dan memungkinkan AWS Data Exchange untuk mengelola akses ke datashare atas nama Anda. AWS Data Exchange memungkinkan akses ke konsumen dengan menambahkan akun konsumen sebagai konsumen data ke AWS Data Exchange datashare ketika mereka berlangganan produk. AWS Data Exchange tidak memiliki akses baca ke datashare.
   - Pilih satu atau beberapa cluster konsumen yang ingin Anda hapus otorisasi. Kemudian pilih H apus otorisasi.

Setelah konsumen data diotorisasi, mereka dapat mengakses objek datashare dan membuat database konsumen untuk menanyakan data.

Setelah otorisasi dihapus, konsumen data segera kehilangan akses ke datashare.

Mengelola data share dari akun lain sebagai konsumen

Menghapus asosiasi datashare dari konsumen data

Sebagai administrator konsumen, Anda dapat menghapus asosiasi berbagi data dari konsumen data.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Datashares. Halaman daftar datashare muncul.
3. Pilih Dari akun lain.
4. Di bagian Ber bagi data dari akun lain, pilih datashare untuk menghapus as osiasi dari konsumen data.
5. Di bagian Konsumen data, pilih satu atau lebih konsumen data untuk menghapus asosiasi. Kemudian pilih H apus asosiasi.
6. Saat halaman Hapus asosiasi muncul, pilih H apus asosiasi.

Setelah asosiasi dihapus, konsumen data akan kehilangan akses ke datashare. Anda dapat mengubah asosiasi konsumen data kapan saja.

Berbagi data menurun

Sebagai administrator konsumen, Anda dapat menolak data apa pun yang statusnya tersedia atau aktif. Setelah Anda menolak datashare, pengguna cluster konsumen kehilangan akses ke datashare. Amazon Redshift tidak mengembalikan data yang ditolak jika Anda memanggil operasi API. `DescribeDataSharesForConsumer` Jika administrator produsen menjalankan operasi `DescribeDataSharesForProducer` API, mereka akan melihat bahwa datashare ditolak. Setelah datashare ditolak, administrator produsen dapat mengotorisasi datashare ke cluster konsumen lagi, dan administrator konsumen dapat memilih untuk mengaitkan AWS akun mereka dengan datashare atau menolaknya.

Jika AWS akun Anda memiliki asosiasi ke datashare dan asosiasi tertunda ke datashare yang dikelola oleh Lake Formation, menolak asosiasi datashare yang dikelola oleh Lake Formation juga menolak datashare asli. Untuk menolak asosiasi tertentu, administrator produsen dapat menghapus otorisasi dari data yang ditentukan. Tindakan ini tidak memengaruhi berbagi data lainnya.

Untuk menolak datashare, gunakan AWS konsol, operasi API`RejectDataShare`, atau `reject-datashare` di. AWS CLI

Untuk menolak datashare menggunakan AWS konsol:

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Di menu navigasi, pilih Datashares.

3. Pilih Dari akun lain.
4. Di bagian Datashares dari akun lain, pilih data yang ingin Anda tolak. Saat halaman Dec line datashare muncul, pilih Decline.

Setelah Anda menolak berbagi data, Anda tidak dapat mengembalikan perubahan. Amazon Redshift menghapus data share dari daftar. Untuk melihat datashare lagi, administrator produsen harus mengizinkannya lagi.

Mengelola data share yang ada

Dengan Amazon Redshift, Anda dapat mengelola berbagi data yang ada untuk mengontrol akses ke data Anda di cluster Amazon Redshift. Bagian berikut memberikan panduan langkah demi langkah tentang mengelola berbagi data di lingkungan Amazon Redshift Anda.

Melihat data share

Lihat berbagi data dari tab DATASHARES atau CLUSTERS.

- Gunakan tab DATASHARES untuk mencantumkan data share di akun Anda atau dari akun lain.
  - Untuk melihat data share yang dibuat di akun Anda, pilih Di akun saya, lalu pilih data yang ingin Anda lihat.
  - Untuk melihat data share yang dibagikan dari akun lain, pilih Dari akun lain, lalu pilih data yang ingin Anda lihat.
- Gunakan tab CLUSTERS untuk membuat daftar data share di cluster Anda atau dari cluster lain.

  Hubungkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.

  Kemudian pilih datashare baik dari Datashares dari cluster lain atau Datashares yang dibuat di bagian cluster saya untuk melihat detailnya.

Menghapus objek datashare dari datashares

Anda dapat menghapus satu atau lebih objek dari datashare dengan menggunakan prosedur berikut.

Untuk menghapus satu atau lebih objek dari datashare

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.

2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares.
4. Di bagian Datashares yang dibuat di akun saya, pilih Hubun gkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.
5. Pilih data yang ingin Anda edit, lalu pilih Edit. Halaman detail datashare muncul.
6. Untuk menghapus satu atau beberapa objek datashare ke datashare, lakukan salah satu hal berikut:

   - Untuk menghapus skema dari datashare, pilih satu atau beberapa skema. Kemudian pilih H apus. Amazon Redshift menghapus skema yang ditentukan dan semua objek dari skema yang ditentukan dari datashare.
   - Untuk menghapus tabel dan tampilan dari datashare, pilih satu atau lebih tabel dan tampilan. Kemudian pilih H apus. Atau, pilih Hapus berdasarkan skema untuk menghapus semua tabel dan tampilan dalam skema yang ditentukan.
   - Untuk menghapus fungsi yang ditentukan pengguna dari datashare, pilih satu atau beberapa fungsi yang ditentukan pengguna. Kemudian pilih H apus. Atau, pilih H apus berdasarkan skema untuk menghapus semua fungsi yang ditentukan pengguna dalam skema yang ditentukan.

Menghapus konsumen data dari datashares

Anda dapat menghapus satu atau lebih konsumen data dari datashare. Konsumen data dapat berupa ruang nama yang secara unik mengidentifikasi cluster atau akun Amazon Redshift. AWS

Pilih satu atau beberapa konsumen data baik dari ID namespace atau AWS akun, lalu pilih H apus.

Amazon Redshift menghapus konsumen data tertentu dari datashare. Mereka kehilangan akses ke datashare segera.

Mengedit data share yang dibuat di akun Anda

Edit data share yang dibuat di akun Anda menggunakan konsol. Hubungkan ke database terlebih dahulu untuk melihat daftar data share yang dibuat di akun Anda.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.

3. Pilih Datashares.
4. Di bagian Datashares yang dibuat di akun saya, pilih Hubungkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.
5. Pilih data yang ingin Anda edit, lalu pilih Edit. Halaman detail datashare muncul.
6. Buat perubahan apa pun di objek Datashare atau bagian Konsumen Data.

> **Note**
>
> Jika Anda memilih untuk mempublikasikan datashare ke AWS Glue Data Catalog, Anda tidak dapat mengedit konfigurasi untuk menerbitkan datashare ke akun Amazon Redshift lainnya.

7. Pilih Simpan perubahan.

Amazon Redshift memperbarui data Anda dengan perubahan.

Menghapus data yang dibuat di akun Anda

Hapus data share yang dibuat di akun Anda menggunakan konsol. Hubungkan ke database terlebih dahulu untuk melihat daftar data share yang dibuat di akun Anda.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares. Daftar datashare muncul.
4. Di bagian Datashares yang dibuat di akun saya, pilih Hubungkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.
5. Pilih satu atau beberapa data share yang ingin Anda hapus, lalu pilih Hapus. Halaman Hapus datashares muncul.

   Menghapus datashare yang dibagikan dengan Lake Formation tidak secara otomatis menghapus izin terkait di Formasi Danau. Untuk menghapusnya, buka konsol Formasi Danau.
6. Ketik H apus untuk mengonfirmasi penghapusan data share yang ditentukan.
7. Pilih Hapus.

Setelah data share dihapus, konsumen datashare kehilangan akses ke datashares.

Mengkueri berbagi data

Dengan Amazon Redshift, Anda dapat menanyakan data di seluruh data share dari cluster produsen untuk mengakses data langsung dengan aman tanpa menyalin atau mentransfernya. Bagian berikut mencakup detail pengaturan dan kueri berbagi data di lingkungan Amazon Redshift Anda.

Membuat database dari datashares

Untuk mulai menanyakan data di datashare, buat database dari datashare. Anda hanya dapat membuat satu database dari datashare tertentu.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares. Daftar datashare muncul.
4. Di bagian Ber bagi data dari cluster lain, pilih Hubun gkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.
5. Pilih datashare tempat Anda ingin membuat database, lalu pilih Buat database dari datashare. Halaman Buat database dari datashare muncul.
6. Dalam nama Database, tentukan nama database. Nama database harus berupa 1—64 karakter alfanumerik (hanya huruf kecil) dan tidak boleh menjadi kata yang dicadangkan.
7. Pilih Buat.

Setelah database dibuat, Anda dapat menanyakan data dalam database.

Mengelola AWS Data Exchange berbagi data

Dengan Amazon Redshift, Anda dapat berbagi dan menerima data langsung dengan aman AWS Data Exchange tanpa harus membuat dan mengelola ekstrak data atau pipeline. Dengan mengelola AWS Data Exchange data share, Anda dapat berlangganan produk data pihak ketiga dan mengintegrasikan aliran data langsung ke gudang data Amazon Redshift Anda. Bagian berikut menunjukkan pengelolaan berbagi AWS Data Exchange data dalam cluster Amazon Redshift Anda.

Membuat kumpulan data pada AWS Data Exchange

Buat kumpulan data aktif AWS Data Exchange.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.

2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares.
4. Di bagian Datashares yang AWS Data Exchange dibuat di bagian akun saya, pilih datashare.
5. Pilih Buat kumpulan data aktif AWS Data Exchange. Untuk informasi selengkapnya, lihat Menerbitkan produk baru.

Mengedit AWS Data Exchange berbagi data

E AWS Data Exchange dit berbagi data menggunakan konsol. Hubungkan ke database terlebih dahulu untuk melihat daftar data share yang dibuat di akun Anda.

Untuk AWS Data Exchange berbagi data, Anda tidak dapat membuat perubahan pada konsumen data.

Untuk mengedit pengaturan yang dapat diakses publik untuk AWS Data Exchange berbagi data, gunakan editor Kueri v2. Amazon Redshift menghasilkan nilai satu kali acak untuk menyetel variabel sesi agar pengaturan ini dapat dimatikan. Untuk informasi selengkapnya, lihat Catatan penggunaan ALTER DATASHARE.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Dari menu navigator, pilih Editor, lalu Query editor v2.
4. Jika ini adalah pertama kalinya Anda menggunakan editor Kueri v2, konfigurasikan Akun AWS. Secara default, kunci yang AWS dimiliki digunakan untuk mengenkripsi sumber daya. Untuk informasi selengkapnya tentang mengonfigurasi Akun AWS, lihat Mengonfigurasi Anda Akun AWS di Panduan Manajemen Amazon Redshift.
5. Untuk menyambung ke cluster tempat AWS Data Exchange datashare Anda berada, pilih Database dan nama cluster di panel tampilan pohon. Jika diminta, masukkan parameter koneksi.
6. Salin pernyataan SQL berikut. Contoh berikut mengubah pengaturan data yang dapat diakses publik dari salesshare datashare.

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
```

7. Untuk menjalankan pernyataan SQL yang disalin, pilih Kueri dan tempel pernyataan SQL yang disalin di area kueri. Kemudian pilih Run.

Kesalahan muncul sebagai berikut:

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
ERROR:  Alter of ADX-managed datashare salesshare requires session variable
 datashare_break_glass_session_var to be set to value 'c670ba4db22f4b'
```

Nilai 'c670ba4db22f4b' adalah nilai satu kali acak yang dihasilkan Amazon Redshift ketika operasi yang tidak direkomendasikan terjadi.

8. Salin dan tempel pernyataan contoh berikut ke area kueri. Kemudian jalankan perintah. Per `SET datashare_break_glass_session_var` intah menerapkan izin untuk mengizinkan operasi yang tidak direkomendasikan untuk AWS Data Exchange datashare.

```
SET datashare_break_glass_session_var to 'c670ba4db22f4b';
```

9. Jalankan pernyataan ALTER DATASHARE lagi.

```
ALTER DATASHARE salesshare;
```

Amazon Redshift memperbarui data Anda dengan perubahan.

Menghapus AWS Data Exchange data share yang dibuat di akun Anda

AWS Data Exchange Hapus data share yang dibuat di akun Anda menggunakan konsol. Hubungkan ke database terlebih dahulu untuk melihat daftar data share yang dibuat di akun Anda.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Dari menu navigator, pilih Editor, lalu Query editor v2.
4. Jika ini adalah pertama kalinya Anda menggunakan editor Kueri v2, konfigurasikan Akun AWS. Secara default, kunci yang AWS dimiliki digunakan untuk mengenkripsi sumber daya. Untuk informasi selengkapnya tentang mengonfigurasi Akun AWS, lihat Mengonfigurasi Anda Akun AWS di Panduan Manajemen Amazon Redshift.
5. Untuk menyambung ke cluster tempat AWS Data Exchange datashare Anda berada, pilih Database dan nama cluster di panel tampilan pohon. Jika diminta, masukkan parameter koneksi.
6. Salin pernyataan SQL berikut. Contoh berikut menjatuhkan data share penjualan.

```
DROP DATASHARE salesshare
```

7. Untuk menjalankan pernyataan SQL yang disalin, pilih Kueri dan tempel pernyataan SQL yang disalin di area kueri. Kemudian pilih Run.

   Kesalahan muncul sebagai berikut:

   ```
   ERROR:  Drop of ADX-managed datashare salesshare requires session variable
    datashare_break_glass_session_var to be set to value '620c871f890c49'
   ```

   Nilai '620c871f890c49' adalah nilai satu kali acak yang dihasilkan Amazon Redshift ketika operasi yang tidak direkomendasikan terjadi.

8. Salin dan tempel pernyataan contoh berikut ke area kueri. Kemudian jalankan perintah. Per `SET datashare_break_glass_session_var` intah menerapkan izin untuk mengizinkan operasi yang tidak direkomendasikan untuk AWS Data Exchange datashare.

   ```
   SET datashare_break_glass_session_var to '620c871f890c49';
   ```

9. Jalankan pernyataan DROP DATASHARE lagi.

   ```
   DROP DATASHARE salesshare;
   ```

Setelah datashare dihapus, konsumen datashare kehilangan akses ke datashare.

Menghapus AWS Data Exchange data bersama dapat melanggar ketentuan produk data di. AWS Data Exchange

Memulai dengan berbagi data read-only dengan antarmuka SQL

Dengan Amazon Redshift, Anda dapat berbagi data dengan aman di seluruh cluster Amazon Redshift, memungkinkan konsumen data untuk menanyakan dan mengakses data langsung tanpa menyalin atau mereplikasinya. Dengan berbagi data, Anda dapat membuat dan mengonfigurasi berbagi data, yang merupakan objek sisi produsen yang mereferensikan objek database yang ingin Anda bagikan.

Anda dapat berbagi data untuk tujuan membaca di berbagai cluster Amazon Redshift di dalam atau di seberang Akun AWS, atau di seluruh Wilayah AWS.

Topik

- Berbagi akses baca ke data dalam Akun AWS
- Bekerja dengan tampilan di berbagi data Amazon Redshift
- Menambahkan tabel data lake ke datashare
- Berbagi data di seluruh Akun AWS
- Berbagi data di seluruh Wilayah AWS
- Berbagi data Amazon Redshift berlisensi di AWS Data Exchange
- Memulai dengan AWS Lake Formation-berbagi data yang dikelola

Berbagi akses baca ke data dalam Akun AWS

Dengan Amazon Redshift, Anda dapat berbagi akses baca ke data di seluruh pengguna atau grup database yang berbeda dalam AWS akun yang sama. Dengan fitur ini, Anda dapat mengontrol hak akses data pada tingkat granular, memastikan bahwa hanya pengguna atau grup yang berwenang yang dapat membaca kumpulan data tertentu.

Bagikan data untuk tujuan membaca sebagai administrator produsen atau pemilik database

1. Buat berbagi data di cluster Anda. Untuk informasi selengkapnya, lihat BUAT DATASHARE.

   ```
   CREATE DATASHARE salesshare;
   ```

   Superuser cluster dan pemilik database dapat membuat datashares. Setiap datashare dikaitkan dengan database selama pembuatan. Hanya objek dari database itu yang dapat dibagikan di datashare itu. Beberapa data share dapat dibuat pada database yang sama dengan perincian objek yang sama atau berbeda. Tidak ada batasan jumlah data share yang dapat dibuat oleh cluster.

   Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat berbagi data. Untuk informasi selengkapnya, lihat Buat datashare.
2. Delegasikan izin untuk beroperasi pada datashare. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

   Contoh berikut memberikan izin untuk `dbuser` on`salesshare`.

   ```
   GRANT ALTER, SHARE ON DATASHARE salesshare TO dbuser;
   ```

Superuser cluster dan pemilik datashare dapat memberikan atau mencabut izin modifikasi pada datashare kepada pengguna tambahan.

3. Tambahkan objek ke atau hapus objek dari datashares. Untuk menambahkan objek ke datashare, tambahkan skema sebelum menambahkan objek. Saat Anda menambahkan skema, Amazon Redshift tidak menambahkan semua objek di bawahnya. Pastikan untuk menambahkan ini secara eksplisit. Untuk informasi selengkapnya, lihat ALTER DATA.

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

Anda juga dapat menambahkan tampilan ke datashare.

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
 public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD VIEW public.sales_data_summary_view;
```

Gunakan ALTER DATASHARE untuk berbagi skema, dan tabel, tampilan, dan fungsi dalam skema yang diberikan. Superuser, pemilik datashare, atau pengguna yang memiliki izin ALTER atau ALL pada datashare dapat mengubah datashare untuk menambahkan objek ke atau menghapus objek darinya. Pengguna harus memiliki izin untuk menambah atau menghapus objek dari datashare. Pengguna juga harus menjadi pemilik objek atau memiliki izin SELECT, USAGE, atau ALL pada objek.

Anda juga dapat menggunakan GRANT untuk menambahkan objek ke datashare. Contoh ini menunjukkan bagaimana:

```
GRANT SELECT ON TABLE public.tickit_sales_redshift TO DATASHARE salesshare;
```

Sintaks ini secara fungsional setara `ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;` dengan.

Gunakan klausa INCLUDENEW untuk menambahkan tabel baru, tampilan, atau fungsi yang ditentukan pengguna SQL (UDF) yang dibuat dalam skema tertentu ke datashare. Hanya pengguna super yang dapat mengubah properti ini untuk setiap pasangan datashare-schema.

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
```

```
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

Anda juga dapat menggunakan konsol Amazon Redshift untuk menambah atau menghapus objek dari berbagi data. Lihat informasi selengkapnya di Tambahkan objek datashare ke datashares, Menghapus objek datashare dari datashares, dan Mengedit data share yang dibuat di akun Anda.

4. Tambahkan konsumen ke atau hapus konsumen dari berbagi data. Contoh berikut menambahkan namespace konsumen kesalesshare. Namespace adalah namespace global unique identifier (GUID) dari cluster konsumen di akun. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

```
GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Anda hanya dapat memberikan izin kepada satu konsumen datashare dalam pernyataan GRANT.

Superuser cluster dan pemilik objek datashare atau pengguna yang memiliki izin SHARE pada datashare dapat menambahkan konsumen ke atau menghapus konsumen dari datashare. Untuk melakukannya, mereka menggunakan GRANT USAGE atau REVOKE USAGE.

Untuk menemukan namespace cluster yang saat ini Anda lihat, Anda dapat menggunakan perintah SELECT CURRENT_NAMESPACE. Untuk menemukan namespace dari cluster yang berbeda dalam klaster yang sama Akun AWS, buka halaman detail cluster konsol Amazon Redshift. Pada halaman itu, temukan bidang namespace yang baru ditambahkan.

Anda juga dapat menggunakan konsol Amazon Redshift untuk menambah atau menghapus konsumen data untuk berbagi data. Untuk informasi selengkapnya, lihat Tambahkan konsumen data ke datashares dan Menghapus konsumen data dari datashares.

5. (Opsional) Tambahkan batasan keamanan ke datashare. Contoh berikut menunjukkan bahwa cluster konsumen dengan akses IP publik diizinkan untuk membaca datashare. Untuk informasi selengkapnya, lihat ALTER DATA.

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE = TRUE;
```

Anda dapat memodifikasi properti tentang jenis konsumen setelah pembuatan datashare. Misalnya, Anda dapat menentukan bahwa cluster yang ingin menggunakan data dari datashare

tertentu tidak dapat diakses publik. Kueri dari cluster konsumen yang tidak memenuhi batasan keamanan yang ditentukan dalam datashare ditolak saat runtime kueri.

Anda juga dapat menggunakan konsol Amazon Redshift untuk mengedit data share. Untuk informasi selengkapnya, lihat Mengedit data share yang dibuat di akun Anda.

6. Buat daftar data share yang dibuat di cluster dan lihat isi datashare.

   Contoh berikut menampilkan informasi dari sebuah nama datashare. `salesshare`

```
DESC DATASHARE salesshare;

 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name          |   include_new
-------------------+--------------------------------------+------------
+------------+-------------+--------------------------------+-------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_users_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_venue_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_category_redshift|
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_date_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_event_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_listing_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_sales_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | schema      | public                         |  t
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | view        | public.sales_data_summary_view |
```

   Contoh berikut menampilkan data share keluar dalam cluster produsen.

```
SHOW DATASHARES LIKE 'sales%';
```

   Output-nya akan terlihat serupa dengan yang berikut ini.

```
share_name | share_owner  | source_database | consumer_database | share_type |
 createdate      | is_publicaccessible  | share_acl | producer_account |
 producer_namespace
-----------+--------------+-----------------+-------------------+------------
+--------------------+----------------------+-----------+-----------------
+--------------------------------------
salesshare |    100       | dev             |                   |  OUTBOUND
 | 2020-12-09 02:27:08 |          True        |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

Anda juga dapat menggunakanSVV_DATASHARES,SVV_DATASHARE_KONSUMEN, dan SVV_DATASHARE_OBJECT untuk melihat data share, objek di dalam datashare, dan konsumen datashare.

7. Jatuhkan data share. Untuk informasi selengkapnya, lihat DROP DATA.

   Anda dapat menghapus objek datashare kapan saja menggunakan. DROP DATA Superuser cluster dan pemilik datashare dapat menghapus datashares.

   Contoh berikut menjatuhkan nama datashare. `salesshare`

   ```
   DROP DATASHARE salesshare;
   ```

   Anda juga dapat menggunakan konsol Amazon Redshift untuk menghapus data share. Untuk informasi selengkapnya, lihat Menghapus data yang dibuat di akun Anda.

8. Gunakan ALTER DATASHARE untuk menghapus objek dari datashares kapan saja dari datashare. Gunakan REVOKE USAGE ON untuk mencabut izin pada datashare ke konsumen tertentu. Ini mencabut izin USAGE pada objek dalam datashare dan langsung menghentikan akses ke semua cluster konsumen. Mendaftarkan berbagi data dan kueri metadata, seperti daftar database dan tabel, tidak mengembalikan objek bersama setelah akses dicabut.

   ```
   ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
   ```

   Anda juga dapat menggunakan konsol Amazon Redshift untuk mengedit data share. Untuk informasi selengkapnya, lihat Mengedit data share yang dibuat di akun Anda.

9. Cabut akses ke datashare dari ruang nama jika Anda tidak ingin berbagi data dengan konsumen lagi.

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Anda juga dapat menggunakan konsol Amazon Redshift untuk mengedit data share. Untuk informasi selengkapnya, lihat Mengedit data share yang dibuat di akun Anda.

Bagikan data untuk tujuan membaca sebagai administrator konsumen

1. Buat daftar data share yang tersedia untuk Anda dan lihat konten dari datashares. Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

Contoh berikut menampilkan informasi dari inbound datashares dari namespace produsen tertentu. Saat Anda menjalankan DESC DATASHARE sebagai administrator konsumen, Anda harus menentukan opsi NAMESPACE untuk melihat data share masuk.

```
DESC DATASHARE salesshare OF NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';


 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name           |   include_new
-------------------+--------------------------------------+------------
+------------+-------------+---------------------------------+------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_users_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_venue_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_category_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_date_redshift     |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_event_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_listing_redshift  |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_sales_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | schema      | public                          |
```

```
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | view        | public.sales_data_summary_view  |
```

Hanya pengguna super cluster yang dapat melakukan ini. Anda juga dapat menggunakan SVV_DATASHARES untuk melihat datashares dan SVV_DATASHARE_OBJECTS untuk melihat objek di dalam datashare.

Contoh berikut menampilkan inbound datashares dalam cluster konsumen.

```
SHOW DATASHARES LIKE 'sales%';


 share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
------------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+------------------
+--------------------------------------
 salesshare |             |                 |                   | INBOUND
    |            |         t           |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

2. Sebagai superuser database, Anda dapat membuat database lokal yang merujuk ke datashares. Untuk informasi selengkapnya, lihat BUAT BASIS DATA.

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Jika Anda ingin kontrol yang lebih terperinci atas akses ke objek dalam database lokal, gunakan klausa WITH PERMISSIONS saat membuat database. Dengan klausa ini, Anda dapat memberikan izin tingkat objek untuk objek dalam database pada langkah 4.

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Anda dapat melihat database yang Anda buat dari datashare dengan menanyakan tampilan. SVV_REDSHIFT_DATABASE Anda dapat terhubung ke database ini secara langsung, atau Anda dapat terhubung ke database lokal di cluster konsumen Anda dan melakukan kueri lintas database untuk menanyakan data dari database datashare. Anda tidak dapat membuat datashare di atas objek database yang dibuat dari datashare yang ada. Namun, Anda dapat

menyalin data ke tabel terpisah di cluster konsumen, melakukan pemrosesan apa pun yang diperlukan, dan kemudian berbagi objek baru yang dibuat.

Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat database dari berbagi data. Untuk informasi selengkapnya, lihat Membuat database dari datashares.

3. (Opsional) Buat skema eksternal untuk merujuk dan menetapkan izin granular ke skema tertentu dalam database konsumen yang diimpor pada cluster konsumen. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
 'public';
```

4. Berikan izin pada database dan referensi skema yang dibuat dari berbagi data kepada pengguna dan peran dalam cluster konsumen sesuai kebutuhan. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

Jika Anda membuat database tanpa DENGAN IZIN, Anda hanya dapat menetapkan izin pada seluruh database yang dibuat dari datashare kepada pengguna dan peran Anda. Dalam beberapa kasus, Anda memerlukan kontrol halus pada subset objek database yang dibuat dari datashare. Jika demikian, Anda dapat membuat referensi skema eksternal yang menunjuk ke skema tertentu di datashare (seperti yang dijelaskan pada langkah sebelumnya) dan memberikan izin granular di tingkat skema.

Anda juga dapat membuat tampilan yang mengikat terlambat di atas objek bersama dan menggunakannya untuk menetapkan izin terperinci. Anda juga dapat mempertimbangkan untuk meminta cluster produsen membuat berbagi data tambahan untuk Anda dengan perincian yang diperlukan.

Jika Anda membuat database dengan WITH PERMISSIONS pada langkah 2, Anda harus menetapkan izin tingkat objek untuk objek dalam database bersama. Pengguna dengan hanya izin USAGE tidak dapat mengakses objek apa pun dalam database yang dibuat dengan WITH PERMISSIONS sampai mereka diberikan izin tingkat objek tambahan..

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

5. Kueri data dalam objek bersama di datashares.

   Pengguna dan peran dengan izin pada database konsumen dan skema pada cluster konsumen dapat menjelajahi dan menavigasi metadata objek bersama apa pun. Mereka juga dapat menjelajahi dan menavigasi objek lokal dalam cluster konsumen. Untuk melakukan ini, mereka menggunakan driver JDBC atau ODBC atau tampilan SVV_ALL dan SVV_REDSHIFT.

   Cluster produsen mungkin memiliki banyak skema dalam database, tabel, dan tampilan dalam setiap skema. Pengguna di sisi konsumen hanya dapat melihat subset objek yang tersedia melalui datashare. Pengguna ini tidak dapat melihat seluruh metadata dari cluster produsen. Pendekatan ini membantu menyediakan kontrol keamanan metadata granular dengan berbagi data.

   Anda terus terhubung ke database cluster lokal. Tapi sekarang, Anda juga dapat membaca dari database dan skema yang dibuat dari datashare menggunakan notasi database.schema.table tiga bagian. Anda dapat melakukan kueri yang menjangkau seluruh dan semua database yang terlihat oleh Anda. Ini bisa berupa database lokal di cluster atau database yang dibuat dari datashares. Cluster konsumen tidak dapat terhubung ke database yang dibuat dari datashares.

   Anda dapat mengakses data menggunakan kualifikasi penuh. Untuk informasi selengkapnya, lihat Cross-database contoh kueri.

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;

 salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime
---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------
       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |    728.00 |
   109.20 | 2008-02-18 02:36:48
       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |     76.00 |
    11.40 | 2008-06-06 05:00:16
       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |    350.00 |
    52.50 | 2008-06-06 08:26:17
       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |    175.00 |
    26.25 | 2008-06-09 08:38:52
```

```
      5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |    154.00 |
   23.10 | 2008-08-31 09:17:02
```

Anda hanya dapat menggunakan pernyataan SELECT pada objek bersama. Namun, Anda dapat membuat tabel di cluster konsumen dengan menanyakan data dari objek bersama di database lokal yang berbeda.

Selain kueri, konsumen dapat membuat tampilan pada objek bersama. Hanya tampilan yang mengikat terlambat atau tampilan terwujud yang didukung. Amazon Redshift tidak mendukung tampilan reguler pada data bersama. Tampilan yang dibuat konsumen dapat menjangkau beberapa database lokal atau database yang dibuat dari berbagi data. Untuk informasi selengkapnya, lihat CREATE VIEW.

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

Bekerja dengan tampilan di berbagi data Amazon Redshift

Cluster produsen dapat berbagi pandangan reguler, terlambat mengikat, dan terwujud. Saat berbagi tampilan reguler, yang mengikat terlambat, atau tampilan yang terwujud, Anda tidak perlu membagikan tabel dasar. Tabel berikut menunjukkan bagaimana tampilan didukung dengan berbagi data.

Nama tampilan	Bisakah tampilan ini ditambahkan ke datashare?	Dapatkah konsumen membuat tampilan ini pada objek datashare di seluruh cluster?	
Tampilan reguler	Ya	Tidak	
Late-binding melihat	Ya	Ya	

Nama tampilan	Bisakah tampilan ini ditambahkan ke datashare?	Dapatkah konsumen membuat tampilan ini pada objek datashare di seluruh cluster?	
Tampilan terwujud	Ya	Ya, tetapi hanya dengan penyegaran lengkap	

Kueri berikut menunjukkan output dari tampilan reguler yang didukung dengan berbagi data. Untuk informasi tentang definisi tampilan reguler, lihatCREATE VIEW.

```
SELECT * FROM tickit_db.public.myevent_regular_vw
ORDER BY eventid LIMIT 5;

   eventid  |  eventname
  ----------+-------------
     3835   | LeAnn Rimes
     3967   | LeAnn Rimes
     4856   | LeAnn Rimes
     4948   | LeAnn Rimes
     5131   | LeAnn Rimes
```

Kueri berikut menunjukkan output dari tampilan pengikatan terlambat yang didukung dengan berbagi data. Untuk informasi tentang definisi tampilan pengikatan terlambat, lihat. CREATE VIEW

```
SELECT * FROM tickit_db.public.event_lbv
ORDER BY eventid LIMIT 5;

 eventid | venueid | catid | dateid |          eventname           |      starttime
 --------+---------+-------+--------+------------------------------
+---------------------
     1   |   305   |   8   |  1851  |        Gotterdammerung       | 2008-01-25
 14:30:00
     2   |   306   |   8   |  2114  |           Boris Godunov      | 2008-10-15
 20:00:00
     3   |   302   |   8   |  1935  |              Salome          | 2008-04-19
 14:30:00
     4   |   309   |   8   |  2090  |  La Cenerentola (Cinderella) | 2008-09-21
 14:30:00
```

```
     5   |   302   |   8   |  1982  |          Il Trovatore        | 2008-06-05
 19:00:00
```

Kueri berikut menunjukkan output dari tampilan terwujud yang didukung dengan berbagi data. Untuk informasi tentang definisi tampilan terwujud, lihatBUAT TAMPILAN YANG TERWUJUD.

```
SELECT * FROM tickit_db.public.tickets_mv;

  catgroup | qtysold
 ----------+---------
  Concerts | 195444
    Shows  | 149905
```

Anda dapat memelihara tabel umum di semua penyewa dalam cluster produsen. Anda juga dapat membagikan subset data yang difilter berdasarkan kolom dimensi, seperti `tenant_id` (`account_id`data`namespace_id`), ke cluster konsumen. Untuk melakukan ini, Anda dapat menentukan tampilan pada tabel dasar dengan filter pada kolom ID ini, misalnya`current_aws_account = tenant_id`. Di sisi konsumen, saat Anda menanyakan tampilan, Anda hanya melihat baris yang memenuhi syarat untuk akun Anda. Untuk melakukan ini, Anda dapat menggunakan fungsi konteks Amazon Redshift `current_aws_account` dan`current_namespace`.

Kueri berikut mengembalikan ID akun tempat cluster Amazon Redshift saat ini berada. Anda dapat menjalankan kueri ini jika Anda terhubung ke Amazon Redshift.

```
select current_user, current_aws_account;

current_user | current_aws_account
-------------+--------------------
dwuser       |    111111111111
(1row)
```

Kueri berikut mengembalikan namespace cluster Amazon Redshift saat ini. Anda dapat menjalankan query ini jika Anda terhubung ke database.

```
select current_user, current_namespace;

current_user | current_namespace
-------------+--------------------------------------
```

```
dwuser       | 86b5169f-01dc-4a6f-9fbb-e2e24359e9a8
(1 row)
```

Penyegaran tambahan untuk tampilan terwujud dalam datashare

Amazon Redshift mendukung penyegaran inkremental untuk tampilan yang terwujud di datashare konsumen saat tabel dasar atau tampilan terwujud dibagikan. Penyegaran inkremental adalah operasi di mana Amazon Redshift mengidentifikasi perubahan pada tabel dasar atau tabel yang terjadi setelah penyegaran sebelumnya dan hanya memperbarui catatan yang sesuai dalam tampilan yang terwujud. Untuk informasi selengkapnya tentang perilaku ini, lihat MEMBUAT TAMPILAN TERMATERIALISASI.

Menambahkan tabel data lake ke datashare

Dengan datashare, produsen data dapat dengan aman berbagi objek database dengan perincian halus, seperti skema dan tabel, dengan konsumen di AWS akun yang sama atau di akun yang berbeda. Produsen juga dapat berbagi objek di seluruh Wilayah. Topik ini menjelaskan cara menambahkan objek dari danau data, khususnya, dari katalog AWS Glue data, ke datashare. Ini mencakup dua kasus penggunaan:

- Menambahkan tampilan pengikatan terlambat ke datashare yang mereferensikan tabel dari danau data — Ini nyaman bagi konsumen, karena konfigurasi awal, seperti mendefinisikan izin pada data sumber eksternal, misalnya dengan Formasi Danau, kemungkinan sudah selesai. Manfaat tambahan adalah bahwa tampilan yang ditambahkan ke datashare dapat menggabungkan tabel dari danau data dengan tabel asli Redshift.
- Menambahkan tabel dari skema eksternal ke datashare secara langsung — Ini membuat objek dari danau data tersedia untuk konsumen tanpa lapisan atau logika tambahan. Konsumen dapat menanyakan tabel atau menggabungkannya dengan tabel pada konsumen.

Kasus-kasus ini berlaku setelah Anda mereferensikan tabel dari katalog AWS data di Redshift menggunakan CREATE EXTERNAL SCHEMA. Setiap tabel dari katalog AWS data dapat menjadi sumbernya.

> **Note**
>
> Tabel danau data yang Anda tambahkan ke datashare dapat mencakup tabel yang terdaftar dengan Lake Formation dan tabel katalog AWS Glue data.

Membuat skema eksternal dan tabel eksternal

Anda membuat skema eksternal dan tabel eksternal untuk menambahkannya ke datashare di bagian berikut. Ini adalah langkah awal. Jika Anda sudah melakukan ini, Anda dapat melewati bagian ini.

1. Pada produsen, buat skema eksternal yang mereferensikan data danau data yang disimpan di Amazon S3. Skema eksternal mereferensi AWS kan Katalog Data Glue. Peran dan wilayah dalam sampel adalah contoh:

```
CREATE EXTERNAL SCHEMA external_schema_name FROM DATA CATALOG
DATABASE 'glue_database_name'
IAM_ROLE 'arn:aws:iam::123456789012:role/sample-role'
REGION 'us-east-1';
```

2. Buat tabel data lake dalam skema eksternal.

```
CREATE EXTERNAL TABLE external_schema_name.sales(
salesid INTEGER,
sellerid INTEGER,
buyerid INTEGER,
saledate DATE,
pricepaid DECIMAL(8,2))
ROW FORMAT delimited
FIELDS TERMINATED BY '\t'
STORED AS textfile
LOCATION 's3://redshift-downloads/tickit/spectrum/sales/';
```

Sampel termasuk`LOCATION`. Itu harus dalam formulir`s3://{bucket_name}/{folder}/`, di mana folder ditentukan. Folder harus memiliki panjang setidaknya satu karakter. Anda dapat secara opsional menyertakan subfolder. Untuk melihat contoh tambahan untuk membuat tabel di danau data, lihat Contoh untuk MEMBUAT TABEL EKSTERNAL.

> **Note**
>
> Berbagi hanya didukung untuk tabel di mana peran IAM pada produsen memiliki akses SELECT pada tabel.

Tambahkan tampilan pengikatan terlambat yang mereferensikan tabel danau data ke datashare

Saat Anda membuat tabel berdasarkan skema eksternal dari katalog AWS data, dan Anda ingin menambahkannya ke datashare, cara paling umum untuk melakukannya adalah dengan menambahkan tampilan pengikatan terlambat Redshift yang mereferensikan tabel yang Anda buat, yang berisi data dari danau data. Prosedur berikut menunjukkan langkah-langkahnya:

1. Buat tampilan pengikatan terlambat yang mereferensikan tabel eksternal yang Anda buat sebelumnya:

```
CREATE VIEW lbv AS
select * from external_schema_name.sales, other_schema.t1
WITH NO SCHEMA BINDING;
```

2. Tambahkan skema tampilan ke datashare. Ini adalah skema lokal yang berisi tampilan pengikatan terlambat.

```
ALTER DATASHARE dsx_datashare ADD SCHEMA public;
```

3. Tambahkan skema yang berisi tabel yang direferensikan oleh tampilan pengikatan terlambat ke datashare. Menambahkan skema diperlukan untuk setiap tabel dasar yang direferensikan dalam tampilan yang ditambahkan ke datashare, apakah skema berisi objek database lokal atau objek dari danau data. Perhatikan bahwa Anda harus menambahkan skema ini sebelum menambahkan tampilan pengikatan terlambat.

```
ALTER DATASHARE dsx_datashare ADD SCHEMA external_schema_name;
ALTER DATASHARE dsx_datashare ADD SCHEMA other_schema;
```

4. Tambahkan tampilan ke datashare, menggunakan perintah SQL. Perhatikan bahwa nama tabel menyertakan awalan skema.

```
ALTER DATASHARE my_datashare ADD TABLE public.lbv;
```

5. Konfirmasikan bahwa tampilan dan skema berhasil ditambahkan ke datashare:

```
SELECT * FROM svv_datashare_objects WHERE share_name = 'my_datashare';
```

6. Administrator konsumen membuat database dari datashare dan kemudian memberikan penggunaan kepada pengguna konsumen.

Setelah Anda menyelesaikan langkah-langkahnya, pengguna konsumen database dengan akses ke tampilan datashare dapat menanyakan data.

Tambahkan tabel data lake langsung ke datashare

Menambahkan tabel dalam skema eksternal ke datashare mirip dengan menambahkan tampilan. Ini berfungsi dengan baik untuk kasus di mana konsumen ingin menanyakan tabel data lake dalam keadaan semula, atau jika konsumen ingin menggabungkannya ke tabel di gudang data konsumen. Langkah-langkah berikut menunjukkan cara menambahkan tabel data lake ke datashare, menggunakan SQL.

1. Buat skema eksternal dan tabel eksternal, seperti yang dijelaskan di bagian pertama topik ini.
2. Temukan tabel yang ada dalam skema eksternal, untuk mengonfirmasi bahwa tabel yang Anda buat tersedia:

   ```
   SELECT * FROM svv_external_tables WHERE schemaname = 'external_schema_name';
   ```

3. Tambahkan skema eksternal ke datashare:

   ```
   ALTER DATASHARE my_datashare ADD SCHEMA external_schema_name;
   ```

4. Tambahkan tabel eksternal ke datashare. Perhatikan bahwa nama tabel menyertakan awalan skema:

   ```
   ALTER DATASHARE my_datashare ADD TABLE external_schema_name.sales;
   ```

5. Konfirmasikan bahwa tabel berhasil ditambahkan ke datashare:

   ```
   SELECT * FROM svv_datashare_objects WHERE share_name = 'my_datashare';
   ```

   Untuk instruksi lebih rinci, lihat Berbagi akses baca ke data dalam AWS akun.

6. Pada konsumen, yang merupakan database yang menerima data bersama, administrator mengaitkan datashare untuk membuat tabel bersama tersedia bagi pengguna untuk kueri. Untuk informasi selengkapnya mengenai cara melakukan langkah ini, lihat M engelola data share dari akun lain sebagai konsumen.

Setelah administrator menyelesaikan langkah-langkahnya, pengguna database pada konsumen dapat menulis kueri untuk mengambil data dari tabel bersama dan menggabungkannya dengan tabel lain pada konsumen.

Catatan penggunaan untuk menambahkan objek danau data ke datashare

Ada beberapa item yang perlu diperhatikan saat Anda menggunakan tabel dan tampilan dari data lake di datashare:

- Login dengan AWS CloudTrail - Akun produsen data dapat menggunakan AWS CloudTrail log untuk mengaudit ketika tabel data lake yang dibagikan melalui datashare diakses:
  - Menggunakan data log untuk mengontrol akses data — CloudTrail Log mencatat detail tentang siapa yang mengakses tabel bersama, termasuk produsen dan konsumen datashare Redshift. Pengidentifikasi tersedia di `ExternalId` bidang di bawah `AssumeRole` CloudTrail log. Pemilik data dapat mengonfigurasi batasan tambahan pada akses data dalam kebijakan IAM melalui tindakan. Untuk informasi selengkapnya tentang menentukan akses data melalui kebijakan, lihat Akses ke AWS akun yang dimiliki oleh pihak ketiga.
- Keamanan dan izin konsumen — Untuk tabel terdaftar Danau Formation, sumber daya Amazon S3 diamankan oleh Lake Formation dan tersedia menggunakan kredenSIAL yang disediakan oleh Lake Formation.

Pertimbangan penagihan untuk menambahkan objek danau data ke datashare

Berikut ini merinci bagaimana biaya dikaitkan untuk menyimpan dan memindai objek danau data di datashare:

- Ketika konsumen menanyakan objek bersama dari danau data, biaya pemindaian ditagih kepada konsumen.
  - Jika konsumen adalah cluster yang disediakan RA3 atau DC2, Redshift menggunakan Redshift Spectrum untuk memindai data Amazon S3. Oleh karena itu, biaya Spectrum ditagih ke akun konsumen.
  - Ketika konsumen adalah cluster yang disediakan RG, tidak ada biaya terpisah untuk Spectrum karena kueri data lake berjalan pada komputasi cluster yang ditagih dengan harga komputasi jam node RG.
  - Ketika konsumen adalah kelompok kerja Amazon Redshift Serverless, tidak ada biaya terpisah untuk Spectrum.

- Biaya Amazon S3 untuk penyimpanan dan operasi, seperti bucket listing, ditagih ke akun yang memiliki setiap bucket Amazon S3.

Untuk detail tambahan mengenai penagihan untuk Amazon Redshift Serverless, lihat Pen agihan untuk Amazon Redshift Serverless. Informasi penagihan dan harga lainnya tersedia di harga https://aws.amazon.com/redshift/pricing/ Amazon Redshift.

Berbagi data di seluruh Akun AWS

Anda dapat berbagi data untuk tujuan membaca di seluruh dunia Akun AWS. Berbagi data Akun AWS lintas berfungsi mirip dengan berbagi data dalam akun. Perbedaannya adalah bahwa ada jabat tangan dua arah yang diperlukan dalam berbagi data. Akun AWS Administrator akun produsen dapat mengotorisasi akun konsumen untuk mengakses data share atau memilih untuk tidak mengotorisasi akses apa pun. Untuk menggunakan datashare resmi, administrator akun konsumen dapat mengaitkan datashare. Administrator dapat mengaitkan datashare dengan keseluruhan Akun AWS atau dengan cluster tertentu di akun konsumen, atau menolak datashare. Untuk informasi selengkapnya tentang berbagi data dalam akun, lihatBerbagi akses baca ke data dalam Akun AWS.

Datashare dapat memiliki konsumen data yang merupakan ruang nama di akun yang sama atau berbeda. Akun AWS Anda tidak perlu membuat berbagi data terpisah untuk berbagi dalam akun dan berbagi lintas akun.

Untuk berbagi data lintas akun, klaster produsen dan konsumen harus dienkripsi.

Saat berbagi data dengan Akun AWS, administrator produsen berbagi dengan Akun AWS sebagai entitas. Administrator konsumen dapat memutuskan ruang nama mana di akun konsumen yang mendapatkan akses ke datashare.

Topik

- tindakan administrator produsen
- Tindakan administrator akun konsumen
- tindakan administrator konsumen

tindakan administrator produsen

Dengan Amazon Redshift, Anda dapat melakukan tugas administratif pada cluster produsen untuk mengelola konsumsi data dan pemrosesan beban.

Jika Anda adalah administrator produsen atau pemilik database - ikuti langkah-langkah ini:

1. Buat berbagi data di cluster Anda dan tambahkan objek datashare ke datashares. Untuk langkah-langkah lebih rinci tentang cara membuat datashares dan menambahkan objek datashare ke datashares, lihat. Berbagi akses baca ke data dalam Akun AWS Untuk informasi tentang CREATE DATASHARE dan ALTER DATASHARE, lihat dan. BUAT DATASHARE ALTER DATA

   Contoh berikut menambahkan objek datashare yang berbeda ke datashare. `salesshare`

   ```
   -- Add schema to datashare
   ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;

   -- Add table under schema to datashare
   ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;

   -- Add view to datashare
   ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;

   -- Add all existing tables and views under schema to datashare (does not include
    future table)
   ALTER DATASHARE salesshare ADD ALL TABLES in schema public;
   ```

   Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat atau mengedit data share. Untuk informasi selengkapnya, lihat Buat datashare dan Mengedit data share yang dibuat di akun Anda.

2. Delegasikan izin untuk beroperasi pada datashare. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

   Contoh berikut memberikan izin untuk `dbuser` on`salesshare`.

   ```
   GRANT ALTER, SHARE ON DATASHARE salesshare TO dbuser;
   ```

   Superuser cluster dan pemilik datashare dapat memberikan atau mencabut izin modifikasi pada datashare kepada pengguna tambahan.

3. Tambahkan konsumen ke atau hapus konsumen dari berbagi data. Contoh berikut menambahkan Akun AWS ID ke`salesshare`. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

   ```
   GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012';
   ```

Anda hanya dapat memberikan izin kepada satu konsumen data dalam pernyataan GRANT.

Superuser cluster dan pemilik objek datashare, atau pengguna yang memiliki izin SHARE pada datashare, dapat menambahkan konsumen ke atau menghapus konsumen dari datashare. Untuk melakukannya, mereka menggunakan GRANT USAGE atau REVOKE USAGE.

Anda juga dapat menggunakan konsol Amazon Redshift untuk menambah atau menghapus konsumen data untuk berbagi data. Untuk informasi selengkapnya, lihat Tambahkan konsumen data ke datashares dan Menghapus konsumen data dari datashares.

4. (Opsional) Cabut akses ke datashare Akun AWS jika Anda tidak ingin berbagi data dengan konsumen lagi.

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012';
```

Jika Anda adalah administrator akun produsen - ikuti langkah-langkah ini:

Setelah memberikan penggunaan ke Akun AWS, status datashare adalah. `pending_authorization` Administrator akun produsen harus mengotorisasi berbagi data menggunakan konsol Amazon Redshift dan memilih konsumen data.

Masuk ke https://console.aws.amazon.com/redshiftv2/. Kemudian pilih konsumen data mana yang akan diotorisasi untuk mengakses data share atau menghapus otorisasi. Konsumen data resmi menerima pemberitahuan untuk mengambil tindakan pada berbagi data. Jika Anda menambahkan namespace sebagai konsumen data, Anda tidak perlu melakukan otorisasi. Setelah konsumen data diotorisasi, mereka dapat mengakses objek datashare dan membuat database konsumen untuk menanyakan data. Untuk informasi selengkapnya, lihat Mengotorisasi atau menghapus otorisasi dari datashare.

Berbagi izin menulis ke data di seluruh akun

Dengan Amazon Redshift, Anda dapat berbagi data di seluruh AWS akun dan memberikan izin menulis, memungkinkan kolaborasi dan berbagi data antar tim atau organisasi. Dengan berbagi data lintas akun, Anda dapat membuat akun penyedia data yang membuat dan mengelola database, skema, dan tabel, yang kemudian dapat dibagikan dengan aman dengan akun konsumen data. Bagian berikut menunjukkan proses mengonfigurasi berbagi data lintas akun dan memberikan akses tulis di Amazon Redshift.

Tindakan administrator akun konsumen

Dengan Amazon Redshift, Anda dapat mengelola akun konsumen dan mengontrol akses mereka ke sumber daya pergudangan data Anda.

Jika Anda adalah administrator akun konsumen - ikuti langkah-langkah ini:

Untuk mengaitkan satu atau lebih berbagi data yang dibagikan dari akun lain dengan seluruh ruang nama Akun AWS atau ruang nama tertentu di akun Anda, gunakan konsol Amazon Redshift.

Masuk ke https://console.aws.amazon.com/redshiftv2/. Kemudian, kaitkan satu atau beberapa data share yang dibagikan dari akun lain dengan seluruh Akun AWS atau ruang nama tertentu di akun Anda. Untuk informasi selengkapnya, lihat Mengaitkan datashare dari yang berbeda Akun AWS di Amazon Redshift.

Setelah ruang Akun AWS nama atau tertentu dikaitkan, berbagi data menjadi tersedia untuk dikonsumsi. Anda juga dapat mengubah asosiasi datashare kapan saja. Saat mengubah asosiasi dari ruang nama individual menjadi sebuah Akun AWS, Amazon Redshift mengganti ruang nama dengan informasi tersebut. Akun AWS Semua ruang nama di akun mendapatkan akses ke data, termasuk ruang nama yang dibuat setelah asosiasi. Saat mengubah asosiasi dari Akun AWS ruang nama ke ruang nama tertentu, Amazon Redshift mengganti Akun AWS informasi dengan informasi namespace. Ruang nama yang memiliki akses sebelumnya akan kehilangan akses jika tidak ditentukan dalam asosiasi baru.

tindakan administrator konsumen

Dengan Amazon Redshift, Anda dapat melakukan tugas administratif pada cluster konsumen untuk mengelola konsumsi data dan pemrosesan beban.

Jika Anda seorang administrator konsumen - ikuti langkah-langkah ini:

1. Buat daftar data share yang tersedia untuk Anda dan lihat konten dari datashares. Konten dari datashares hanya tersedia ketika administrator produsen telah mengizinkan berbagi data dan administrator konsumen telah menerima dan mengaitkan berbagi data. Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

   Contoh berikut menampilkan informasi dari inbound datashares dari namespace produsen tertentu. Saat menjalankan DESC DATAHSARE sebagai administrator konsumen, Anda harus menentukan NAMESPACE dan ID akun untuk melihat data share masuk. Untuk berbagi data keluar, tentukan nama datashare.

```
SHOW DATASHARES LIKE 'sales%';

share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
-----------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+-----------------
+--------------------------------------
salesshare |             |                 |                   | INBOUND    |
            |        t            |           | 123456789012    | 'dd8772e1-
d792-4fa4-996b-1870577efc0d'
```

```
DESC DATASHARE salesshare OF ACCOUNT '123456789012' NAMESPACE 'dd8772e1-
d792-4fa4-996b-1870577efc0d';


 producer_account |          producer_namespace          | share_type | share_name |
 object_type |           object_name
------------------+--------------------------------------+------------+------------
+-------------+---------------------------------
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_users_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_venue_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_category_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_date_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_event_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_listing_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_sales_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 schema      | public
(8 rows)
```

Hanya pengguna super cluster yang dapat melakukan ini. Anda juga dapat menggunakan SVV_DATASHARES untuk melihat datashares dan SVV_DATASHARE_OBJECTS untuk melihat objek di dalam datashare.

Contoh berikut menampilkan inbound datashares dalam cluster konsumen.

```
SELECT * FROM SVV_DATASHARES WHERE share_name LIKE 'sales%';

share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
-----------+-------------+-----------------+-------------------+-----------
+------------+---------------------+-----------+------------------
+--------------------------------------
salesshare |             |                 |                   | INBOUND    |
            |        t            |           | 123456789012      | 'dd8772e1-
d792-4fa4-996b-1870577efc0d'
```

```
SELECT * FROM SVV_DATASHARE_OBJECTS WHERE share_name LIKE 'sales%';
 share_type | share_name | object_type |           object_name           |
 producer_account |          producer_namespace
------------+------------+-------------+---------------------------------
+------------------+--------------------------------------
 INBOUND    | salesshare | table       | public.tickit_users_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_venue_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_category_redshift |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_date_redshift     |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_event_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_listing_redshift  |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_sales_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | schema      | public                          |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
(8 rows)
```

2. Buat database lokal yang merujuk ke datashares. Tentukan NAMESPACE dan ID akun saat membuat database dari datashare. Untuk informasi selengkapnya, lihat BUAT BASIS DATA.

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF ACCOUNT '123456789012'
 NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

Jika Anda ingin kontrol yang lebih terperinci atas akses ke objek dalam database lokal, gunakan klausa WITH PERMISSIONS saat membuat database. Dengan klausa ini, Anda dapat memberikan izin tingkat objek untuk objek dalam database pada langkah 4.

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF ACCOUNT
 '123456789012' NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

Anda dapat melihat database yang Anda buat dari datashare dengan kueri tampilan. SVV_REDSHIFT_DATABASE Anda dapat terhubung ke database ini secara langsung, atau Anda dapat terhubung ke database lokal di cluster konsumen Anda dan melakukan kueri lintas database untuk menanyakan data dari database datashare. Anda tidak dapat membuat datashare di atas objek database yang dibuat dari datashare yang ada. Namun, Anda dapat menyalin data ke tabel terpisah di cluster konsumen, melakukan pemrosesan apa pun yang diperlukan, dan kemudian berbagi objek baru yang dibuat.

3. (Opsional) Buat skema eksternal untuk merujuk dan menetapkan izin granular ke skema tertentu dalam database konsumen yang diimpor pada cluster konsumen. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
 'public';
```

4. Berikan izin pada database dan referensi skema yang dibuat dari berbagi data kepada pengguna atau peran dalam cluster konsumen sesuai kebutuhan. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

Jika Anda membuat database tanpa DENGAN IZIN, Anda hanya dapat menetapkan izin pada seluruh database yang dibuat dari datashare kepada pengguna atau peran Anda. Dalam beberapa kasus, Anda memerlukan kontrol halus pada subset objek database yang dibuat dari datashare. Jika demikian, Anda dapat membuat referensi skema eksternal yang menunjuk ke skema

tertentu di datashare, seperti yang dijelaskan pada langkah sebelumnya. Anda kemudian dapat memberikan izin granular di tingkat skema. Anda juga dapat membuat tampilan pengikatan terlambat di atas objek bersama dan menggunakannya untuk menetapkan izin granular. Anda juga dapat mempertimbangkan untuk meminta cluster produsen membuat berbagi data tambahan untuk Anda dengan perincian yang diperlukan. Anda dapat membuat referensi skema sebanyak mungkin ke database yang dibuat dari datashare yang Anda butuhkan.

Jika Anda membuat database dengan WITH PERMISSIONS pada langkah 2, Anda harus menetapkan izin tingkat objek untuk objek dalam database bersama. Pengguna dengan hanya izin USAGE tidak dapat mengakses objek apa pun dalam database yang dibuat dengan WITH PERMISSIONS sampai mereka diberikan izin tingkat objek tambahan..

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

5. Kueri data dalam objek bersama di datashares.

   Pengguna dan peran dengan izin pada database konsumen dan skema pada cluster konsumen dapat menjelajahi dan menavigasi metadata objek bersama apa pun. Mereka juga dapat menjelajahi dan menavigasi objek lokal dalam cluster konsumen. Untuk melakukan ini, gunakan driver JDBC atau ODBC atau tampilan SVV_ALL dan SVV_REDSHIFT.

   Cluster produsen mungkin memiliki banyak skema dalam database, tabel, dan tampilan dalam setiap skema. Pengguna di sisi konsumen hanya dapat melihat subset objek yang tersedia melalui datashare. Pengguna ini tidak dapat melihat semua metadata dari cluster produsen. Pendekatan ini membantu menyediakan kontrol keamanan metadata granular dengan berbagi data.

   Anda terus terhubung ke database cluster lokal. Tapi sekarang, Anda juga dapat membaca dari database dan skema yang dibuat dari datashare menggunakan notasi database.schema.table tiga bagian. Anda dapat melakukan kueri yang menjangkau seluruh dan semua database yang terlihat oleh Anda. Ini bisa berupa database lokal di cluster atau database yang dibuat dari datashares. Cluster konsumen tidak dapat terhubung ke database yang dibuat dari datashares.

   Anda dapat mengakses data menggunakan kualifikasi penuh. Untuk informasi selengkapnya, lihat Cross-database contoh kueri.

   ```
   SELECT * FROM sales_db.public.tickit_sales_redshift;
   ```

Anda hanya dapat menggunakan pernyataan SELECT pada objek bersama. Namun, Anda dapat membuat tabel di cluster konsumen dengan menanyakan data dari objek bersama di database lokal yang berbeda.

Selain melakukan kueri, konsumen dapat membuat tampilan pada objek bersama. Hanya tampilan yang mengikat terlambat dan tampilan terwujud yang didukung. Amazon Redshift tidak mendukung tampilan reguler pada data bersama. Tampilan yang dibuat konsumen dapat menjangkau beberapa database lokal atau database yang dibuat dari berbagi data. Untuk informasi selengkapnya, lihat CREATE VIEW.

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

Berbagi data di seluruh Wilayah AWS

Anda dapat berbagi data untuk tujuan membaca di seluruh cluster Amazon Redshift di Wilayah AWS. Dengan berbagi data lintas wilayah, Anda dapat berbagi data Wilayah AWS tanpa perlu menyalin data secara manual. Anda tidak perlu membongkar data ke Amazon S3 dan menyalin data ke cluster Amazon Redshift baru atau melakukan salinan snapshot lintas wilayah.

Dengan berbagi data lintas wilayah, Anda dapat berbagi data di seluruh cluster yang sama Akun AWS, atau berbeda Akun AWS bahkan ketika cluster berada di Wilayah yang berbeda. Saat berbagi data dengan cluster Amazon Redshift yang sama Akun AWS tetapi berbeda Wilayah AWS, ikuti alur kerja yang sama seperti berbagi data dalam. Akun AWS Untuk informasi selengkapnya, lihat Berbagi akses baca ke data dalam Akun AWS.

Jika cluster berbagi data berbeda Akun AWS dan Wilayah AWS, Anda dapat mengikuti alur kerja yang sama seperti berbagi data di seluruh Akun AWS dan menyertakan Region-level asosiasi pada cluster konsumen. Cross-Region berbagi data mendukung asosiasi datashare dengan seluruh Akun AWS, seluruh Wilayah AWS, atau ruang nama tertentu dalam. Wilayah AWS Untuk informasi selengkapnya tentang berbagi data Akun AWS, lihatBerbagi data di seluruh Akun AWS.

Saat mengonsumsi data dari Wilayah yang berbeda, konsumen membayar biaya transfer Cross-Region data dari Wilayah produsen ke Wilayah konsumen.

Untuk menggunakan datashare, administrator akun konsumen dapat mengaitkan datashare dengan salah satu dari tiga cara berikut.

- Asosiasi dengan keseluruhan Akun AWS yang mencakup semua Wilayah AWS
- Asosiasi dengan yang spesifik Wilayah AWS dalam Akun AWS
- Asosiasi dengan ruang nama tertentu dalam Wilayah AWS

Ketika administrator memilih keseluruhan Akun AWS, semua ruang nama yang ada dan yang akan datang Wilayah AWS di berbagai akun memiliki akses ke berbagi data. Administrator akun konsumen juga dapat memilih ruang Wilayah AWS nama tertentu atau dalam Wilayah untuk memberi mereka akses ke berbagi data.

Jika Anda adalah administrator produsen atau pemilik database, buat datashare, tambahkan objek database dan konsumen data ke datashare, dan berikan izin kepada konsumen data. Untuk informasi selengkapnya, lihat tindakan administrator produsen.

Jika Anda adalah administrator akun produsen, otorisasi berbagi data menggunakan AWS Command Line Interface (AWS CLI) atau konsol Amazon Redshift dan pilih konsumen data.

Jika Anda adalah administrator akun konsumen - ikuti langkah-langkah ini:

Untuk mengaitkan satu atau lebih berbagi data yang dibagikan dari akun lain ke seluruh Akun AWS atau ruang nama spesifik Wilayah AWS Anda dalam Wilayah AWS, gunakan konsol Amazon Redshift.

Dengan berbagi data lintas wilayah, Anda dapat menambahkan cluster di suatu kelompok tertentu Wilayah AWS menggunakan AWS Command Line Interface (AWS CLI) atau konsol Amazon Redshift.

Untuk menentukan satu atau lebih AWS Wilayah, Anda dapat menggunakan perintah `associate-data-share-consumer` CLI dengan `consumer-region` opsi opsional.

Dengan CLI, contoh berikut mengaitkan `Salesshare` dengan keseluruhan Akun AWS dengan `associate-entire-account` opsi. Anda hanya dapat mengasosiasikan satu Wilayah pada satu waktu.

```
aws redshift associate-data-share-consumer
```

```
--region {PRODUCER_REGION}
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--associate-entire-account
```

Contoh berikut mengaitkan `Salesshare` dengan Wilayah Timur AS (Ohio) (`us-east-2`).

```
aws redshift associate-data-share-consumer
--region {PRODUCER_REGION}
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:0123456789012:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--consumer-region 'us-east-2'
```

Contoh berikut mengait `Salesshare` kan dengan namespace konsumen tertentu Akun AWS di wilayah Asia Pasifik (Sydney) (`ap-southeast-2`).

```
aws redshift associate-data-share-consumer
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--consumer-arn 'arn:aws:redshift:ap-southeast-2:{CONSUMER_ACCOUNT}:namespace:
{ConsumerImmutableClusterId}'
```

Anda dapat menggunakan konsol Amazon Redshift untuk mengaitkan berbagi data dengan seluruh Akun AWS atau spesifik Wilayah AWS atau ruang nama Anda dalam. Wilayah AWS Untuk melakukan ini, masuk ke https://console.aws.amazon.com/redshiftv2/. Kemudian kaitkan satu atau lebih data share yang dibagikan dari akun lain dengan seluruh Akun AWS, seluruh Wilayah AWS, atau namespace tertentu di dalam. Wilayah AWS Untuk informasi selengkapnya, lihat Mengaitkan datashare dari yang berbeda Akun AWS di Amazon Redshift.

Setelah ruang Akun AWS nama atau tertentu dikaitkan, berbagi data menjadi tersedia untuk dikonsumsi. Anda juga dapat mengubah asosiasi datashare kapan saja. Saat mengubah asosiasi dari ruang nama individual menjadi sebuah Akun AWS, Amazon Redshift mengganti ruang nama dengan informasi tersebut. Akun AWS Saat mengubah asosiasi dari Akun AWS ruang nama ke ruang nama tertentu, Amazon Redshift mengganti Akun AWS informasi dengan informasi namespace. Saat mengubah asosiasi dari keseluruhan Akun AWS ke Wil AWS ayah dan ruang nama tertentu, Amazon Redshift mengganti Akun AWS informasi dengan informasi Wilayah dan namespace tertentu.

Jika Anda adalah administrator konsumen, Anda dapat membuat database lokal yang mereferensikan ke berbagi data dan memberikan izin pada database yang dibuat dari berbagi data

kepada pengguna atau peran dalam cluster konsumen sesuai kebutuhan. Anda juga dapat membuat tampilan pada objek bersama dan membuat skema eksternal untuk merujuk dan menetapkan izin granular ke skema tertentu dalam database konsumen yang diimpor pada cluster konsumen. Untuk informasi selengkapnya, lihat tindakan administrator konsumen.

Mengelola kontrol biaya untuk berbagi data lintas wilayah

Dengan Amazon Redshift, Anda dapat mengelola kontrol biaya untuk berbagi data lintas wilayah dengan mengonfigurasi berbagi data untuk membatasi jumlah data yang ditransfer antar AWS Wilayah. Dengan mengelola kontrol biaya untuk berbagi data lintas wilayah, Anda dapat menetapkan batas transfer data, memantau penggunaan transfer data, dan menerima pemberitahuan saat mendekati atau melampaui batas tersebut.

Saat mengonsumsi data dari Wilayah yang berbeda, konsumen membayar biaya transfer Cross-Region data dari Wilayah produsen ke Wilayah konsumen. Harga transfer data berbeda untuk Wilayah yang berbeda. Biaya didasarkan pada byte data yang dipindai untuk setiap kueri yang berhasil dijalankan. Untuk informasi selengkapnya tentang harga Amazon Redshift, lihat harga Amazon Redshift.

Anda dikenakan biaya untuk jumlah byte, dibulatkan ke megabyte berikutnya, dengan minimum 10MB per kueri. Anda dapat mengatur kontrol biaya pada penggunaan kueri dan melihat jumlah data yang ditransfer per kueri pada cluster Anda.

Untuk memantau dan mengontrol penggunaan dan biaya terkait penggunaan berbagi data lintas wilayah, Anda dapat membuat batas penggunaan harian, mingguan, bulanan, dan menentukan tindakan yang dilakukan Amazon Redshift secara otomatis jika batas tersebut tercapai untuk membantu menjaga anggaran Anda dengan mudah diprediksi.

Bergantung pada batas penggunaan yang Anda tetapkan, tindakan yang dilakukan Amazon Redshift dapat berupa mencatat peristiwa ke tabel sistem, mengirim CloudWatch alarm, dan memberi tahu administrator dengan Amazon SNS, atau mematikan berbagi data lintas wilayah untuk penggunaan lebih lanjut.

Untuk membuat batas penggunaan di konsol Amazon Redshift, pilih Konfigurasikan batas penggunaan di bawah Tind akan untuk cluster Anda. Anda dapat memantau tren penggunaan dan mendapatkan peringatan tentang penggunaan yang melebihi batas yang ditentukan dengan CloudWatch metrik yang dibuat secara otomatis dari tab Kin erja Cluster atau Monitoring. Anda juga dapat membuat, memodifikasi, dan menghapus batas penggunaan secara terprogram dengan menggunakan operasi API AWS CLI atau Amazon Redshift.

Berbagi data Amazon Redshift berlisensi di AWS Data Exchange

Saat membuat data AWS Data Exchange share dan menambahkannya ke AWS Data Exchange produk, penyedia dapat melisensikan data di Amazon Redshift sehingga konsumen dapat menemukan, berlangganan, dan menanyakan data terbaru di Amazon Redshift ketika mereka memiliki langganan aktif. AWS Data Exchange

Dengan AWS Data Exchange data share yang ditambahkan ke AWS Data Exchange produk, konsumen secara otomatis memiliki akses ke data share produk ketika langganan mereka dimulai dan mempertahankan akses mereka selama langganan mereka aktif.

Topik

- Bekerja dengan AWS Data Exchange datashares sebagai produser
- Bekerja dengan AWS Data Exchange berbagi data sebagai konsumen

Bekerja dengan AWS Data Exchange datashares sebagai produser

Dengan Amazon Redshift, Anda dapat berbagi produk data langsung dengan AWS Data Exchange sebagai produsen dengan membuat dan mengelola berbagi data.

Jika Anda adalah administrator produsen, ikuti langkah-langkah berikut untuk mengelola berbagi AWS Data Exchange data di konsol Amazon Redshift:

1. Buat berbagi data di cluster Anda untuk berbagi data AWS Data Exchange dan memberikan akses AWS Data Exchange ke data share.

   Superuser cluster dan pemilik database dapat membuat datashares. Setiap datashare dikaitkan dengan database selama pembuatan. Hanya objek dari database itu yang dapat dibagikan di datashare itu. Beberapa data share dapat dibuat pada database yang sama dengan perincian objek yang sama atau berbeda. Tidak ada batasan jumlah data share yang dapat Anda buat di cluster.

   Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat berbagi data. Untuk informasi selengkapnya, lihat Buat datashare.

   Gunakan opsi MANAGEDBY ADX untuk secara implisit memberikan akses ke datashare AWS Data Exchange saat menjalankan pernyataan CREATE DATASHARE. Ini menunjukkan bahwa AWS Data Exchange mengelola datashare ini. Anda hanya dapat menggunakan opsi MANAGEDBY ADX saat Anda membuat datashare baru. Anda tidak dapat menggunakan

pernyataan ALTER DATASHARE untuk memodifikasi datashare yang ada untuk menambahkan opsi MANAGEDBY ADX. Setelah datashare dibuat dengan opsi MANAGEDBY ADX, hanya AWS Data Exchange dapat mengakses dan mengelola datashare.

```
CREATE DATASHARE salesshare
[[SET] MANAGEDBY [=] {ADX} ];
```

2. Tambahkan objek ke datashares. Administrator produser terus mengelola objek datashare yang tersedia di datashare AWS Data Exchange .

   Untuk menambahkan objek ke datashare, tambahkan skema sebelum menambahkan objek. Saat Anda menambahkan skema, Amazon Redshift tidak menambahkan semua objek di bawahnya. Anda harus menambahkannya secara eksplisit. Untuk informasi selengkapnya, lihat ALTER DATA.

   ```
   ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
   ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
   ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
   ```

   Anda juga dapat menambahkan tampilan ke datashare.

   ```
   CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
    public.tickit_sales_redshift;
   ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;
   ```

   Gunakan ALTER DATASHARE untuk berbagi skema, dan tabel, tampilan, dan fungsi dalam skema yang diberikan. Superuser, pemilik datashare, atau pengguna yang memiliki izin ALTER atau ALL pada datashare dapat mengubah datashare untuk menambahkan objek ke atau menghapus objek darinya. Pengguna harus memiliki izin untuk menambah atau menghapus objek dari datashare. Pengguna juga harus menjadi pemilik objek atau memiliki izin SELECT, USAGE, atau ALL pada objek.

   Gunakan klausa INCLUDENEW untuk menambahkan tabel baru, tampilan, atau fungsi yang ditentukan pengguna SQL (UDF) yang dibuat dalam skema tertentu ke datashare. Hanya pengguna super yang dapat mengubah properti ini untuk setiap pasangan datashare-schema.

   ```
   ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
   ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
   ```

Anda juga dapat menggunakan konsol Amazon Redshift untuk menambah atau menghapus objek dari berbagi data. Lihat informasi selengkapnya di Tambahkan objek datashare ke datashares, Menghapus objek datashare dari datashares, dan Mengedit AWS Data Exchange berbagi data.

3. Untuk mengotorisasi akses ke data share untuk AWS Data Exchange, lakukan salah satu hal berikut:

   - Secara eksplisit mengotorisasi akses ke datashare AWS Data Exchange dengan menggunakan ADX kata kunci di API. `aws redshift authorize-data-share` Hal ini memungkinkan AWS Data Exchange untuk mengenali datashare di akun layanan dan mengelola menghubungkan konsumen ke datashare.

     ```
     aws redshift authorize-data-share
     --data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
     {PRODUCER_CLUSTER_NAMESPACE}/salesshare
     --consumer-identifier ADX
     ```

     Anda dapat menggunakan kunci kondisi `ConsumerIdentifier` onal untuk `DeauthorizeDataShare` API `AuthorizeDataShare` dan untuk secara eksplisit mengiz AWS Data Exchange inkan atau menolak melakukan panggilan ke dua API dalam kebijakan IAM.

     JSON

     ```
     {
         "Version":"2012-10-17",
         "Statement": [
             {
                 "Sid": "VisualEditor0",
                 "Effect": "Deny",
                 "Action": [
                     "redshift:AuthorizeDataShare",
                     "redshift:DeauthorizeDataShare"
                 ],
                 "Resource": "*",
                 "Condition": {
                     "StringEqualsIgnoreCase": {
                         "redshift:ConsumerIdentifier": "ADX"
                     }
                 }
     ```

```
        }
    ]
}
```

- Gunakan konsol Amazon Redshift untuk mengotorisasi atau menghapus otorisasi AWS Data Exchange berbagi data. Untuk informasi selengkapnya, lihat Mengotorisasi atau menghapus otorisasi dari datashare.
- Secara opsional, Anda dapat secara implisit mengotorisasi akses ke AWS Data Exchange datashare saat mengimpor datashare ke dataset. AWS Data Exchange

Untuk menghapus otorisasi untuk akses ke AWS Data Exchange data share, gunakan `ADX` kata kunci dalam operasi `aws redshift deauthorize-data-share` API. Dengan melakukan ini, Anda memungkinkan AWS Data Exchange untuk mengenali datashare di akun layanan dan mengelola penghapusan asosiasi dari datashare.

```
aws redshift deauthorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier ADX
```

4. Buat daftar data share yang dibuat di cluster dan lihat isi datashare.

   Contoh berikut menampilkan informasi dari sebuah datashare bernama salesshare. Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

```
DESC DATASHARE salesshare;

 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name          |   include_new
-------------------+--------------------------------------+------------
+------------+-------------+---------------------------------+-------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_users_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_venue_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_category_redshift|
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_date_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_event_redshift   |
```

```
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_listing_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_sales_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | schema      | public                         |  t
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | view        | public.sales_data_summary_view |
```

Contoh berikut menampilkan data share keluar dalam cluster produsen.

```
SHOW DATASHARES LIKE 'sales%';
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
share_name | share_owner  | source_database | consumer_database | share_type |
 createdate      | is_publicaccessible  | share_acl | producer_account |
 producer_namespace
-----------+--------------+-----------------+-------------------+------------
+---------------------+----------------------+-----------+-----------------
+--------------------------------------
salesshare |    100       | dev             |                   |  OUTBOUND
 | 2020-12-09 02:27:08 |          True        |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

Anda juga dapat menggunakanSVV_DATASHARES,SVV_DATASHARE_KONSUMEN, dan SVV_DATASHARE_OBJECT untuk melihat data share, objek di dalam datashare, dan konsumen datashare.

5. Jatuhkan data share. Kami menyarankan agar Anda tidak menghapus data yang AWS Data Exchange dibagikan ke orang lain Akun AWS menggunakan pernyataan DROP DATASHARE. Akun tersebut akan kehilangan akses ke datashare. Tindakan ini tidak dapat diubah. Ini mungkin melanggar ketentuan penawaran produk data AWS Data Exchange. Jika Anda ingin menghapus AWS Data Exchange datashare, lihat. Catatan penggunaan DROP DATASHARE

   Contoh berikut menjatuhkan datashare bernama salesshare.

   ```
   DROP DATASHARE salesshare;
   ```

```
ERROR:  Drop of ADX-managed datashare salesshare requires session variable
 datashare_break_glass_session_var to be set to value '620c871f890c49'
```

Untuk mengizinkan penghapusan AWS Data Exchange datashare, atur variabel datashare_break_glass_session_var dan jalankan pernyataan DROP DATASHARE lagi. Jika Anda ingin menghapus AWS Data Exchange datashare, lihat. Catatan penggunaan DROP DATASHARE

Anda juga dapat menggunakan konsol Amazon Redshift untuk menghapus data share. Untuk informasi selengkapnya, lihat Menghapus AWS Data Exchange data share yang dibuat di akun Anda.

6. Gunakan ALTER DATASHARE untuk menghapus objek dari datashares kapan saja dari datashare. Gunakan REVOKE USAGE ON untuk mencabut izin pada datashare ke konsumen tertentu. Ini mencabut izin USAGE pada objek dalam datashare dan langsung menghentikan akses ke semua cluster konsumen. Mendaftarkan berbagi data dan kueri metadata, seperti daftar database dan tabel, tidak mengembalikan objek bersama setelah akses dicabut.

   ```
   ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
   ```

   Anda juga dapat menggunakan konsol Amazon Redshift untuk mengedit data share. Untuk informasi selengkapnya, lihat Mengedit AWS Data Exchange berbagi data.

7. Berikan atau cabut PENGGUNAAN GRANT dari AWS Data Exchange data share. Anda tidak dapat memberikan atau mencabut PENGGUNAAN GRANT untuk AWS Data Exchange datashare. Contoh berikut menunjukkan kesalahan saat izin GRANT USAGE diberikan kepada Akun AWS data yang AWS Data Exchange mengelola.

   ```
   CREATE DATASHARE salesshare MANAGEDBY ADX;
   ```

   ```
   GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '012345678910';
   ERROR:  Permission denied to add/remove consumer to/from datashare salesshare.
    Datashare consumers are managed by ADX.
   ```

   Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

Jika Anda adalah administrator produsen, ikuti langkah-langkah ini untuk membuat dan menerbitkan produk datashare di konsol: AWS Data Exchange

- Ketika AWS Data Exchange datashare telah dibuat, produsen membuat dataset baru, mengimpor aset, membuat revisi, dan membuat dan menerbitkan produk baru.

  Gunakan konsol Amazon Redshift untuk membuat kumpulan data. Untuk informasi selengkapnya, lihat Membuat kumpulan data pada AWS Data Exchange.

  Untuk informasi selengkapnya, lihat Men yediakan produk data pada AWS Data Exchange.

Bekerja dengan AWS Data Exchange berbagi data sebagai konsumen

Dengan Amazon Redshift, Anda dapat mengakses dan menganalisis kumpulan data AWS Data Exchange tanpa harus menyimpan atau mengelola salinan data.

Jika Anda seorang konsumen, ikuti langkah-langkah berikut untuk menemukan produk data yang berisi AWS Data Exchange datashares dan menanyakan data Amazon Redshift:

1. Di AWS Data Exchange konsol, temukan dan berlangganan produk data yang berisi data AWS Data Exchange share.

   Setelah langganan dimulai, Anda dapat mengakses data Amazon Redshift berlisensi yang diimpor sebagai aset ke kumpulan data yang berisi AWS Data Exchange data share.

   Untuk informasi selengkapnya tentang cara memulai menggunakan produk data yang berisi data AWS Data Exchange share, lihat Ber langganan produk data pada. AWS Data Exchange
2. Di konsol Amazon Redshift, buat cluster Amazon Redshift, jika diperlukan.

   Untuk informasi tentang cara membuat cluster, lihat Membuat cluster.
3. Buat daftar data share yang tersedia untuk Anda dan lihat konten dari datashares. Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

   Contoh berikut menampilkan informasi dari inbound datashares dari namespace produsen tertentu. Saat Anda menjalankan DESC DATASHARE sebagai administrator konsumen, Anda harus menentukan opsi AKUN dan NAMESPACE untuk melihat data share masuk.

```
DESC DATASHARE salesshare of ACCOUNT '123456789012' NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';

 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name           |   include_new
```

```
-------------------+--------------------------------------+------------
+------------+-------------+----------------------------------+------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_users_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_venue_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_category_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_date_redshift     |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_event_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_listing_redshift  |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_sales_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | schema      | public                          |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | view        | public.sales_data_summary_view  |
```

Hanya pengguna super cluster yang dapat melakukan ini. Anda juga dapat menggunakan SVV_DATASHARES untuk melihat datashares dan SVV_DATASHARE_OBJECTS untuk melihat objek di dalam datashare.

Contoh berikut menampilkan inbound datashares dalam cluster konsumen.

```
SHOW DATASHARES LIKE 'sales%';


 share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
------------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+------------------
+--------------------------------------
 salesshare |             |                 |                   | INBOUND
    |            |         t           |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

4. Buat database lokal yang merujuk ke datashares. Anda harus menentukan opsi AKUN dan NAMESPACE untuk membuat database lokal untuk berbagi AWS Data Exchange data. Untuk informasi selengkapnya, lihat BUAT BASIS DATA.

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF ACCOUNT '123456789012'
 NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Jika Anda ingin kontrol yang lebih terperinci atas akses ke objek dalam database lokal, gunakan klausa WITH PERMISSIONS saat membuat database. Dengan klausa ini, Anda dapat memberikan izin tingkat objek untuk objek dalam database pada langkah 6.

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF ACCOUNT
 '123456789012' NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Anda dapat melihat database yang Anda buat dari datashare dengan menanyakan tampilan. SVV_REDSHIFT_DATABASE Anda dapat terhubung ke database ini secara langsung, atau Anda dapat terhubung ke database lokal di cluster konsumen Anda dan melakukan kueri lintas database untuk menanyakan data dari database datashare. Anda tidak dapat membuat datashare di atas objek database yang dibuat dari datashare yang ada. Namun, Anda dapat menyalin data ke tabel terpisah di cluster konsumen, melakukan pemrosesan apa pun yang diperlukan, dan kemudian berbagi objek baru yang dibuat.

Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat database dari berbagi data. Untuk informasi selengkapnya, lihat Membuat database dari datashares.

5. (Opsional) Buat skema eksternal untuk merujuk dan menetapkan izin granular ke skema tertentu dalam database konsumen yang diimpor pada cluster konsumen. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
 'public';
```

6. Berikan izin pada database dan referensi skema yang dibuat dari berbagi data kepada pengguna atau peran dalam cluster konsumen sesuai kebutuhan. Untuk informasi selengkapnya, lihat HIBAH atau MENCABUT.

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

Jika Anda membuat database tanpa DENGAN IZIN, Anda hanya dapat menetapkan izin pada seluruh database yang dibuat dari datashare kepada pengguna dan peran Anda. Dalam beberapa kasus, Anda memerlukan kontrol halus pada subset objek database yang dibuat dari datashare. Jika demikian, Anda dapat membuat referensi skema eksternal yang menunjuk ke skema tertentu di datashare (seperti yang dijelaskan pada langkah sebelumnya) dan memberikan izin granular di tingkat skema.

Anda juga dapat membuat tampilan yang mengikat terlambat di atas objek bersama dan menggunakannya untuk menetapkan izin terperinci. Anda juga dapat mempertimbangkan untuk meminta cluster produsen membuat berbagi data tambahan untuk Anda dengan perincian yang diperlukan. Anda dapat membuat referensi skema sebanyak mungkin ke database yang dibuat dari datashare yang Anda butuhkan.

Jika Anda membuat database dengan WITH PERMISSIONS pada langkah 4, Anda harus menetapkan izin tingkat objek untuk objek dalam database bersama. Pengguna dengan hanya izin USAGE tidak dapat mengakses objek apa pun dalam database yang dibuat dengan WITH PERMISSIONS sampai mereka diberikan izin tingkat objek tambahan..

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

7. Kueri data dalam objek bersama di datashares.

   Pengguna dan peran dengan izin pada database konsumen dan skema pada cluster konsumen dapat menjelajahi dan menavigasi metadata objek bersama apa pun. Mereka juga dapat menjelajahi dan menavigasi objek lokal dalam cluster konsumen. Untuk melakukan ini, mereka menggunakan driver JDBC atau ODBC, perintah SHOW, atau tampilan SVV_ALL dan SVV_REDSHIFT.

   Cluster produsen mungkin memiliki banyak skema dalam database, tabel, dan tampilan dalam setiap skema. Pengguna di sisi konsumen hanya dapat melihat subset objek yang tersedia melalui datashare. Pengguna ini tidak dapat melihat seluruh metadata dari cluster produsen. Pendekatan ini membantu menyediakan kontrol keamanan metadata granular dengan berbagi data.

   Anda terus terhubung ke database cluster lokal. Tapi sekarang, Anda juga dapat membaca dari database dan skema yang dibuat dari datashare menggunakan notasi database.schema.table

tiga bagian. Anda dapat melakukan kueri yang menjangkau seluruh dan semua database yang terlihat oleh Anda. Ini bisa berupa database lokal di cluster atau database yang dibuat dari datashares. Atau, Anda dapat langsung terhubung ke database konsumen ini dan menjalankan kueri terhadap objek bersama dengan notasi parSIAL.

Anda dapat mengakses data menggunakan kualifikasi penuh. Untuk informasi selengkapnya, lihat Cross-database contoh kueri.

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;

 salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime
---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------
       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |    728.00 |
   109.20 | 2008-02-18 02:36:48
       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |     76.00 |
    11.40 | 2008-06-06 05:00:16
       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |    350.00 |
    52.50 | 2008-06-06 08:26:17
       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |    175.00 |
    26.25 | 2008-06-09 08:38:52
       5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |    154.00 |
    23.10 | 2008-08-31 09:17:02
```

Anda hanya dapat menggunakan pernyataan SELECT pada objek bersama. Namun, Anda dapat membuat tabel di cluster konsumen dengan menanyakan data dari objek bersama di database lokal yang berbeda.

Selain kueri, konsumen dapat membuat tampilan pada objek bersama. Hanya tampilan yang mengikat terlambat atau tampilan terwujud yang didukung. Amazon Redshift tidak mendukung tampilan reguler pada data bersama. Tampilan yang dibuat konsumen dapat menjangkau beberapa database lokal atau database yang dibuat dari berbagi data. Untuk informasi selengkapnya, lihat CREATE VIEW.

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
```

```
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

Memulai dengan AWS Lake Formation-berbagi data yang dikelola

Dengan Amazon Redshift, Anda dapat mengakses dan berbagi data langsung di seluruh AWS akun dan cluster Amazon Redshift melalui berbagi data yang dikelola AWS Lake Formation. AWS Lake Formation datashares memungkinkan penyedia data untuk secara aman berbagi data langsung dari Amazon S3 data lake mereka dengan konsumen mana pun, termasuk AWS akun lain dan cluster Amazon Redshift.

Bekerja dengan Lake Formation-managed datashares sebagai produser

Dengan Amazon Redshift, Anda dapat mengakses dan menganalisis data yang dibagikan melalui berbagi data AWS Lake Formation . AWS Lake Formation datashares memungkinkan berbagi data yang aman di seluruh AWS akun dan cluster Amazon Redshift tanpa harus menyalin atau memindahkan data yang mendasarinya.

Dengan berbagi data ke AWS Lake Formation, Anda dapat secara terpusat menentukan AWS Lake Formation izin untuk berbagi data Amazon Redshift dan membatasi akses pengguna ke objek dalam datashare.

Dengan Amazon Redshift, Anda dapat berbagi data langsung dengan aman di seluruh AWS akun dan cluster Amazon Redshift menggunakan pembagian data yang AWS Lake Formation dikelola sebagai produsen. Lake Formation-managed datashare adalah objek yang Anda gunakan untuk berbagi data langsung dari cluster Amazon Redshift Anda dengan AWS akun dan layanan lain.

Sebagai administrator cluster produsen atau workgroup, ikuti langkah-langkah ini untuk berbagi data ke Lake Formation:

1. Buat berbagi data di cluster Anda dan otorisasi AWS Lake Formation untuk mengakses data share.

   Hanya superuser cluster dan pemilik database yang dapat membuat berbagi data. Setiap datashare dikaitkan dengan database selama pembuatan. Hanya objek dari database itu yang dapat dibagikan di datashare itu. Beberapa data share dapat dibuat pada database yang sama dengan perincian objek yang sama atau berbeda. Tidak ada batasan jumlah data share yang dapat Anda buat di cluster.

```
CREATE DATASHARE salesshare;
```

2. Tambahkan objek ke datashare. Administrator cluster produsen atau workgroup terus mengelola objek datashare yang tersedia. Untuk menambahkan objek ke datashare, tambahkan skema sebelum menambahkan objek. Saat Anda menambahkan skema, Amazon Redshift tidak menambahkan semua objek di bawahnya. Anda harus menambahkannya secara eksplisit. Untuk informasi lebih lanjut, lihat ALTER DATASHARE.

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

Anda juga dapat menambahkan tampilan ke datashare. Tampilan yang didukung adalah tampilan standar, tampilan pengikatan terlambat, dan tampilan terwujud.

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
 public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
```

Gunakan ALTER DATASHARE untuk berbagi skema, tabel, dan tampilan, dalam skema yang diberikan. Superuser, pemilik datashare, atau pengguna yang memiliki izin ALTER atau ALL pada datashare dapat mengubah datashare untuk menambahkan objek ke atau menghapus objek darinya. Pengguna database harus menjadi pemilik objek atau memiliki izin SELECT, USAGE, atau ALL pada objek.

Gunakan klausa INCLUDENEW untuk menambahkan tabel dan tampilan baru yang dibuat dalam skema tertentu ke datashare. Hanya pengguna super yang dapat mengubah properti ini untuk setiap pasangan datashare-schema.

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

3. Berikan akses datashare ke akun administrator Danau Formation.

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '012345678910' VIA DATA CATALOG;
```

Untuk mencabut penggunaan, gunakan perintah berikut.

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '012345678910' VIA DATA CATALOG;
```

4. Otorisasi akses ke datashare untuk Formasi Danau dengan menggunakan operasi API. `aws redshift authorize-data-share` Melakukannya memungkinkan Lake Formation mengenali datashare di akun layanan dan mengelola menghubungkan konsumen ke datashare.

```
aws redshift authorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier {"DataCatalog/<consumer-account-id>"}
```

Untuk menghapus otorisasi dari Lake Formation-managed datashares, gunakan operasi `aws redshift deauthorize-data-share` API. Dengan demikian, Anda mengizinkan AWS Lake Formation untuk mengenali datashare di akun layanan dan menghapus otorisasi.

```
aws redshift deauthorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier {"DataCatalog/<consumer-account-id>"}
```

Kapan saja, jika cluster produsen atau administrator grup kerja memutuskan bahwa tidak perlu lagi berbagi data dengan cluster konsumen atau kelompok kerja, mereka dapat menggunakan DROP DATASHARE untuk menghapus datashare, mencabut otorisasi datashare, atau mencabut izin datashare. Izin dan objek terkait dalam Formasi Danau tidak dihapus secara otomatis.

```
DROP DATASHARE salesshare;
```

Setelah mengotorisasi akun Lake Formation untuk mengelola datashare, administrator Lake Formation dapat menemukan datashare bersama, mengaitkan dateshare dengan Katalog Data ARN, dan membuat database di tautan ke datashare. AWS Glue Data Catalog Untuk mengaitkan berbagi data menggunakan AWS CLI, gunakan perintah associate-data-share-consumer. Untuk berbagi data Wilayah AWS, tentukan `--region` parameter dalam `associate-data-share-consumer` perintah atau gunakan AWS konsol untuk memilih konsumen data Anda. Contoh berikut menunjukkan cara berbagi Formation-managed datashare Danau di seluruh Wilayah.

```
aws redshift associate-data-share-consumer --region <region-1>
```

```
--data-share-arn 'arn:aws:redshift:us-
east-1:12345678912:datashare:035c45ea-61ce-86f0-8b75-19ac6102c3b7/sample_share'
--consumer-arn 'arn:aws:glue:<region-1>:111912345678:catalog'
```

Administrator Formasi Danau juga harus membuat sumber daya lokal yang menentukan bagaimana objek dalam datashare harus dipetakan ke objek dalam Formasi Danau. Untuk informasi selengkapnya tentang menemukan berbagi data dan membuat sumber daya lokal, lihat M engelola izin untuk data di Amazon Redshift datashare.

Bekerja dengan Lake Formation-managed datashares sebagai konsumen

Dengan Amazon Redshift, Anda dapat mengakses dan menganalisis data yang dibagikan dengan Anda melalui berbagi data AWS Lake Formation . Datashare adalah produk data yang berisi kumpulan objek data, seperti tabel atau database, dari sumber data yang berbeda.

Setelah AWS Lake Formation administrator menemukan undangan datashare dan membuat database di dalam AWS Glue Data Catalog yang terhubung ke datashare, administrator cluster konsumen atau kelompok kerja dapat mengaitkan cluster dengan datashare dan database di dalamnya AWS Glue Data Catalog, membuat database lokal ke cluster konsumen atau kelompok kerja, dan memberikan akses ke pengguna dan peran di cluster konsumen atau kelompok kerja Amazon Redshift untuk memulai kueri. Ikuti langkah-langkah ini untuk mengatur izin kueri.

1. Pada konsol Amazon Redshift, buat cluster Redshift untuk berfungsi sebagai cluster konsumen atau kelompok kerja, jika diperlukan. Untuk informasi tentang cara membuat cluster, lihat Membuat cluster.
2. Untuk membuat daftar database mana di cluster AWS Glue Data Catalog konsumen atau kelompok kerja yang dapat diakses pengguna, jalankan perintah SHOW DATABASE.

   ```
   SHOW DATABASES FROM DATA CATALOG [ACCOUNT <account-id>,<account-id2>] [LIKE
    <expression>]
   ```

   Melakukannya mencantumkan sumber daya yang tersedia dari Katalog Data, seperti ARN AWS Glue database, nama database, dan informasi tentang datashare.
3. Menggunakan AWS Glue database ARN dari SHOW DATABASES, buat database lokal di cluster konsumen atau workgroup. Untuk informasi selengkapnya, lihat MEMBUAT DATABASE.

```
CREATE DATABASE lf_db FROM ARN <lake-formation-database-ARN> WITH [NO] DATA CATALOG
 SCHEMA [<schema>];
```

4. Berikan akses pada database dan referensi skema yang dibuat dari berbagi data kepada pengguna dan peran dalam cluster konsumen atau kelompok kerja sesuai kebutuhan. Untuk informasi lebih lanjut, lihat MEMBERIKAN https://docs.aws.amazon.com/redshift/latest/dg/r_GRANT.html atau MENC ABUT. Perhatikan bahwa pengguna yang dibuat dari perintah CREATE USER tidak dapat mengakses objek di datashare yang telah dibagikan ke Lake Formation. Hanya pengguna dengan akses ke Redshift dan Lake Formation yang dapat mengakses berbagi data yang telah dibagikan dengan Lake Formation.

   ```
   GRANT USAGE ON DATABASE sales_db TO IAM:Bob;
   ```

   Sebagai administrator cluster konsumen atau workgroup, Anda hanya dapat menetapkan izin pada seluruh database yang dibuat dari datashare kepada pengguna dan peran Anda. Dalam beberapa kasus, Anda memerlukan kontrol halus pada subset objek database yang dibuat dari datashare.

   Anda juga dapat membuat tampilan yang mengikat terlambat di atas objek bersama dan menggunakannya untuk menetapkan izin terperinci. Anda juga dapat mempertimbangkan untuk meminta cluster produsen atau kelompok kerja membuat berbagi data tambahan untuk Anda dengan perincian yang diperlukan. Anda dapat membuat sebanyak mungkin referensi skema ke database yang dibuat dari datashare.

5. Pengguna database dapat menggunakan tampilan SVV_EXTERNAL_TABLES dan SVV_EXTERNAL_COLUMNS untuk menemukan semua tabel atau kolom bersama dalam database AWS Glue

   ```
   SELECT * from svv_external_tables WHERE redshift_database_name = 'lf_db';

   SELECT * from svv_external_columns WHERE redshift_database_name = 'lf_db';
   ```

6. Kueri data dalam objek bersama di datashares.

   Pengguna dan peran dengan izin pada database konsumen dan skema pada cluster konsumen atau kelompok kerja dapat menjelajahi dan menavigasi metadata objek bersama apa pun. Mereka juga dapat menjelajahi dan menavigasi objek lokal dalam cluster konsumen atau

kelompok kerja. Untuk melakukannya, mereka dapat menggunakan driver JDBC atau ODBC atau tampilan SVV_ALL dan SVV_EXTERNAL.

```
SELECT * FROM lf_db.schema.table;
```

Anda hanya dapat menggunakan pernyataan SELECT pada objek bersama. Namun, Anda dapat membuat tabel di cluster konsumen dengan menanyakan data dari objek bersama di database lokal yang berbeda.

```
// Connect to a local cluster database

// Create a view on shared objects and access it.

CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

Memulai penulisan multi-gudang menggunakan berbagi data di Amazon Redshift

Anda dapat berbagi objek database untuk membaca dan menulis di berbagai cluster Amazon Redshift atau grup kerja Amazon Redshift Serverless dalam kelompok kerja yang sama Akun AWS, di seluruh akun, dan di seluruh Wilayah. Prosedur dalam topik ini menunjukkan cara mengatur berbagi data yang mencakup izin menulis. Anda dapat memberikan izin seperti SELECT, INSERT, dan UPDATE untuk tabel yang berbeda dan USAGE dan CREATE untuk skema.

Data langsung dan tersedia untuk semua gudang segera setelah Anda melakukan transaksi penulisan Administrator akun produsen dapat menentukan apakah ruang nama atau Wilayah tertentu mendapatkan baca-saja, baca-dan-tulis, atau akses apa pun ke data. Prosedur mengasumsikan Anda bekerja di database di cluster yang disediakan atau grup kerja Amazon Redshift Serverless.

Dengan Amazon Redshift, Anda dapat mengelola berbagi data dengan menulis menggunakan konsol atau antarmuka SQL untuk mengontrol akses dan mengatur data di seluruh cluster dan

AWS akun Amazon Redshift. Bagian berikut memberikan petunjuk langkah demi langkah tentang mengonfigurasi dan mengelola berbagi data dengan penulisan menggunakan Amazon Redshift.

Untuk daftar Wilayah tempat berbagi data tersedia, lihatWilayah AWS Di mana berbagi data tersedia. Untuk pertimbangan dan batasan penulisan, lihatPertimbangan untuk berbagi data di Amazon Redshift.

> **Note**
>
> Penulisan multi-gudang Amazon Redshift menggunakan berbagi data hanya didukung di Amazon Redshift patch 186 untuk cluster yang disediakan pada trek versi saat ini 1.0.78881 atau lebih tinggi, dan untuk grup kerja Amazon Redshift Serverless pada versi 1.0.78890 atau lebih tinggi.

Topik

- Menghubungkan ke database di Amazon Redshift
- Tindakan produser untuk berbagi data baru di Amazon Redshift
- Tindakan konsumen untuk berbagi data baru di Amazon Redshift
- Tindakan produsen untuk berbagi data yang ada di Amazon Redshift
- Tindakan konsumen untuk berbagi data yang ada di Amazon Redshift

Menghubungkan ke database di Amazon Redshift

Dengan Amazon Redshift, Anda dapat membuat koneksi ke cluster gudang data Anda dan menjalankan kueri SQL, memuat data, atau melakukan tugas administratif. Menghubungkan ke database mengacu pada proses pembuatan saluran aman antara aplikasi atau alat klien dan cluster Amazon Redshift. Bagian berikut memberikan petunjuk langkah demi langkah tentang cara menyambung ke database Amazon Redshift.

Console

Hubungkan ke database untuk melihat database dan objek dalam database atau untuk melihat berbagi data di gudang data Amazon Redshift. KredenSIAL pengguna yang digunakan untuk terhubung ke database tertentu harus memiliki izin yang diperlukan untuk melihat semua data share.

Jika tidak ada koneksi lokal, lakukan salah satu hal berikut:

- Jika Anda adalah administrator produsen, buka tab Cluster untuk cluster yang disediakan, atau tab konfigurasi ruang nama untuk titik akhir tanpa server. Pilih cluster atau namespace masing-masing dari daftar.
- Di halaman detail cluster atau namespace, dari tab Datashares, pilih Hubungkan ke database dan lakukan salah satu hal berikut:
  - Di bagian Datashares dari ruang nama dan Akun AWS bagian lain, lihat berbagi data dari cluster lain, ruang nama, atau akun.
  - Di Datashares yang dibuat di bagian cluster saya, lihat berbagi data di cluster Anda.
- Pada jendela Sambungkan ke database, lakukan salah satu hal berikut:
  - Jika Anda memilih Buat koneksi baru, pilih Manajer AWS Rahasia untuk menggunakan rahasia yang disimpan untuk mengotentikasi akses koneksi.

    Atau, pilih Kredensial sementara untuk menggunakan kredensial database untuk mengotentikasi akses untuk koneksi. Tentukan nilai untuk nama Database dan pengguna Database.

    Pilih Hubungkan.
  - Pilih Gunakan koneksi terbaru untuk menyambung ke database lain yang memiliki izin yang diperlukan.

    Amazon Redshift secara otomatis membuat koneksi.

Setelah koneksi database dibuat, Anda dapat mulai membuat data share, menanyakan data share, atau membuat database dari datashares.

Tindakan produser untuk berbagi data baru di Amazon Redshift

Dengan Amazon Redshift, Anda dapat berbagi data langsung di seluruh cluster atau AWS akun Amazon Redshift menggunakan berbagi data. Datashare adalah objek produser-konsumen yang Anda gunakan untuk berbagi data langsung dari cluster Amazon Redshift Anda dengan cluster atau akun lain. AWS Membuat berbagi data memungkinkan berbagi data yang aman sambil mempertahankan kontrol atas akses dan memastikan data tetap mutakhir. Bagian berikut memberikan detail tentang membuat berbagi data dan menambahkan objek database seperti skema, tabel, dan tampilan untuk berbagi data langsung dengan aman.

Topik

- Membuat datashare di Amazon Redshift
- Menambahkan objek ke datashare di Amazon Redshift
- Menambahkan konsumen data ke datashare di Amazon Redshift
- Mengotorisasi datashare di Amazon Redshift

Membuat datashare di Amazon Redshift

Datashare adalah wadah logis objek database, izin, dan konsumen. Konsumen adalah cluster yang disediakan Amazon Redshift atau ruang nama Amazon Redshift Serverless di akun Anda dan lainnya. Akun AWS Setiap datashare dikaitkan dengan database yang dibuat dan hanya objek dari database itu yang dapat ditambahkan. Sebagai administrator produsen, Anda dapat membuat data share di konsol dan dengan SQL dengan mengikuti salah satu prosedur di bawah ini.

Console

Di konsol, Anda dapat membuat data share dari tab Datashares di halaman detail cluster atau namespace. Setelah datashare dibuat, Anda dapat membuat database dari datashare pada konsumen sebagai administrator konsumen.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Di halaman detail cluster atau namespace, dari tab Datashares, di bagian Datashares, sambungkan ke database jika Anda tidak memiliki koneksi database. Di bagian Datashares yang dibuat di akun saya, pilih Buat datashare. Halaman Buat datashare muncul.
4. Pilih Buat datashare. Anda hanya dapat membuat datashare dari database lokal. Jika Anda belum terhubung ke database, halaman Connect to database akan muncul. Ikuti langkah-langkah Menghubungkan ke basis data untuk terhubung ke database. Jika ada koneksi baru-baru ini, halaman Buat datashare muncul.
5. Di bagian informasi Datashare, pilih salah satu dari yang berikut ini:
   - Pilih Datashare untuk membuat berbagi data untuk berbagi data untuk tujuan baca atau tulis di berbagai gudang data Amazon Redshift (cluster yang disediakan atau titik akhir Tanpa Server) atau di gudang data yang sama atau berbeda. Akun AWS Akun AWS
   - Pilih AWS Data Exchange datashare untuk membuat data share untuk melisensikan data Anda. AWS Data Exchange

6. Tentukan nilai untuk nama Datashare, Nama Database, dan Akses publik. Saat Anda mengubah nama database, buat koneksi database baru.
7. Tambahkan objek ke datashare Anda menggunakan bagian Izin yang dic akup atau Izin Lang sung. Untuk menambahkan objek ke datashare, lihat. Membuat datashare di Amazon Redshift
8. Di bagian Konsumen data, Anda dapat memilih untuk mempublikasikan ke Amazon Redshift, atau mempublikasikan ke AWS Glue Data Catalog, yang memulai proses berbagi data dengan Lake Formation. Menerbitkan datashare Anda ke Amazon Redshift berarti berbagi data Anda dengan namespace lain atau akun Amazon Redshift yang bertindak sebagai konsumen.

   > **Note**
   >
   > Setelah datashare dibuat, Anda tidak dapat mengedit konfigurasi untuk dipublikasikan ke opsi lain.

9. Pilih Buat datashare.

SQL

Perintah berikut membuat datashare:

```
CREATE DATASHARE salesshare;
```

Pada saat pembuatan datashare, setiap datashare dikaitkan dengan database. Hanya objek dari database itu yang dapat dibagikan di datashare itu. Beberapa data share dapat dibuat pada database yang sama dengan perincian objek yang sama atau berbeda. Tidak ada batasan jumlah data share yang dapat dibuat oleh cluster. Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat berbagi data. Untuk informasi selengkapnya, lihat BUAT DATASHARE.

Anda juga dapat mengontrol pembatasan keamanan ke datashare selama pembuatan. Contoh berikut menunjukkan bahwa konsumen dengan akses IP publik diizinkan untuk membaca datashare.

```
CREATE DATASHARE my_datashare [PUBLICACCESSIBLE = TRUE];
```

Menyetel PUBLICACCESSIBLE = TRUE memungkinkan konsumen untuk menanyakan data Anda dari cluster yang dapat diakses publik dan kelompok kerja yang disediakan. Tinggalkan ini atau secara eksplisit atur ke false jika Anda tidak ingin mengizinkannya.

Anda dapat memodifikasi properti tentang jenis konsumen setelah Anda membuat datashare. Misalnya, Anda dapat menentukan bahwa cluster yang ingin menggunakan data dari datashare tertentu tidak dapat diakses publik. Kueri dari cluster konsumen yang tidak memenuhi batasan keamanan yang ditentukan dalam datashare ditolak saat runtime kueri. Untuk informasi selengkapnya, lihat ALTER DATA.

Menambahkan objek ke datashare di Amazon Redshift

Anda dapat menambahkan objek database dari berbagai jenis pada konsol dan dengan SQL dengan mengikuti salah satu prosedur di bawah ini.

Console

Anda dapat menambahkan objek ke datashare Anda menggunakan izin terc akup atau bagian izin langsung. Pilih Berikan izin terlingkup atau Ber ikan izin langsung untuk menambahkan objek. Pilih tombol Tambah untuk menambahkan objek. Dialog muncul. Lakukan langkah-langkah berikut ini:

1. Jika Anda memilih Ber ikan izin bercakupan, halaman Berikan izin cakupan akan muncul di mana Anda dapat memberikan izin cakupan baik di tingkat database atau skema. Berbagi data dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema. Untuk lebih jelasnya lihat,Izin terlingkup.

    1. Selanjutnya, pilih Iz in cakupan database untuk memberikan izin terlingkup di tingkat database. Saat Anda memberikan izin cakupan, izin tersebut berlaku untuk database saat ini saat membuat datashare. Izin ini tidak dapat diberikan ke objek individual dan berlaku untuk objek yang ada dan baru (skema, tabel, tampilan, UDF).
    2. Pilih izin cakupan untuk skema, tabel atau tampilan atau fungsi yang ditentukan pengguna. Ini berarti semua objek dalam database memiliki izin yang dipilih yang diberikan kepada konsumen. Pilih Grant untuk menyelesaikan pemberian izin bercakupan database.
    3. Kemudian, pilih izin cakupan skema untuk memberikan izin cakupan di tingkat skema. Saat Anda memberikan izin cakupan skema, semua objek yang ditambahkan ke skema memiliki izin datashare yang ditentukan.
    4. Pilih skema yang ingin Anda tambahkan ke datashare dari dropdown. Anda hanya dapat memilih satu skema pada satu waktu. Kemudian, pilih izin langsung yang ingin Anda berikan pada skema yang dipilih.

5. Pilih izin cakupan untuk objek skema Anda seperti tabel, tampilan, dan fungsi yang ditentukan pengguna. Izin diberikan pada semua objek yang cocok dalam skema. Ini bisa berupa objek yang ada atau yang ditambahkan di masa mendatang. Saat diterapkan, Anda tidak dapat menghapus izin dari objek tanpa mencabut izin cakupan.
6. Pilih Grant untuk menyelesaikan pemberian izin bercakupan skema.

2. Jika Anda memilih Berikan izin langsung, halaman Berikan izin langsung muncul di mana Anda dapat memberikan izin langsung pada setiap tingkat objek seperti skema, tabel, tampilan, atau fungsi yang ditentukan pengguna. Untuk memberikan izin langsung, Anda harus terlebih dahulu menambahkan skema yang relevan ke datashare.

   1. Selanjutnya, pilih Berikan izin langsung ke skema untuk menerapkan izin langsung pada skema tertentu. Kemudian, pilih izin skema untuk objek skema Anda seperti tabel, tampilan, dan fungsi yang ditentukan pengguna dan pilih skema yang ingin Anda tambahkan ke datashare. Pilih Grant untuk menyelesaikan penambahan skema ke datashare.
   2. Setelah Anda memiliki skema yang ditambahkan ke datashare Anda, Anda dapat melanjutkan dengan menambahkan izin langsung untuk objek skema Anda. Pilih Berikan izin langsung lagi. Halaman Beri izin langsung muncul. Kemudian, arahkan ke tab izin langsung untuk objek skema.
   3. Pilih Berikan izin langsung ke tabel dan tampilan untuk memberikan izin langsung tingkat objek pada objek ini. Pilih izin langsung yang diperlukan dan objek yang diperlukan dari daftar. Gunakan bidang pencarian untuk menemukan objek datashare. Pilih Grant untuk menyelesaikan penambahan tabel dan tampilan ke datashare.
   4. Pilih Berikan izin langsung ke fungsi yang ditentukan pengguna untuk memberikan izin langsung tingkat objek pada fungsi yang ditentukan pengguna. Pilih izin langsung yang diperlukan dan objek yang diperlukan dari daftar. Gunakan bidang pencarian untuk menemukan objek datashare. Pilih Grant untuk menyelesaikan penambahan fungsi yang ditentukan pengguna ke datashare.

3. Anda juga dapat memilih apakah Anda ingin menambahkan objek masa depan. Ketika Anda memilih untuk menyertakan objek datashare yang ditambahkan ke skema, itu berarti bahwa objek yang ditambahkan ke skema ditambahkan ke datashare secara otomatis.
4. Pilih Tambah untuk menyelesaikan bagian dan menambahkan objek. Mereka terdaftar di bawah objek Datashare.
5. Setelah menambahkan objek, Anda dapat memilih objek individual dan mengedit izin mereka. Jika Anda memilih skema, dialog muncul yang menanyakan apakah Anda ingin

menambahkan izin yang dicakup. Ini membuatnya sehingga setiap objek yang ada atau ditambahkan ke skema memiliki kumpulan izin yang telah dipilih sebelumnya, sesuai untuk jenis objek. Misalnya, administrator dapat mengatur bahwa semua tabel yang ditambahkan memiliki izin SELECT dan UPDATE, misalnya.

6. Semua objek datashare terdaftar di bawah bagian Izin terc akup atau Izin Lang sung.
7. Di bagian Konsumen data, Anda dapat menambahkan ruang nama atau menambahkan AWS akun sebagai konsumen datashare.
8. Pilih Buat datashare untuk menyimpan perubahan Anda.

Setelah Anda membuat datashare, itu muncul dalam daftar di bawah Datashares yang dibuat di namespace saya. Jika Anda memilih datashare dari daftar, Anda dapat melihat konsumen, objeknya, dan properti lainnya.

SQL

Dengan SQL, pemilik datashare harus memberikan USAGE pada skema yang ingin mereka tambahkan ke datashare. GRANT digunakan untuk memberikan berbagai tindakan pada skema, termasuk CREATE dan USAGE. Skema menyimpan objek bersama:

```
CREATE SCHEMA myshared_schema1;
CREATE SCHEMA myshared_schema2;

GRANT USAGE ON SCHEMA myshared_schema1 TO DATASHARE my_datashare;
GRANT CREATE, USAGE ON SCHEMA myshared_schema2 TO DATASHARE my_datashare;
```

Atau, administrator dapat terus menjalankan perintah ALTER untuk menambahkan skema ke datashare. Hanya izin USAGE yang diberikan ketika skema ditambahkan dengan cara ini.

```
ALTER DATASHARE my_datashare ADD SCHEMA myshared_schema1;
```

Setelah administrator menambahkan skema, mereka dapat memberikan izin datashare pada objek dalam skema. Ini bisa berupa izin baca dan tulis. Sampel GRANT ALL menunjukkan cara memberikan semua izin.

```
GRANT SELECT, INSERT ON TABLE myshared_schema1.table1, myshared_schema1.table2,
 myshared_schema2.table1
TO DATASHARE my_datashare;
```

```
GRANT ALL ON TABLE myshared_schema1.table4 TO DATASHARE my_datashare;
```

Anda dapat terus menjalankan perintah seperti ALTER DATASHARE untuk menambahkan tabel. Ketika Anda melakukannya, hanya izin SELECT yang diberikan pada objek yang ditambahkan.

```
ALTER DATASHARE my_datashare ADD TABLE myshared_schema1.table1,
 myshared_schema1.table2, myshared_schema2.table1;
```

Anda dapat memberikan izin cakupan ke datashare pada semua objek jenis dalam database atau skema. Berbagi data dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema.

Anda dapat melihat cakupan izin cakupan tingkat database di. SVV_DATABASE_HAK ISTIMEWA Anda dapat melihat cakupan izin cakupan tingkat skema di. SVV_SCHEMA_HAK ISTIMEWA

Berikut ini adalah sintaks untuk memberikan izin cakupan untuk berbagi data. Untuk informasi selengkapnya tentang izin bercakupan, lihatIzin terlingkup.

```
GRANT { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }FOR SCHEMAS IN
DATABASE db_name
TO DATASHARE { datashare_name}

GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [PRIVILEGES] } }FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO DATASHARE { datashare_name}

GRANT { EXECUTE | ALL [ PRIVILEGES ] }FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO DATASHARE { datashare_name}
```

Menambahkan konsumen data ke datashare di Amazon Redshift

Anda dapat menambahkan satu atau lebih konsumen data ke berbagi data di konsol atau dengan SQL. Konsumen data dapat berupa ruang nama yang secara unik mengidentifikasi cluster Amazon Redshift atau. Akun AWS

Console

Anda harus secara eksplisit memilih untuk mematikan atau mengaktifkan berbagi data Anda ke cluster dengan akses publik.

- Pilih Tambahkan ruang nama ke datashare. Ruang nama adalah pengenal unik global (GUID) untuk cluster Amazon Redshift.
- Pilih Tambahkan Akun AWS ke datashare. Yang ditentukan Akun AWS harus memiliki izin akses ke datashare.

SQL

Dengan SQL, administrator memberikan penggunaan pada datashare ke namespace tertentu di akun. Anda dapat menemukan ID namespace sebagai bagian dari ARN di halaman detail cluster, di halaman detail namespace Amazon Redshift Serverless, atau dengan menjalankan perintah. `SELECT current_namespace;` Untuk informasi selengkapnya, lihat CURRENT_NAMESPACE.

```
GRANT USAGE ON DATASHARE my_datashare TO NAMESPACE '86b5169f-012a-234b-9fbb-
e2e24359e9a8';
```

Berikut ini adalah contoh bagaimana memberikan penggunaan datashare ke. Akun AWS

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012';
```

Berikut ini adalah contoh bagaimana memberikan penggunaan datashare ke akun Formasi Danau.

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012' VIA DATA CATALOG;
```

Mengotorisasi datashare di Amazon Redshift

Dengan Amazon Redshift, Anda dapat mengontrol akses ke berbagi data dengan mengotorisasi konsumen tertentu. Dengan berbagi data, Anda dapat berbagi data langsung di seluruh cluster Amazon Redshift di AWS akun yang sama atau berbeda, menyediakan cara yang mulus untuk mendistribusikan dan menggunakan data analitis. Bagian ini memberikan petunjuk langkah demi langkah untuk mengotorisasi dan mencabut akses konsumen ke data share Anda di Amazon Redshift.

> **Note**
>
> Jika Anda menambahkan namespace sebagai konsumen data, Anda tidak perlu melakukan otorisasi. Untuk mengotorisasi datashare, setidaknya harus ada satu konsumen data yang ditambahkan ke datashare.

Console

Sebagai administrator produsen di konsol, Anda dapat memilih konsumen data mana yang akan diotorisasi untuk mengakses data share atau menghapus otorisasi. Konsumen data resmi menerima pemberitahuan untuk mengambil tindakan pada berbagi data. Jika Anda menambahkan namespace sebagai konsumen data, Anda tidak perlu melakukan otorisasi.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Datashares. Dari sini Anda dapat melihat daftar yang disebut konsumen Datashares. Pilih satu atau lebih cluster konsumen yang ingin Anda otorisasi. Lalu, pilih Otorisasi.
3. Dialog Otorisasi akun muncul. Anda dapat memilih di antara beberapa jenis otorisasi.
   - Read-only pada [nama cluster atau nama kelompok kerja] - Ini berarti bahwa tidak ada izin menulis yang tersedia pada konsumen, bahkan jika pembuat datashare memberikan izin tulis.
   - Baca dan tulis di [nama cluster atau nama grup kerja] - Ini berarti bahwa semua izin yang diberikan oleh pembuat, termasuk izin menulis, tersedia pada konsumen.
4. Pilih Simpan.

Anda juga dapat mengotorisasi AWS Data Exchange sebagai konsumen.

1. Jika Anda memilih Publish to AWS Glue Data Catalog saat membuat datashare, Anda hanya dapat memberikan otorisasi datashare ke akun Lake Formation.

   Untuk AWS Data Exchange datashare, Anda hanya dapat mengotorisasi satu datashare pada satu waktu.

   Ketika Anda mengotorisasi AWS Data Exchange datashare, Anda berbagi datashare dengan AWS Data Exchange layanan dan memungkinkan AWS Data Exchange untuk

mengelola akses ke datashare atas nama Anda. AWS Data Exchange memungkinkan akses ke konsumen dengan menambahkan akun konsumen sebagai konsumen data ke AWS Data Exchange datashare ketika mereka berlangganan produk. AWS Data Exchange tidak memiliki akses baca ke datashare.

2. Pilih Simpan.

Setelah konsumen data diotorisasi, mereka dapat mengakses objek datashare dan membuat database konsumen untuk menanyakan data.

API

Administrator keamanan produsen menentukan yang berikut:

- Apakah akun lain dapat memiliki akses ke datashare atau tidak.
- Jika akun memiliki akses ke datashare, apakah akun tersebut memiliki izin menulis atau tidak.

Izin IAM berikut diperlukan untuk mengotorisasi datashare:

pergeseran merah: AuthorizeDataShare

Anda dapat mengotorisasi penggunaan dan penulisan menggunakan panggilan CLI atau dengan API:

```
authorize-data-share
--data-share-arn <value>
--consumer-identifier <value>
[--allow-writes | --no-allow-writes]
```

Untuk informasi selengkapnya tentang perintah, lihat otoriz e-data-share.

Pengidentifikasi konsumen dapat berupa:

- ID AWS akun dua belas digit.
- Pengidentifikasi namespace ARN.

> **Note**
>
> Izin menulis tidak diberikan pada langkah otorisasi. Mengotorisasi datashare untuk penulisan hanya memungkinkan akun memiliki izin menulis yang diberikan oleh

administrator datashare. Jika administrator tidak mengizinkan penulisan, satu-satunya izin yang tersedia untuk konsumen tertentu adalah SELECT, USAGE, dan EXECUTE.

Anda dapat mengubah otorisasi konsumen datashare dengan menelepon `authorize-data-share` lagi, tetapi dengan nilai yang berbeda. Otorisasi lama diganti oleh otorisasi baru. Jadi jika Anda awalnya mengotorisasi dan mengizinkan penulisan, tetapi mengotorisasi ulang dan menentukan `no-allow-writes` atau hanya tidak menentukan nilai, izin penulisan konsumen akan dicabut.

Tindakan konsumen untuk berbagi data baru di Amazon Redshift

Dengan Amazon Redshift, Anda dapat menggunakan berbagi data dari AWS akun lain, memungkinkan berbagi data lintas akun dan kolaborasi. Datashare adalah cara aman untuk berbagi data langsung di seluruh cluster Amazon Redshift, bahkan jika mereka berada di akun yang berbeda. AWS Bagian berikut memberikan langkah-langkah terperinci untuk mengonfigurasi akses, membuat database dari data share, memberikan izin tingkat objek, dan menanyakan data bersama.

Topik

- Mengaitkan datashare dari yang berbeda Akun AWS di Amazon Redshift
- Membuat database dari datashare di Amazon Redshift
- Memberikan izin tingkat objek kepada pengguna dan peran konsumen di Amazon Redshift
- Mengkueri data di datashare di Amazon Redshift

Mengaitkan datashare dari yang berbeda Akun AWS di Amazon Redshift

Dengan Amazon Redshift, Anda dapat mengaitkan berbagi data yang dibagikan oleh orang lain Akun AWS, memungkinkan berbagi data yang lancar dan aman melintasi batas-batas organisasi. Datashares adalah objek database yang dapat dibagikan yang merangkum data dari satu atau lebih database Amazon Redshift. Bagian berikut menunjukkan proses mengasosiasikan berbagi data.

Console

Sebagai administrator konsumen, Anda dapat mengaitkan satu atau beberapa data share yang dibagikan dari akun lain ke seluruh AWS akun atau ruang nama tertentu di akun Anda.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Datashares. Halaman daftar datashare muncul. Pilih Dari akun lain.
3. Di bagian Berbagi data dari akun lain, pilih datashare yang ingin Anda kaitkan dan pilih Associate. Saat halaman Associate datashare muncul, pilih salah satu jenis asosiasi berikut:
   - Pilih Seluruh AWS akun untuk mengaitkan semua ruang nama yang ada dan yang akan datang di berbagai Wil AWS ayah di AWS akun Anda dengan datashare.
   - Jika datashare dipublikasikan ke AWS Glue Data Catalog, Anda hanya dapat mengaitkan datashare dengan seluruh akun. AWS
4. Dari sini Anda dapat memilih Iz in yang diizinkan. Pilihannya adalah:
   - Read-only— Jika Anda memilih baca saja, izin tulis seperti UPDATE atau INSERT tidak tersedia pada konsumen, bahkan jika izin ini diberikan dan disahkan pada produsen.
   - Baca dan tulis - Pengguna datashare konsumen akan memiliki semua izin, baik baca maupun tulis, yang diberikan dan disahkan oleh produsen.
5. Atau, pilih Wil AWS ayah Tertentu dan ruang nama untuk mengaitkan satu atau lebih Wil AWS ayah dan ruang nama tertentu dengan datashare. Pilih Tam bah Wilayah untuk menambahkan Wil AWS ayah dan ruang nama tertentu ke datashare. Halaman Tambah AWS Wilayah muncul.
6. Pilih Wil AWS ayah.
7. Lakukan salah satu tindakan berikut:
   - Pilih Tambahkan semua ruang nama untuk menambahkan semua ruang nama yang ada dan yang akan datang di Wilayah ini ke datashare.
   - Pilih Tambahkan ruang nama tertentu untuk menambahkan satu atau lebih ruang nama spesifik di Wilayah ini ke datashare.
   - Pilih satu atau lebih ruang nama dan pilih Tambah Wilayah. AWS
8. Pilih Kaitkan.

Produsen dapat kembali dan mengubah pengaturan untuk otorisasi, yang dapat memengaruhi pengaturan asosiasi pada konsumen.

Jika Anda mengaitkan datashare dengan akun Lake Formation, buka konsol Lake Formation untuk membuat database, lalu tentukan izin atas database. Untuk informasi selengkapnya, lihat Menyiapkan izin untuk berbagi data Amazon Redshift di Panduan Peng AWS Lake Formation embang. Setelah Anda membuat AWS Glue database atau database federasi, Anda dapat menggunakan editor kueri v2 atau klien SQL pilihan dengan cluster konsumen Anda untuk menanyakan data.

Setelah datashare dikaitkan, berbagi data menjadi tersedia.

> **Note**
>
> Anda juga dapat mengubah asosiasi datashare kapan saja. Saat mengubah asosiasi dari AWS Wilayah dan ruang nama tertentu ke seluruh AWS akun, Amazon Redshift mengganti informasi Wilayah dan ruang nama tertentu dengan informasi akun. AWS Semua Wil AWS ayah dan ruang nama di saat itu memiliki Akun AWS akses ke datashare.

API

> **Note**
>
> Langkah-langkah di bagian ini dilakukan setelah administrator produsen memberikan tindakan spesifik pada objek database bersama dan, jika datashare sedang dibagikan dengan akun lain, administrator keamanan produsen mengotorisasi akses.

Administrator keamanan konsumen menentukan hal berikut:

- Apakah semua ruang nama dalam akun, ruang nama di Wilayah tertentu di akun, atau ruang nama tertentu memiliki akses ke datashare atau tidak.
- Jika ruang nama memiliki akses ke datashare, apakah namespace tersebut memiliki izin menulis atau tidak.

Administrator keamanan konsumen dapat mengaitkan datashare dengan perintah berikut:

```
associate-data-share-consumer
--data-share-arn <value>
```

```
--consumer-identifier <value>
[--allow-writes | --no-allow-writes]
```

Untuk informasi selengkapnya tentang perintah, lihat associate- data-share-consumer.

Administrator keamanan konsumen harus secara eksplisit menyet `allow-writes` el ke true saat mengaitkan datashare dengan namespace, untuk mengizinkan penggunaan perintah INSERT dan UPDATE. Jika tidak, pengguna hanya dapat melakukan operasi baca, seperti hak istimewa SELECT, USAGE, atau EXECUTE.

Anda dapat mengubah asosiasi namespace untuk datashare dengan memanggil `associate-data-share-consumer` lagi, dengan nilai yang berbeda. Asosiasi lama diganti oleh asosiasi baru, jadi jika Anda awalnya mengasosiasikan dan mengatur`allow-writes`, tetapi mengaitkan dan menentukan`no-allow-writes`, atau hanya tidak menentukan nilai, izin penulisan konsumen akan dicabut.

Membuat database dari datashare di Amazon Redshift

Dengan Amazon Redshift, Anda dapat menggunakan datashare untuk membuat database, lalu menanyakan data di seluruh data share dari cluster produsen untuk mengakses data langsung dengan aman tanpa menyalin atau mentransfernya. Langkah-langkah berikut mencakup detail pengaturan database di lingkungan Amazon Redshift Anda.

Console

Sebelum Anda dapat menanyakan data di datashare, Anda harus membuat database dari datashare. Anda hanya dapat membuat satu database dari datashare tertentu.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares. Daftar datashare muncul.
4. Di bagian Ber bagi data dari cluster lain, pilih Hubun gkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.
5. Pilih datashare tempat Anda ingin membuat database, lalu pilih Buat database dari datashare. Halaman Buat database dari datashare muncul.
6. Dalam nama Database, tentukan nama database. Nama database harus berupa 1—64 karakter alfanumerik (hanya huruf kecil) dan tidak boleh menjadi kata yang dicadangkan.

7. Pilih Buat.

Setelah database dibuat, Anda dapat menanyakan data dalam database atau melakukan operasi penulisan, jika telah diberikan, diotorisasi, dan dikaitkan oleh administrator konsumen.

API

Untuk berbagi data untuk tujuan membaca sebagai administrator konsumen, lakukan langkah-langkah berikut.

1. Buat daftar data share yang tersedia untuk Anda dan lihat konten dari datashares. Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

   Contoh berikut menampilkan informasi dari inbound datashares dari namespace produsen tertentu. Saat Anda menjalankan DESC DATASHARE sebagai administrator konsumen, Anda harus menentukan opsi NAMESPACE untuk melihat data share masuk.

```
DESC DATASHARE salesshare OF NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';


 producer_account  |          producer_namespace          | share_type |
 share_name | object_type |           object_name           |   include_new
-------------------+-------------------------------------+------------
+------------+-------------+---------------------------------+-----------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_users_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_venue_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_category_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_date_redshift     |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_event_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_listing_redshift  |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | table       | public.tickit_sales_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | schema      | public                          |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    |
 salesshare | view        | public.sales_data_summary_view  |
```

Hanya pengguna super cluster yang dapat melakukan ini. Anda juga dapat menggunakan SVV_DATASHARES untuk melihat datashares dan SVV_DATASHARE_OBJECTS untuk melihat objek di dalam datashare.

Contoh berikut menampilkan inbound datashares dalam cluster konsumen.

```
SHOW DATASHARES LIKE 'sales%';


 share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
------------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+-----------------
+--------------------------------------
 salesshare |             |                 |                   | INBOUND
    |            |         t           |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

2. Sebagai superuser database, Anda dapat membuat database lokal yang merujuk ke datashares. Untuk informasi selengkapnya, lihat BUAT BASIS DATA.

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Jika Anda ingin kontrol yang lebih terperinci atas akses ke objek dalam database lokal, gunakan klausa WITH PERMISSIONS saat membuat database. Dengan klausa ini, Anda dapat memberikan izin tingkat objek untuk objek dalam database pada langkah 4.

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Anda dapat melihat database yang Anda buat dari datashare dengan menanyakan tampilan. SVV_REDSHIFT_DATABASE Anda dapat terhubung ke database ini secara langsung, atau Anda dapat terhubung ke database lokal di cluster konsumen Anda dan melakukan kueri lintas database untuk menanyakan data dari database datashare.

> **Note**
>
> Anda tidak dapat membuat datashare di atas objek database yang dibuat dari datashare yang ada. Namun, Anda dapat menyalin data ke tabel terpisah di cluster konsumen, melakukan pemrosesan apa pun yang diperlukan, dan kemudian berbagi objek baru yang dibuat.

Anda juga dapat menggunakan konsol Amazon Redshift untuk membuat database dari berbagi data. Untuk informasi selengkapnya, lihat Membuat database dari datashares.

Memberikan izin tingkat objek kepada pengguna dan peran konsumen di Amazon Redshift

Sebagai administrator konsumen, Anda dapat memberikan izin kepada pengguna dan peran konsumen di tingkat objek dengan menyelesaikan langkah-langkah berikut.

SQL

Jika Anda membuat database tanpa DENGAN IZIN, Anda hanya dapat menetapkan izin pada seluruh database yang dibuat dari datashare kepada pengguna dan peran Anda.

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

Anda juga dapat membuat tampilan yang mengikat terlambat di atas objek bersama dan menggunakannya untuk menetapkan izin terperinci. Anda juga dapat mempertimbangkan untuk meminta cluster produsen membuat berbagi data tambahan untuk Anda dengan perincian yang diperlukan.

Jika Anda membuat database dengan WITH PERMISSIONS, Anda harus menetapkan izin tingkat objek untuk objek dalam database bersama. Pengguna dengan hanya izin USAGE tidak dapat mengakses objek apa pun dalam database yang dibuat dengan WITH PERMISSIONS sampai mereka diberikan izin tingkat objek tambahan..

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

Untuk informasi selengkapnya tentang pemberian izin dengan penulisan multi-gudang, lihatMengelola izin untuk berbagi data di Amazon Redshift.

Mengkueri data di datashare di Amazon Redshift

Dengan Amazon Redshift, Anda dapat menanyakan data di seluruh data share dari cluster produsen untuk mengakses data langsung dengan aman tanpa menyalin atau mentransfernya. Bagian berikut mencakup kueri berbagi data di lingkungan Amazon Redshift Anda.

Pengguna dan peran dengan izin pada database konsumen dan skema pada cluster konsumen dapat menjelajahi dan menavigasi metadata objek bersama apa pun. Mereka juga dapat menjelajahi dan menavigasi objek lokal dalam cluster konsumen. Untuk melakukan ini, mereka menggunakan driver JDBC atau ODBC atau tampilan SVV_ALL dan SVV_REDSHIFT.

Cluster produsen mungkin memiliki banyak skema dalam database, tabel, dan tampilan dalam setiap skema. Pengguna di sisi konsumen hanya dapat melihat subset objek yang tersedia melalui datashare. Pengguna ini tidak dapat melihat seluruh metadata dari cluster produsen. Pendekatan ini membantu menyediakan kontrol keamanan metadata granular dengan berbagi data.

Anda terus terhubung ke database cluster lokal. Tapi sekarang, Anda juga dapat membaca dari database dan skema yang dibuat dari datashare menggunakan notasi database.schema.table tiga bagian. Anda dapat melakukan kueri yang menjangkau seluruh dan semua database yang terlihat oleh Anda. Ini bisa berupa database lokal di cluster atau database yang dibuat dari datashares. Cluster konsumen tidak dapat terhubung ke database yang dibuat dari datashares.

Anda dapat mengakses data menggunakan kualifikasi penuh. Untuk informasi selengkapnya, lihat Cross-database contoh kueri.

SQL

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;

 salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime
---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------
       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |    728.00 |
  109.20 | 2008-02-18 02:36:48
       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |     76.00 |
   11.40 | 2008-06-06 05:00:16
       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |    350.00 |
   52.50 | 2008-06-06 08:26:17
```

```
     4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |    175.00 |
26.25 | 2008-06-09 08:38:52
     5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |    154.00 |
23.10 | 2008-08-31 09:17:02
```

Anda hanya dapat menggunakan pernyataan SELECT pada objek bersama. Namun, Anda dapat membuat tabel di cluster konsumen dengan menanyakan data dari objek bersama di database lokal yang berbeda.

Tindakan produsen untuk berbagi data yang ada di Amazon Redshift

Dengan Amazon Redshift, Anda dapat mengelola berbagi data yang ada untuk mengontrol akses ke data Anda di cluster Amazon Redshift. Bagian berikut memberikan panduan langkah demi langkah tentang memodifikasi objek datashare, mengelola izin datashare, dan memperbarui properti datashare untuk mengontrol dan mengaudit akses data secara efektif di lingkungan Amazon Redshift Anda.

Topik

- Melihat datashare di Amazon Redshift
- Mengedit data share yang dibuat di akun Anda di Amazon Redshift
- Menghapus otorisasi dari datashare di Amazon Redshift
- Menghapus objek datashare dari datashare di Amazon Redshift
- Menghapus konsumen data dari datashare di Amazon Redshift
- Menghapus data yang dibuat di akun Anda di Amazon Redshift

Melihat datashare di Amazon Redshift

Anda dapat melihat data share dari konsol atau dengan SQL.

Console

Anda dapat melihat data share dari tab Datashares atau Clusters.

- Gunakan tab Datashares untuk mencantumkan data share di akun Anda atau dari akun lain.
  - Untuk melihat data share yang dibuat di akun Anda, pilih Di akun saya, lalu pilih data yang ingin Anda lihat.

- Untuk melihat data share yang dibagikan dari akun lain, pilih Dari akun lain, lalu pilih data yang ingin Anda lihat.
- Gunakan tab Cluster untuk membuat daftar data share di cluster Anda atau dari cluster lain.

  Pertama, hubungkan ke database. Kemudian, pilih datashare baik dari Datashares dari cluster lain atau Datashares yang dibuat di bagian cluster saya untuk melihat detailnya.

SQL

Anda dapat membuat daftar data share yang dibuat di cluster dan melihat ke dalam konten datashare.

Contoh berikut menampilkan informasi dari sebuah nama datashare. `salesshare`

```
DESC DATASHARE salesshare;

 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name          |   include_new
-------------------+--------------------------------------+------------+------------
+-------------+--------------------------------+------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_users_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_venue_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_category_redshift|
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_date_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_event_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_listing_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_sales_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | schema      | public                         |  t
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | view        | public.sales_data_summary_view |
```

Contoh berikut menampilkan data share keluar dalam cluster produsen.

```
SHOW DATASHARES LIKE 'sales%';
```

Output-nya akan terlihat serupa dengan yang berikut ini.

```
share_name | share_owner  | source_database | consumer_database | share_type |
  createdate      | is_publicaccessible  | share_acl | producer_account |
 producer_namespace
-----------+--------------+-----------------+-------------------+------------
+---------------------+----------------------+-----------+-----------------
+--------------------------------------
salesshare |    100       | dev             |                   |  OUTBOUND
 | 2020-12-09 02:27:08 |          True        |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

Untuk informasi selengkapnya, lihat DESKRIPSI DATA dan TAMPILKAN DATA SHARE.

Anda juga dapat menggunakanSVV_DATASHARES,SVV_DATASHARE_KONSUMEN, dan SVV_DATASHARE_OBJECT untuk melihat data share, objek di dalam datashare, dan konsumen datashare.

Mengedit data share yang dibuat di akun Anda di Amazon Redshift

Anda dapat mengedit data yang dibuat di akun Anda menggunakan konsol dan dengan SQL.

Console

Di konsol, lakukan langkah-langkah berikut untuk terhubung ke database terlebih dahulu untuk melihat daftar data share yang dibuat di akun Anda.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares.
4. Di bagian Datashares yang dibuat di akun saya, pilih Hubun gkan ke database.
5. Pilih data yang ingin Anda edit, lalu pilih Edit. Halaman detail datashare muncul.
6. Buat perubahan apa pun di objek Datashare atau bagian Konsumen Data.
7. Pilih Simpan perubahan. Amazon Redshift memperbarui data Anda dengan perubahan.

> **Note**
>
> Jika Anda memilih untuk mempublikasikan datashare ke AWS Glue Data Catalog, Anda tidak dapat mengedit konfigurasi untuk menerbitkan datashare ke akun Amazon Redshift lainnya.

SQL

Gunakan ALTER DATASHARE untuk menghapus objek dari datashares kapan saja dari datashare. Untuk menghapus skema, gunakan perintah berikut:

```
ALTER DATASHARE salesshare REMOVE SCHEMA PUBLIC;
```

Untuk menghapus tabel, gunakan perintah berikut:

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

Gunakan REVOKE USAGE ON untuk mencabut izin pada datashare ke konsumen tertentu. Ini mencabut izin USAGE pada objek dalam datashare dan langsung menghentikan akses ke semua cluster konsumen. Mendaftarkan berbagi data dan kueri metadata, seperti daftar database dan tabel, tidak mengembalikan objek bersama setelah akses dicabut. Cabut akses ke datashare dari ruang nama jika Anda tidak ingin berbagi data dengan konsumen lagi.

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Jika Anda tidak ingin berbagi data dengan konsumen lagi, cabut akses ke datashare dari: Akun AWS

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012';
```

Menghapus otorisasi dari datashare di Amazon Redshift

Dengan Amazon Redshift, Anda dapat mengontrol akses ke data share dengan mencabut otorisasi untuk konsumen tertentu. Bagian ini memberikan petunjuk untuk mencabut akses konsumen ke data share Anda di Amazon Redshift.

> **Note**
>
> Untuk menghapus otorisasi untuk datashare, setidaknya harus ada satu konsumen data ditambahkan ke datashare.

Console

Pilih satu atau beberapa cluster konsumen yang ingin Anda hapus otorisasi. Kemudian, pilih H apus otorisasi.

Setelah otorisasi dihapus, konsumen data segera kehilangan akses ke datashare.

API

Administrator keamanan produsen menentukan yang berikut:

- Apakah akun lain dapat memiliki akses ke datashare atau tidak.
- Jika akun memiliki akses ke datashare, apakah akun tersebut memiliki izin menulis atau tidak.

Izin IAM berikut diperlukan untuk membatalkan otorisasi datashare:

pergeseran merah: DeauthorizeDataShare

Anda dapat membatalkan otorisasi penggunaan dan penulisan menggunakan panggilan CLI atau dengan API:

```
deauthorize-data-share
--data-share-arn <value>
--consumer-identifier <value>
```

Untuk informasi selengkapnya tentang perintah, lihat deauthor ize-data-share.

Menghapus objek datashare dari datashare di Amazon Redshift

Anda dapat menghapus satu atau lebih objek dari datashare dengan menggunakan prosedur berikut.

Console

Untuk menghapus satu atau lebih objek dari datashare pada consoe, ikuti langkah-langkah ini.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares.
4. Di bagian Datashares yang dibuat di akun saya, pilih Hubungkan ke database. Untuk informasi selengkapnya, lihat Menghubungkan ke basis data.
5. Pilih data yang ingin Anda edit, lalu pilih Edit. Halaman detail datashare muncul.
6. Untuk menghapus satu atau beberapa objek datashare ke datashare, lakukan salah satu hal berikut:

   - Untuk menghapus skema dari datashare, pilih satu atau beberapa skema. Kemudian pilih Hapus. Amazon Redshift menghapus skema yang ditentukan dan semua objek dari skema yang ditentukan dari datashare.
   - Untuk menghapus tabel dan tampilan dari datashare, pilih satu atau lebih tabel dan tampilan. Kemudian pilih Hapus. Atau, pilih Hapus berdasarkan skema untuk menghapus semua tabel dan tampilan dalam skema yang ditentukan.
   - Untuk menghapus fungsi yang ditentukan pengguna dari datashare, pilih satu atau beberapa fungsi yang ditentukan pengguna. Kemudian pilih Hapus. Atau, pilih Hapus berdasarkan skema untuk menghapus semua fungsi yang ditentukan pengguna dalam skema yang ditentukan.

SQL

Gunakan ALTER DATASHARE untuk menghapus objek dari datashares kapan saja dari datashare. Untuk menghapus skema, gunakan perintah berikut:

```
ALTER DATASHARE salesshare REMOVE SCHEMA PUBLIC;
```

Untuk menghapus tabel, gunakan perintah berikut:

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

Menghapus konsumen data dari datashare di Amazon Redshift

Anda dapat menghapus satu atau lebih konsumen data dari datashare. Konsumen data dapat berupa ruang nama yang secara unik mengidentifikasi cluster atau akun Amazon Redshift. AWS

Console

Untuk menghapus satu atau lebih konsumen data dari datashare di konsol, pilih satu atau lebih konsumen data baik dari ID namespace atau akun. AWS Kemudian, pilih H apus.

Amazon Redshift menghapus konsumen data tertentu dari datashare. Mereka kehilangan akses ke datashare segera.

SQL

Gunakan REVOKE USAGE ON untuk mencabut izin pada datashare ke konsumen tertentu. Ini mencabut izin USAGE pada objek dalam datashare dan langsung menghentikan akses ke semua cluster konsumen. Mendaftarkan berbagi data dan kueri metadata, seperti daftar database dan tabel, tidak mengembalikan objek bersama setelah akses dicabut. Cabut akses ke datashare dari ruang nama jika Anda tidak ingin berbagi data dengan konsumen lagi.

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Menghapus data yang dibuat di akun Anda di Amazon Redshift

Anda dapat menghapus data yang dibuat di akun Anda menggunakan konsol atau dengan SQL.

Console

Untuk menghapus data yang dibuat di akun Anda menggunakan konsol, sambungkan terlebih dahulu ke database untuk melihat daftar data share yang dibuat di akun Anda.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih cluster Anda. Halaman detail cluster muncul.
3. Pilih Datashares. Daftar datashare muncul.
4. Di bagian Datashares yang dibuat di akun saya, pilih Hubun gkan ke database.
5. Pilih satu atau beberapa data share yang ingin Anda hapus, lalu pilih Hapus. Halaman Hapus datashares muncul.

   Menghapus datashare yang dibagikan dengan Lake Formation tidak secara otomatis menghapus izin terkait di Formasi Danau. Untuk menghapusnya, buka konsol Formasi Danau.
6. Ketik H apus untuk mengonfirmasi penghapusan data share yang ditentukan.

7. Pilih Hapus.

Setelah data share dihapus, konsumen datashare kehilangan akses ke datashares.

SQL

Anda dapat menggunakan SQL untuk menghapus objek datashare di setiap titik menggunakan. DROP DATA Superuser cluster dan pemilik datashare dapat menghapus datashares.

Contoh berikut menjatuhkan nama datashare. `salesshare`

```
DROP DATASHARE salesshare;
```

Tindakan konsumen untuk berbagi data yang ada di Amazon Redshift

Dengan Amazon Redshift, Anda dapat mengelola berbagi data yang ada untuk mengontrol akses ke data Anda di cluster Amazon Redshift. Bagian berikut memberikan panduan langkah demi langkah tentang mengelola berbagi data di lingkungan Amazon Redshift Anda sebagai administrator konsumen.

Topik

- Mengelola izin untuk berbagi data di Amazon Redshift
- Menghapus asosiasi datashare dari konsumen data di tempat yang berbeda Akun AWS di Amazon Redshift
- Menolak datashare dari yang berbeda Akun AWS di Amazon Redshift

Mengelola izin untuk berbagi data di Amazon Redshift

Sebagai administrator produsen, Anda mempertahankan kontrol untuk kumpulan data yang Anda bagikan. Anda dapat menambahkan objek baru ke atau menghapusnya dari datashare. Anda juga dapat memberikan atau mencabut akses ke berbagi data secara keseluruhan untuk cluster konsumen, AWS akun, atau Wilayah. AWS Ketika izin dicabut, cluster konsumen segera kehilangan akses ke objek bersama dan berhenti melihatnya dalam daftar inbound datashares di SVV_DATASHARES.

Contoh berikut membuat datashare`salesshare`, menambahkan skema`public`, dan menambahkan tabel `public.tickit_sales_redshift` ke. `salesshare` Ini juga memberikan izin penggunaan `salesshare` ke namespace yang ditentukan.

```
CREATE DATASHARE salesshare;

ALTER DATASHARE salesshare ADD SCHEMA public;

ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;

GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Untuk CREATE DATASHARE, pengguna super dan pemilik database dapat membuat datashares. Untuk informasi selengkapnya, lihat BUAT DATASHARE. Untuk ALTER DATASHARE, pemilik datashare dengan izin yang diperlukan pada objek datashare yang akan ditambahkan atau dihapus dapat mengubah datashare. Untuk informasi, lihat ALTER DATA.

Sebagai administrator produsen, ketika Anda menghapus datashare, itu berhenti terdaftar di cluster konsumen. Basis data dan referensi skema yang dibuat pada cluster konsumen dari data yang dijatuhkan terus ada tanpa objek di dalamnya. Administrator konsumen harus menghapus database ini secara manual.

Di sisi konsumen, administrator konsumen dapat menentukan pengguna dan peran mana yang harus mendapatkan akses ke data bersama dengan membuat database dari datashare. Tergantung pada opsi yang Anda pilih saat membuat database, Anda dapat mengontrol akses ke sana sebagai berikut. Untuk informasi selengkapnya tentang membuat database dari datashare, lihat. BUAT BASIS DATA

Untuk informasi selengkapnya tentang menyiapkan datashare dan membaca data dari konsumen, lihat Berbagi akses baca ke data dalam akun AWS .

Membuat database tanpa klausa WITH PERMISSIONS

Administrator dapat mengontrol akses pada tingkat database atau skema. Untuk mengontrol akses di tingkat skema, administrator harus membuat skema eksternal dari database Amazon Redshift yang dibuat dari datashare.

Contoh berikut memberikan izin untuk mengakses tabel bersama di tingkat database dan tingkat skema.

```
GRANT USAGE ON DATABASE sales_db TO Bob;

CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE sales_db SCHEMA 'public';
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

Untuk membatasi akses lebih lanjut, Anda dapat membuat tampilan di atas objek bersama, hanya mengekspos data yang diperlukan. Anda kemudian dapat menggunakan tampilan ini untuk memberikan akses ke pengguna dan peran.

Setelah pengguna diberikan akses ke database atau skema, mereka akan memiliki akses ke semua objek bersama dalam database atau skema itu.

Membuat database dengan klausa WITH PERMISSIONS

Setelah memberikan hak penggunaan pada database atau skema, administrator dapat mengontrol akses lebih lanjut menggunakan proses pemberian izin yang sama seperti pada database atau skema lokal. Tanpa izin objek individual, pengguna tidak dapat mengakses objek apa pun dalam database atau skema yang dibagikan data bahkan setelah diberikan izin USAGE.

Contoh berikut memberikan izin untuk mengakses tabel bersama di tingkat database.

```
GRANT USAGE ON DATABASE sales_db TO Bob;
GRANT USAGE FOR SCHEMAS IN DATABASE sales_db TO Bob;
GRANT SELECT ON sales_db.public.tickit_sales_redshift TO Bob;
```

Setelah diberikan akses ke database atau skema, pengguna masih perlu diberi izin yang relevan untuk objek apa pun dalam database atau skema yang Anda ingin mereka akses.

Berbagi granular menggunakan WITH PERMISSIONS

Anda dapat menggunakan berbagi granular menggunakan WITH PERMISSIONS untuk mengaktifkan cluster atau kelompok kerja Tanpa Server untuk menanyakan datashare. Proses ini mengasumsikan datashare berasal dari cluster lain atau namespace Amazon Redshift Serverless di akun Anda, atau berasal dari akun lain dan telah dikaitkan dengan namespace yang Anda gunakan.

1. Administrator database konsumen dapat membuat database dari datashare.

   ```
   CREATE DATABASE my_ds_db [WITH PERMISSIONS] FROM DATASHARE my_datashare OF
    NAMESPACE 'abc123def';
   ```

   Jika Anda membuat database DENGAN IZIN, Anda dapat memberikan izin granular pada objek datashare ke pengguna dan peran yang berbeda. Tanpa ini, semua pengguna dan peran yang

diberikan izin USAGE pada database datashare diberikan semua izin pada semua objek dalam database datashare.

2. Berikut ini menunjukkan cara memberikan izin kepada pengguna atau peran database Redshift. Anda harus terhubung ke database lokal untuk menjalankan pernyataan ini. Anda tidak dapat menjalankan pernyataan ini jika Anda menjalankan perintah USE pada database datashare sebelum menjalankan pernyataan hibah.

```
GRANT USAGE ON DATABASE my_ds_db TO ROLE data_eng;
GRANT CREATE, USAGE ON SCHEMA my_ds_db.my_shared_schema TO ROLE data_eng;
GRANT ALL ON ALL TABLES IN SCHEMA my_ds_db.my_shared_schema TO ROLE data_eng;

GRANT USAGE ON DATABASE my_ds_db TO bi_user;
GRANT USAGE ON SCHEMA my_ds_db.my_shared_schema TO bi_user;
GRANT SELECT ON my_ds_db.my_shared_schema.table1 TO bi_user;
```

Menghapus asosiasi datashare dari konsumen data di tempat yang berbeda Akun AWS di Amazon Redshift

Dengan Amazon Redshift, Anda dapat menghapus asosiasi dari berbagi data yang dibagikan oleh orang lain. Akun AWS Datashares adalah objek database yang dapat dibagikan yang merangkum data dari satu atau lebih database Redshift. Bagian berikut menunjukkan proses pemisahan data berbagi dalam lingkungan Redshift Anda.

Console

Sebagai administrator konsumen, Anda dapat menghapus asosiasi berbagi data dari konsumen data di konsol.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Datashares. Halaman daftar datashare muncul.
3. Pilih Dari akun lain.
4. Di bagian Ber bagi data dari akun lain, pilih datashare untuk menghapus as osiasi dari konsumen data.
5. Di bagian Konsumen data, pilih satu atau lebih konsumen data untuk menghapus asosiasi. Kemudian pilih H apus asosiasi.
6. Saat halaman Hapus asosiasi muncul, pilih H apus asosiasi.

Setelah asosiasi dihapus, konsumen data akan kehilangan akses ke datashare. Anda dapat mengubah asosiasi konsumen data kapan saja.

SQL

> **Note**
>
> Langkah-langkah di bagian ini dilakukan setelah administrator produsen memberikan tindakan spesifik pada objek database bersama dan, jika datashare sedang dibagikan dengan akun lain, administrator keamanan produsen mengotorisasi akses.

Administrator keamanan konsumen dapat memutuskan hubungan datashare dengan perintah berikut:

```
disassociate-data-share-consumer
--data-share-arn <value>
```

Untuk informasi selengkapnya tentang perintah, lihat disassoci ate-data-share-consumer.

Menolak datashare dari yang berbeda Akun AWS di Amazon Redshift

Dengan Amazon Redshift, Anda dapat menolak berbagi data yang dibagikan oleh AWS akun lain, memungkinkan berbagi data yang lancar dan aman melintasi batas-batas organisasi. Datashares adalah objek database yang dapat dibagikan yang merangkum data dari satu atau lebih database Amazon Redshift.

Sebagai administrator konsumen, Anda dapat menolak data apa pun yang statusnya tersedia atau aktif. Setelah Anda menolak datashare, pengguna cluster konsumen kehilangan akses ke datashare. Amazon Redshift tidak mengembalikan data yang ditolak jika Anda memanggil operasi API. `DescribeDataSharesForConsumer` Jika administrator produsen menjalankan operasi `DescribeDataSharesForProducer` API, mereka akan melihat bahwa datashare ditolak. Setelah datashare ditolak, administrator produsen dapat mengotorisasi datashare ke cluster konsumen lagi, dan administrator konsumen dapat memilih untuk mengaitkan AWS akun mereka dengan datashare atau menolaknya.

Jika AWS akun Anda memiliki asosiasi ke datashare dan asosiasi tertunda ke datashare yang dikelola oleh Lake Formation, menolak asosiasi datashare yang dikelola oleh Lake Formation juga

menolak datashare asli. Untuk menolak asosiasi tertentu, administrator produsen dapat menghapus otorisasi dari data yang ditentukan. Tindakan ini tidak memengaruhi berbagi data lainnya.

Untuk menolak datashare, gunakan AWS konsol, operasi API`RejectDataShare`, atau `reject-datashare` di. AWS CLI

Console

Untuk menolak datashare menggunakan AWS konsol, lakukan langkah-langkah berikut.

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Di menu navigasi, pilih Datashares.
3. Pilih Dari akun lain.
4. Di bagian Datashares dari akun lain, pilih data yang ingin Anda tolak. Saat halaman Dec line datashare muncul, pilih Decline.

Setelah Anda menolak berbagi data, Anda tidak dapat mengembalikan perubahan. Amazon Redshift menghapus data share dari daftar. Untuk melihat datashare lagi, administrator produsen harus mengizinkannya lagi.

CLI

Untuk menolak datashare, administrator menggunakan perintah berikut:

```
reject-data-share
--data-share-arn <value>
```

Untuk informasi selengkapnya tentang perintah, lihat rejec t-data-share.

Memulai berbagi data dengan AWS CloudFormation di Amazon Redshift

Anda dapat mengotomatiskan pengaturan berbagi data dengan menggunakan AWS CloudFormation tumpukan, yang menyediakan AWS sumber daya. Tum CloudFormation pukan mengatur berbagi data antara dua cluster Amazon Redshift di AWS akun yang sama. Dengan demikian, Anda dapat memulai berbagi data tanpa menjalankan pernyataan SQL untuk menyediakan sumber daya Anda.

Tumpukan membuat datashare pada cluster yang Anda tentukan. Datashare mencakup tabel dan sampel data read-only. Data ini dapat dibaca oleh cluster Amazon Redshift Anda yang lain.

Jika Anda ingin mulai berbagi data di AWS akun dengan menjalankan pernyataan SQL untuk mengatur datashare dan memberikan izin, tanpa menggunakan CloudFormation, lihat. Berbagi akses baca ke data dalam Akun AWS

Sebelum menjalankan CloudFormation tumpukan berbagi data, Anda harus masuk dengan pengguna yang memiliki izin untuk membuat peran IAM dan fungsi Lambda. Anda juga memerlukan dua cluster Amazon Redshift di akun yang sama. Anda menggunakan satu, produsen, untuk berbagi data sampel, dan yang lainnya, konsumen, untuk membacanya. Persyaratan utama untuk cluster ini adalah bahwa masing-masing menggunakan node RG atau RA3. Untuk keperluan tambahan, lihat Pertimbangan untuk berbagi data di Amazon Redshift.

Untuk informasi selengkapnya tentang memulai menyiapkan cluster Amazon Redshift, lihat Mem ulai dengan gudang data yang disediakan Amazon Redshift. Untuk informasi selengkapnya tentang mengotomatisasi pengaturan dengan CloudFormation, lihat Apa itu? AWS CloudFormation

> **Important**
>
> Sebelum meluncurkan CloudFormation tumpukan Anda, pastikan Anda memiliki dua cluster Amazon Redshift di akun yang sama dan kluster menggunakan node RG atau RA3. Pastikan setiap cluster memiliki database dan superuser. Untuk informasi selengkapnya, lihat BUAT BASIS DATA dan superuser.

Untuk meluncurkan CloudFormation tumpukan Anda untuk berbagi data Amazon Redshift:

1. Pilih Luncurkan tumpukan CFN , yang membawa Anda ke CloudFormation layanan di Konsol Manajemen AWS.

   Jika Anda diminta, masuk.

   Proses pembuatan tumpukan dimulai, merujuk file CloudFormation template, yang disimpan di Amazon S3. CloudFormation Template adalah file teks dalam format JSON yang menyatakan AWS sumber daya yang membentuk tumpukan. Untuk informasi selengkapnya tentang CloudFormation template, lihat Mem pelajari dasar-dasar template.
2. Pilih Berikutnya untuk memasukkan detail tumpukan.
3. Di bawah Param eter, untuk setiap cluster, masukkan yang berikut:

   - Nama cluster Amazon Redshift Anda, misalnya **`ra3-consumer-cluster`**
   - Nama database Anda, misalnya **`dev`**

- Nama pengguna database Anda, misalnya **`consumeruser`**

Sebaiknya gunakan cluster uji, karena tumpukan membuat beberapa objek database.

Pilih Berikutnya.

4. Opsi tumpukan muncul.

   Pilih Berikutnya untuk menerima pengaturan default.

5. Di bawah Kemampuan, pilih Saya mengakui bahwa AWS CloudFormation mungkin membuat sumber daya IAM.
6. Pilih Buat tumpukan.

CloudFormation membutuhkan waktu sekitar 10 menit untuk membangun tumpukan Amazon Redshift menggunakan template, membuat datashare yang disebut. `myproducer_share` Tumpukan membuat datashare dalam database yang ditentukan dalam detail tumpukan. Hanya objek dari database itu yang dapat dibagikan.

Jika terjadi kesalahan saat tumpukan dibuat, lakukan hal berikut:

- Pastikan Anda memasukkan nama cluster, nama database, dan nama pengguna database yang benar untuk setiap cluster Redshift.
- Pastikan cluster Anda memiliki node RG atau RA3.
- Pastikan Anda masuk dengan pengguna yang memiliki izin untuk membuat peran IAM dan fungsi Lambda. Untuk informasi selengkapnya tentang membuat peran IAM, lihat Membuat peran IAM. Untuk informasi selengkapnya tentang kebijakan pembuatan fungsi Λ, lihat Peng embangan fungsi.

Menanyakan data yang Anda buat

Untuk menggunakan prosedur berikut, pastikan Anda memiliki izin yang diperlukan untuk menjalankan kueri pada setiap cluster yang dijelaskan.

Untuk menanyakan data Anda:

1. Hubungkan ke cluster produsen pada database yang dimasukkan saat CloudFormation tumpukan Anda dibuat, menggunakan alat klien seperti editor kueri Amazon Redshift v2.

2. Kueri untuk berbagi data.

```
SHOW DATASHARES;

+-----------------+-------------+-----------------+------------------
+------------+------------+--------------------+-----------+-----------------
+-------------------------------------+
|    share_name    | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace          |
+-----------------+-------------+-----------------+------------------
+------------+------------+--------------------+-----------+-----------------
+-------------------------------------+
| myproducer_share | 100         | sample_data_dev | myconsumer_db     | INBOUND
  | NULL       | true               | NULL      |  producer-acct   |        your-
producer-namespace       |
+-----------------+-------------+-----------------+------------------
+------------+------------+--------------------+-----------+-----------------
+-------------------------------------+
```

Perintah sebelumnya mengembalikan nama datashare yang dibuat oleh tumpukan, disebut. `myproducer_share` Ini juga mengembalikan nama database yang terkait dengan datashare,. `myconsumer_db`

Salin pengidentifikasi namespace produsen untuk digunakan pada langkah selanjutnya.

3. Jelaskan objek di datashare.

```
DESC DATASHARE myproducer_share;

+-----------------+-------------------------------------+-----------
+-----------------+-------------+-------------------------------------
+-------------+
| producer_account |          producer_namespace          | share_type |
 share_name    | object_type |             object_name             | include_new |
+-----------------+-------------------------------------+-----------
+-----------------+-------------+-------------------------------------
+-------------+
|   producer-acct  |        your-producer-namespace       | OUTBOUND   |
 myproducer_share | schema      | myproducer_schema                   | true
 |
```

```
|   producer-acct  |        your-producer-namespace       | OUTBOUND   |
 myproducer_share | table       | myproducer_schema.tickit_sales      | NULL
 |
|   producer-acct  |        your-producer-namespace       | OUTBOUND   |
 myproducer_share | view        | myproducer_schema.ticket_sales_view | NULL
 |
+------------------+--------------------------------------+------------
+------------------+-------------+-------------------------------------
+-------------+
```

Ketika Anda mendeskripsikan datashare, ia mengembalikan properti untuk tabel dan tampilan. Tumpukan menambahkan tabel dan tampilan dengan data sampel ke database produsen, misalnya `tickit_sales` dan`tickit_sales_view`. Untuk informasi lebih lanjut tentang database sampel TICKIT, lihatDatabase sampel.

Anda tidak perlu mendelegasikan izin pada datashare untuk menjalankan kueri. Tumpukan memberikan izin yang diperlukan.

4. Hubungkan ke cluster konsumen menggunakan alat klien Anda. Jelaskan datashare, menentukan namespace produsen.

```
DESC DATASHARE myproducer_share OF NAMESPACE '<namespace id>'; --specify the unique
 identifier for the producer namespace

+------------------+--------------------------------------+------------
+------------------+-------------+-------------------------------------
+-------------+
| producer_account |          producer_namespace          | share_type |
 share_name    | object_type |             object_name             | include_new |
+------------------+--------------------------------------+------------
+------------------+-------------+-------------------------------------
+-------------+
|   producer-acct  |        your-producer-namespace       | INBOUND    |
 myproducer_share | schema      | myproducer_schema                   | NULL
 |
|   producer-acct  |        your-producer-namespace       | INBOUND    |
 myproducer_share | table       | myproducer_schema.tickit_sales      | NULL
 |
|   producer-acct  |        your-producer-namespace       | INBOUND    |
 myproducer_share | view        | myproducer_schema.ticket_sales_view | NULL
 |
```

```
+-----------------+---------------------------------------+-----------
+-----------------+-------------+-----------------------------------
+-------------+
```

5. Anda dapat menanyakan tabel di datashare dengan menentukan database dan skema datashare. Untuk informasi selengkapnya, lihat Cross-database contoh kueri. Kueri berikut mengembalikan data penjualan dan penjual dari tabel PENJUALAN di database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

```
SELECT * FROM myconsumer_db.myproducer_schema.tickit_sales_view;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |       728 |
     109.2 | 2008-02-18 02:36:48 |
|       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |        76 |
      11.4 | 2008-06-06 05:00:16 |
|       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |       350 |
      52.5 | 2008-06-06 08:26:17 |
|       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |       175 |
     26.25 | 2008-06-09 08:38:52 |
|       5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |       154 |
      23.1 | 2008-08-31 09:17:02 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

> **Note**
>
> Kueri berjalan melawan tampilan dalam skema bersama. Anda tidak dapat terhubung langsung ke database yang dibuat dari datashares. Mereka hanya baca.

6. Untuk menjalankan kueri yang mencakup agregasi, gunakan contoh berikut.

```
SELECT * FROM myconsumer_db.myproducer_schema.tickit_sales ORDER BY 1,2 LIMIT 5;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

```
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |       728 |
     109.2 | 2008-02-18 02:36:48 |
|       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |        76 |
      11.4 | 2008-06-06 05:00:16 |
|       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |       350 |
      52.5 | 2008-06-06 08:26:17 |
|       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |       175 |
     26.25 | 2008-06-09 08:38:52 |
|       5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |       154 |
      23.1 | 2008-08-31 09:17:02 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

Kueri mengembalikan data penjualan dan penjual dari sampel data TICKIT.

Untuk contoh lebih lanjut dari kueri datashare, lihat. Berbagi akses baca ke data dalam Akun AWS

Jenis berbagi data di Amazon Redshift

Datashare adalah unit berbagi data di Amazon Redshift. Gunakan datashares untuk berbagi data dalam hal yang sama Akun AWS atau berbeda. Akun AWS Juga, bagikan data untuk tujuan membaca di berbagai cluster Amazon Redshift.

Setiap datashare dikaitkan dengan database tertentu di cluster Amazon Redshift Anda.

Administrator produsen dapat membuat data share dan menambahkan objek datashare untuk berbagi data dengan cluster lain, yang disebut sebagai pembagian keluar. Administrator konsumen dapat menerima data share dari cluster lain, yang disebut sebagai in bound share. Untuk detail tentang produsen dan konsumen, lihatProdusen dan konsumen Datashare.

Objek Datashare adalah objek dari database tertentu pada cluster yang dapat ditambahkan administrator produsen ke berbagi data untuk dibagikan dengan konsumen data. Objek datashare bersifat hanya baca untuk konsumen data. Contoh objek datashare adalah tabel, tampilan, dan fungsi yang ditentukan pengguna. Anda dapat menambahkan objek datashare ke datashares saat membuat datashares atau mengedit datashare kapan saja.

Berbagi data terus berfungsi saat cluster diubah ukurannya atau saat cluster produsen dijeda.

Ada berbagai jenis berbagi data: berbagi data standar, datashares, dan data yang dikel AWS Data Exchange ola. AWS Lake Formation Halaman-halaman berikut memberikan ikhtisar masing-masing.

Berbagi data standar

Dengan berbagi data standar, Anda dapat berbagi data di seluruh cluster yang disediakan, grup kerja tanpa server, Zona Ketersediaan, dan. Akun AWS Wilayah AWS Anda dapat berbagi antar tipe cluster serta antara cluster yang disediakan dan Amazon Redshift Serverless.

Untuk berbagi data, perhatikan cluster yang disediakan, namespace tanpa server, dan pengidentifikasi berikut ini: Akun AWS

- Ruang nama yang disediakan adalah pengidentifikasi yang mengidentifikasi cluster yang disediakan Amazon Redshift. Pengidentifikasi unik global (GUID) namespace dibuat secara otomatis selama pembuatan cluster yang disediakan dan dilampirkan ke cluster. Namespace Amazon Resource Name (ARN) berada dalam format arn: {partition} :redshift: {region}: {account-id} :namespace: {namespace-guid}. Anda dapat melihat namespace cluster yang disediakan di halaman detail cluster di konsol Amazon Redshift.

  Dalam alur kerja berbagi data, nilai GUID namespace dan namespace ARN digunakan untuk berbagi data dengan cluster di. Akun AWS Anda juga dapat menemukan namespace untuk cluster saat ini dengan menggunakan `current_namespace` fungsi.
- Ruang nama tanpa server adalah pengidentifikasi yang mengidentifikasi Amazon Redshift Serverless. Pengidentifikasi unik global (GUID) namespace dibuat secara otomatis selama pembuatan Amazon Redshift Serverless dan dilampirkan ke instance. ARN namespace tanpa server berada dalam format arn: {partition} :redshift-serverless: {region}: {account-id} :namespace/{namespace-guid}.
- Akun AWS dapat menjadi konsumen untuk berbagi data dan masing-masing diwakili oleh ID 12 Akun AWS digit.

Untuk berbagi data standar, pertimbangkan hal berikut:

- Saat cluster produsen dihapus, Amazon Redshift menghapus data share yang dibuat oleh cluster produsen. Ketika cluster produsen dicadangkan dan dipulihkan, berbagi data yang dibuat masih tetap ada di cluster yang dipulihkan. Jika pemulihan dilakukan pada namespace tanpa server yang sama, izin datashare mungkin dipertahankan. Lihat Pertimbangan untuk berbagi data di Amazon

Redshift Serverless restore skenario yang didukung. Untuk skenario lain, izin datashare yang diberikan kepada cluster konsumen tidak lagi valid pada cluster yang dipulihkan. Re-grant izin penggunaan data share ke cluster konsumen yang diinginkan. Basis data konsumen pada cluster konsumen menunjuk ke datashare dari cluster asli tempat snapshot diambil. Untuk menanyakan data bersama dari cluster yang dipulihkan, administrator konsumen membuat database yang berbeda. Atau administrator dapat menghapus dan membuat ulang database konsumen yang ada untuk menggunakan datashare dari cluster yang baru dipulihkan.

- Ketika cluster konsumen dihapus dan dipulihkan dari snapshot, akses sebelumnya yang dibagikan ke cluster ini tidak lagi valid dan terlihat. Jika akses ke berbagi data masih diperlukan pada cluster konsumen yang dipulihkan, administrator produsen harus memberikan penggunaan data share ke cluster konsumen yang dipulihkan lagi. Administrator konsumen harus menghapus database konsumen basi yang dibuat dari berbagi data yang tidak aktif. Kemudian administrator harus membuat ulang database konsumen dari datashare, setelah produsen memberikan kembali izin. Karena GUID namespace berbeda pada cluster yang dipulihkan dari cluster asli, berikan kembali izin datashare saat cluster konsumen atau produsen dipulihkan dari cadangan.

AWS Data Exchange berbagi data

Anda dapat menggunakan AWS Data Exchange data share untuk mengelola penagihan untuk berbagi data Amazon Redshift.

AWS Data Exchange Datashare adalah unit lisensi untuk berbagi data Anda melalui. AWS Data Exchange AWS mengelola semua tagihan dan pembayaran yang terkait dengan langganan AWS Data Exchange dan penggunaan berbagi data Amazon Redshift. Penyedia data yang disetujui dapat menambahkan AWS Data Exchange data share ke AWS Data Exchange produk. Ketika pelanggan berlangganan produk dengan AWS Data Exchange datashares, mereka mendapatkan akses ke data share dalam produk.

AWS Data Exchange untuk Amazon Redshift memudahkan untuk melisensikan akses ke data Amazon Redshift Anda melalui. AWS Data Exchange Ketika pelanggan berlangganan produk dengan AWS Data Exchange datashares, AWS Data Exchange secara otomatis menambahkan pelanggan sebagai konsumen data pada semua AWS Data Exchange data share yang disertakan dengan produk. Faktur dibuat secara otomatis, dan pembayaran dikumpulkan secara terpusat dan dicairkan secara otomatis. AWS Marketplace Entitlement Service

Penyedia dapat melisensikan data di Amazon Redshift pada tingkat granular, seperti skema, tabel, tampilan, dan fungsi yang ditentukan pengguna. Anda dapat menggunakan data yang AWS Data

Exchange sama di beberapa AWS Data Exchange produk. Setiap objek yang ditambahkan ke AWS Data Exchange datashare tersedia untuk konsumen. Produsen dapat melihat semua AWS Data Exchange data share yang dikel AWS Data Exchange ola atas nama mereka menggunakan operasi Amazon Redshift API, perintah SQL, dan konsol Amazon Redshift. Pelanggan yang berlangganan AWS Data Exchange data share produk memiliki akses read-only ke objek di datashares.

Pelanggan yang ingin menggunakan data produsen pihak ketiga dapat menelusuri AWS Data Exchange katalog untuk menemukan dan berlangganan kumpulan data di Amazon Redshift. Setelah AWS Data Exchange langganan aktif, mereka dapat membuat database dari datashare di cluster mereka dan menanyakan data di Amazon Redshift.

Bagaimana AWS Data Exchange berbagi data bekerja

Mengelola AWS Data Exchange berbagi data sebagai administrator produser

Jika Anda adalah produsen data (juga dikenal sebagai penyedia pada AWS Data Exchange), Anda dapat membuat berbagi AWS Data Exchange data yang terhubung ke database Amazon Redshift Anda. Untuk menambahkan AWS Data Exchange data share ke produk AWS Data Exchange, Anda harus menjadi penyedia terdaftar AWS Data Exchange .

Untuk informasi lebih lanjut tentang cara memulai dengan AWS Data Exchange datashares, lihat. Berbagi data Amazon Redshift berlisensi di AWS Data Exchange

Penggunaan AWS Data Exchange berbagi data sebagai konsumen dengan aktif AWS Data Exchange langganan

Jika Anda adalah konsumen dengan AWS Data Exchange langganan aktif (juga dikenal sebagai pelanggan aktif AWS Data Exchange), Anda dapat menelusuri AWS Data Exchange katalog di AWS Data Exchange konsol untuk menemukan produk yang berisi AWS Data Exchange data share.

Setelah Anda berlangganan produk yang berisi AWS Data Exchange datashares, buat database dari datashare dalam cluster Anda. Anda kemudian dapat menanyakan data di Amazon Redshift secara langsung tanpa mengekstraksi, mengubah, dan memuat data.

Untuk informasi lebih lanjut tentang cara memulai dengan AWS Data Exchange datashares, lihat. Berbagi data Amazon Redshift berlisensi di AWS Data Exchange

Untuk AWS Data Exchange berbagi data, pertimbangkan hal berikut:

- Saat cluster produsen dihapus, Amazon Redshift menghapus data share yang dibuat oleh cluster produsen. Ketika cluster produsen dicadangkan dan dipulihkan, berbagi data yang dibuat masih

tetap ada di cluster yang dipulihkan. Agar pelanggan data dapat terus mengakses data, buat AWS Data Exchange datashares lagi dan publikasikan ke kumpulan data produk. Basis data konsumen pada cluster konsumen menunjuk ke datashare dari cluster asli tempat snapshot diambil. Untuk menanyakan data bersama dari cluster yang dipulihkan, administrator konsumen membuat database yang berbeda, atau menghapus dan membuat ulang database konsumen yang ada untuk menggunakan AWS Data Exchange datashare yang baru dibuat dari cluster yang baru dipulihkan.

- Kami menyarankan Anda untuk tidak menghapus cluster Anda jika Anda memiliki berbagi AWS Data Exchange data. Melakukan perubahan jenis ini dapat melanggar ketentuan produk data AWS Data Exchange.

Produsen dan konsumen Datashare

Produsen data (juga dikenal sebagai produsen berbagi data atau produsen datashare) adalah cluster yang ingin Anda bagikan data dari. administrator produsen dan pemilik database dapat membuat berbagi data menggunakan perintah CREATE DATASHARE. Anda dapat menambahkan objek seperti skema, tabel, tampilan, dan fungsi yang ditentukan pengguna SQL (UDF) dari database yang Anda ingin cluster produsen berbagi dengan cluster konsumen.

Produsen data (juga dikenal sebagai penyedia pada AWS Data Exchange) untuk berbagi data AWS Data Exchange dapat melisensikan data melalui. AWS Data Exchange Penyedia yang disetujui dapat AWS Data Exchange menambahkan data share ke AWS Data Exchange produk.

Ketika pelanggan berlangganan produk dengan AWS Data Exchange datashares, AWS Data Exchange secara otomatis menambahkan pelanggan sebagai konsumen data pada semua AWS Data Exchange data share yang disertakan dengan produk. AWS Data Exchange juga menghapus semua pelanggan dari AWS Data Exchange datashares saat langganan mereka berakhir. AWS Data Exchange juga secara otomatis mengelola penagihan, faktur, pengumpulan pembayaran, dan distribusi pembayaran untuk produk berbayar dengan AWS Data Exchange datashares. Untuk informasi selengkapnya, lihat AWS Data Exchange berbagi data. Untuk mendaftar sebagai penyedia AWS Data Exchange data, lihat Memulai sebagai penyedia.

Konsumen data (juga dikenal sebagai konsumen berbagi data atau konsumen datashare) adalah cluster yang menerima berbagi data dari cluster produsen.

Cluster Amazon Redshift yang berbagi data dapat berada di tempat yang sama atau berbeda Akun AWS atau berbeda Wilayah AWS, sehingga Anda dapat berbagi data di seluruh organisasi dan berkolaborasi dengan pihak lain. administrator konsumen menerima berbagi data yang diberikan

penggunaannya dan meninjau konten setiap datashare. Untuk menggunakan data bersama, administrator konsumen membuat database Amazon Redshift dari datashare. Administrator kemudian memberikan izin untuk database kepada pengguna dan peran dalam cluster konsumen. Setelah izin diberikan, pengguna dan peran dapat mencantumkan objek bersama sebagai bagian dari kueri metadata standar, bersama dengan data lokal pada cluster konsumen. Mereka dapat mulai menanyakan segera.

Jika Anda adalah konsumen dengan AWS Data Exchange langganan aktif (juga dikenal sebagai pelanggan aktif AWS Data Exchange), Anda dapat menemukan, berlangganan, dan menanyakan data terperinci dan terkini di Amazon Redshift tanpa perlu mengekstrak, mengubah, dan memuat data. Untuk informasi selengkapnya, lihat AWS Data Exchange berbagi data.

AWS Lake Formation-berbagi data yang dikelola

Dengan Amazon Redshift, Anda dapat mengakses dan berbagi data langsung di seluruh AWS akun dan cluster Amazon Redshift melalui berbagi data yang dikelola AWS Lake Formation. AWS Lake Formation datashares memungkinkan penyedia data untuk secara aman berbagi data langsung dari Amazon S3 data lake mereka dengan konsumen mana pun, termasuk AWS akun lain dan cluster Amazon Redshift.

Dengan menggunakan AWS Lake Formation, Anda dapat secara terpusat menentukan dan menerapkan izin akses tingkat basis data, tabel, kolom, dan baris dari pembagian data Amazon Redshift dan membatasi akses pengguna ke objek dalam datashare. Dengan berbagi data melalui Formasi Danau, Anda dapat menentukan izin di Formasi Danau dan menerapkan izin tersebut ke datashare dan objeknya. Misalnya, jika Anda memiliki tabel yang berisi informasi karyawan, Anda dapat menggunakan filter tingkat kolom Lake Formation untuk mencegah karyawan yang tidak bekerja di departemen SDM melihat informasi identitas pribadi (PII), seperti nomor jaminan sosial. Untuk informasi selengkapnya tentang filter data, lihat Pemfilteran data dan keamanan tingkat sel di Lake Formation di Panduan Peng AWS Lake Formation embang.

Anda juga dapat menggunakan tag di Formasi Danau untuk mengonfigurasi izin pada sumber daya Formasi Danau. Untuk informasi lebih lanjut, lihat kontrol Tag-based akses Formasi Danau.

Amazon Redshift saat ini mendukung berbagi data melalui Lake Formation saat berbagi dalam akun yang sama atau di seluruh akun. Cross-Region Sharing saat ini tidak didukung.

Berikut ini adalah ikhtisar tingkat tinggi tentang cara menggunakan Formasi Danau untuk mengontrol izin datashare:

1. Di Amazon Redshift, cluster produsen atau administrator grup kerja membuat datashare di cluster produsen atau kelompok kerja dan memberikan penggunaan ke akun Lake Formation.
2. Cluster produsen atau administrator kelompok kerja mengotorisasi akun Formasi Danau untuk mengakses datashare.
3. Administrator Formasi Danau menemukan dan mendaftarkan data share. Mereka juga harus menemukan AWS Glue ARN yang mereka akses dan mengaitkan berbagi data dengan ARN. AWS Glue Data Catalog Jika Anda menggunakan, AWS CLI Anda dapat menemukan dan menerima berbagi data dengan operasi Redshift CLI dan. `describe-data-shares` `associate-data-share-consumer` Untuk mendaftarkan datashare, gunakan operasi Lake Formation CLI. `register-resource`
4. Administrator Lake Formation membuat database federasi di AWS Glue Data Catalog, dan mengonfigurasi izin Formasi Danau untuk mengontrol akses pengguna ke objek dalam datashare. Untuk informasi selengkapnya tentang database federasi di AWS Glue, lihat M engelola izin untuk data di Amazon Redshift datashare.
5. Administrator Danau Formation menemukan AWS Glue database yang AWS Glue Data Catalog mereka akses dan mengaitkan datashare dengan ARN.
6. Administrator Redshift menemukan ARN AWS Glue database yang dapat mereka akses, membuat database eksternal di cluster konsumen Amazon Redshift menggunakan database ARN, dan memberikan penggunaan kepada pengguna AWS Glue database yang diautentikasi dengan kredentif IAM untuk mulai menanyakan database Amazon Redshift.
7. Pengguna database dapat menggunakan tampilan SVV_EXTERNAL_TABLES dan SVV_EXTERNAL_COLUMNS untuk menemukan semua tabel atau kolom dalam AWS Glue database yang mereka akses, dan kemudian mereka dapat menanyakan tabel database. AWS Glue
8. Ketika cluster produsen atau administrator kelompok kerja memutuskan untuk tidak lagi berbagi data dengan cluster konsumen, administrator produsen dapat mencabut penggunaan, mencabut otorisasi, atau menghapus datashare dari Redshift. Izin dan objek terkait dalam Formasi Danau tidak dihapus secara otomatis.

Untuk informasi selengkapnya tentang berbagi data AWS Lake Formation sebagai administrator cluster produsen atau workgroup, lihat. Bekerja dengan Lake Formation-managed datashares sebagai produser Untuk menggunakan data bersama dari cluster produsen atau kelompok kerja, lihatBekerja dengan Lake Formation-managed datashares sebagai konsumen.

Nilai status datashare di Amazon Redshift

Dengan Amazon Redshift, Anda dapat berbagi data langsung dengan aman di seluruh cluster Amazon Redshift tanpa harus menyalin atau mentransfer data. Dengan berbagi data untuk Amazon Redshift, Anda dapat membagikan hasil kueri langsung, termasuk pembaruan ke data sumber, dengan cluster Amazon Redshift apa pun di akun dan Wilayah yang sama atau berbeda AWS . AWS Topik ini menjelaskan kemungkinan status yang dapat dimiliki oleh data share di Amazon Redshift.

Dengan berbagi data lintas akun, ada berbagai status berbagi data yang memerlukan tindakan Anda. Datashare Anda dapat memiliki status aktif, tindakan diperlukan, atau tidak aktif.

Berikut ini menjelaskan setiap status datashare dan tindakan yang diperlukan:

- Ketika administrator produsen membuat datashare, status datashare pada cluster produsen adalah Pending otorisasi. Administrator produsen dapat mengotorisasi konsumen data untuk mengakses datashare. Tidak ada tindakan apa pun untuk administrator konsumen.
- Ketika administrator produsen mengotorisasi datashare, status datashare menjadi Otorisasi pada cluster produsen. Tidak ada tindakan apa pun untuk administrator produser. Ketika setidaknya ada satu asosiasi dengan konsumen data untuk datashare, status datashare berubah dari Diotorisasi menjadi Aktif.

  Status berbagi datashare kemudian menjadi Ter sedia (Tindakan diperlukan di konsol Amazon Redshift) di cluster konsumen. Administrator konsumen dapat mengaitkan datashare dengan konsumen data atau menolak datashare. Administrator konsumen juga dapat menggunakan AWS CLI perintah `describeDatashareforConsumer` untuk melihat status berbagi data. Atau administrator dapat menggunakan perintah CLI `describeDatashare` dan memberikan datashare Amazon Resource Name (ARN) untuk melihat status datashare.
- Ketika administrator konsumen mengaitkan datashare dengan konsumen data, status datashare menjadi Aktif di cluster produsen. Ketika setidaknya ada satu asosiasi dengan konsumen data untuk datashare, status datashare berubah dari Diotorisasi menjadi Aktif. Tidak ada tindakan apa pun yang diperlukan untuk administrator produsen.

  Status datashare menjadi Aktif di cluster konsumen. Tidak ada tindakan apa pun yang diperlukan untuk administrator konsumen.
- Ketika administrator konsumen menghapus asosiasi konsumen dari datashare, status datashare menjadi Aktif atau Resmi. Ini menjadi Aktif ketika setidaknya ada satu asosiasi yang ada untuk datashare dengan konsumen data lain. Itu menjadi Resmi ketika tidak ada asosiasi konsumen dengan datashare di cluster produsen. Tidak ada tindakan apa pun untuk administrator produser.

Status datashare menjadi T indakan yang diperlukan pada cluster konsumen jika semua asosiasi dihapus. Administrator konsumen dapat mengasosiasikan kembali datashare dengan konsumen data ketika datashare tersedia untuk konsumen.

- Ketika administrator konsumen menolak datashare, status datashare pada cluster produsen menjadi T indakan diperlukan dan Ditolak pada cluster konsumen. Administrator produsen dapat mengotorisasi ulang datashare. Tidak ada tindakan apa pun untuk administrator konsumen.
- Ketika administrator produsen menghapus otorisasi dari datashare, status datashare menjadi Tindakan yang diperlukan pada cluster produsen. Administrator produsen dapat memilih untuk mengotorisasi ulang data, jika perlu. Tidak ada tindakan apa pun yang diperlukan untuk administrator konsumen.

Mengelola akses ke operasi API berbagi data dengan kebijakan IAM

Untuk mengontrol akses ke operasi API berbagi data, gunakan kebijakan berbasis tindakan IAM. Untuk informasi tentang cara mengelola kebijakan IAM, lihat Mengelola kebijakan IAM di Panduan Pengguna IAM.

Untuk informasi tentang izin yang diperlukan untuk menggunakan operasi API berbagi data, lihat Izin yang diperlukan untuk menggunakan operasi API berbagi data di Panduan Manajemen Amazon Redshift.

Untuk membuat berbagi data lintas akun lebih aman, Anda dapat menggunakan kunci kondisional `ConsumerIdentifier` untuk operasi `AuthorizeDataShare` dan `DeauthorizeDataShare` API. Dengan melakukan ini, Anda dapat secara eksplisit mengontrol mana yang Akun AWS dapat melakukan panggilan ke dua operasi API.

Anda dapat menolak otorisasi atau mencabut otorisasi berbagi data untuk konsumen mana pun yang bukan akun Anda sendiri. Untuk melakukannya, tentukan Akun AWS nomor dalam kebijakan IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "VisualEditor0",
            "Effect": "Deny",
            "Action": [
                "redshift:AuthorizeDataShare",
                "redshift:DeauthorizeDataShare"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "redshift:ConsumerIdentifier": "555555555555"
                }
            }
        }
    ]
}
```

Anda dapat mengizinkan produsen dengan a DataShareArn **`testshare2`** untuk secara eksplisit berbagi dengan konsumen dengan nomor 111122223333 dalam kebijakan IAM. Akun AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "redshift:AuthorizeDataShare",
                "redshift:DeauthorizeDataShare"
            ],
            "Resource": "arn:aws:redshift:us-
east-1:666666666666:datashare:af06285e-8a45-4ee9-b598-648c218c8ff1/testshare2",
            "Condition": {
                "StringEquals": {
                    "redshift:ConsumerIdentifier": "111122223333"
                }
            }
        }
    ]
}
```

Menghubungkan ke database konsumen di Amazon Redshift

Dengan koneksi langsung ke database datashare, Anda dapat langsung terhubung ke database yang dibuat dari datashare dengan cara yang sama seperti Anda dapat terhubung ke jenis database Amazon Redshift lainnya. Misalnya, Anda dapat terhubung ke database yang dibuat dari datashare menggunakan driver JDBC atau ODBC, editor kueri Amazon Redshift v2, atau alat lain yang dapat terhubung ke database Amazon Redshift. Untuk informasi selengkapnya, lihat Menghubungkan ke gudang data Amazon Redshift menggunakan alat klien SQL.

Mengakses data bersama

Saat Anda terhubung ke database yang dibuat dari datashare, Anda dapat menanyakan objek bersama menggunakan notasi dua bagian (). *schema_name*.*table_name* Jika tabel dapat ditemukan di jalur pencarian basis data konsumen, Anda juga dapat menggunakan notasi satu bagian (*table_name*).

Jika Anda ingin melakukan kueri lintas database, Anda dapat menggunakan notasi tiga bagian ()*consumer_database_name*.*schema_name*.*table_name*. Kueri ini dapat mereferensikan objek bersama dari database konsumen lain di cluster, atau objek lokal dari database lokal. Mereka juga dapat mereferensikan database lokal dan data yang dibagikan dari cluster lain dalam kueri yang sama.

> **Note**
>
> Database yang dibuat dari datashare tidak memiliki katalog lokal. Oleh karena itu, setiap kueri yang mengakses tabel katalog lokalpg_class, seperti, mengembalikan hasil kosong.

Mengakses metadata untuk objek bersama

Untuk membantu administrator cluster menemukan objek bersama dalam database konsumen, Amazon Redshift menyediakan serangkaian tampilan metadata dan perintah SHOW yang mencantumkan metadata untuk objek ini. Saat Anda terhubung ke database konsumen, tampilan dan perintah metadata ini tidak mendukung penemuan metadata lintas database. Mereka hanya mengembalikan metadata untuk objek bersama di datashare yang terkait dengan database yang terhubung.

Gunakan SHOW SCHEMAS untuk melihat daftar skema bersama di datashare yang terkait dengan database yang terhubung. Untuk informasi selengkapnya, lihat TAMPILKAN SKEMA.

Gunakan SHOW TABLES untuk melihat daftar tabel dalam skema bersama dari data yang terkait dengan database yang terhubung. Untuk informasi selengkapnya, lihat TAMPILKAN TABEL.

Gunakan SHOW COLUMNS untuk melihat daftar kolom dari tabel bersama di datashare yang terkait dengan database yang terhubung. Untuk informasi selengkapnya, lihat TAMPILKAN KOLOM.

Gunakan SVV_ALL_SCHEMAS untuk melihat daftar skema bersama di datashare yang terkait dengan database yang terhubung. Untuk informasi selengkapnya, lihat SVV_ALL_SCHEMAS.

Gunakan SVV_ALL_TABLES untuk melihat daftar tabel bersama di datashare yang terkait dengan database yang terhubung. Untuk informasi selengkapnya, lihat SVV_ALL_TABEL.

Gunakan SVV_ALL_COLUMNS untuk melihat daftar kolom bersama di data yang terkait dengan database yang terhubung. Untuk informasi selengkapnya, lihat SVV_ALL_KOLOM.

Mengintegrasikan berbagi data Amazon Redshift dengan alat intelijen bisnis

Untuk mengintegrasikan berbagi data dengan alat intelijen bisnis (BI), sebaiknya gunakan driver Amazon Redshift JDBC atau ODBC. Driver Amazon Redshift JDBC dan ODBC mendukung operasi `GetCatalogs` API di driver. Operasi ini mengembalikan daftar semua database, termasuk yang dibuat dari datashares.

Driver juga mendukung operasi hilir, seperti `GetSchemas` dan GetTables, yang mengembalikan data dari semua database yang dikem `GetCatalogs` balikan. Driver memberikan dukungan ini bahkan ketika Anda tidak secara eksplisit menentukan katalog dalam panggilan. Untuk informasi selengkapnya tentang driver JDBC atau ODBC, lihat Mengon figurasi koneksi di Panduan Manajemen Amazon Redshift.

Editor kueri Amazon Redshift v2 menyertakan database konsumen dalam antarmuka pengalihan koneksinya. Namun, sebagian besar alat mengecualikan database ini dan hanya menyertakan database cluster lokal sebagai database yang dapat dihubungkan.

> **Note**
>
> Sebuah database sistem baru `sys:internal` bernama ditambahkan untuk pemeliharaan internal. Beberapa alat menyertakan database sistem ini sebagai database yang dapat dihubungkan. Namun, Anda tidak dapat terhubung ke sana atau menjalankan kueri terhadap objeknya.

Memantau dan mengaudit berbagi data di Amazon Redshift

Dengan Amazon Redshift, Anda dapat memantau dan mengaudit aktivitas berbagi data untuk menjaga kepatuhan dan keamanan.

Dengan mengaudit berbagi data, produsen dapat melacak evolusi datashare. Misalnya, audit membantu melacak kapan data share dibuat, objek ditambahkan atau dihapus, dan izin diberikan atau dicabut ke cluster, akun, AWS atau Wilayah Amazon Redshift. AWS

Selain audit, produsen dan konsumen melacak penggunaan datashare di berbagai granularitas, seperti level akun, cluster, dan objek. Untuk informasi selengkapnya tentang pelacakan penggunaan dan tampilan audit, lihat SVL_DATASHARE_CHANGE_LOG danSVL_DATASHARE_USAGE_PRODUCER.

Anda dapat memantau berbagi data dengan menanyakan tampilan sistem.

1. Administrator produsen yang ingin berbagi data membuat datashare Amazon Redshift. Administrator produsen kemudian menambahkan objek database yang diperlukan. Ini mungkin skema, tabel, dan tampilan ke datashare dan menentukan daftar konsumen yang akan dibagikan dengan objek.

   Gunakan tampilan sistem berikut untuk melihat tampilan konsolidasi untuk melacak perubahan dan penggunaan data share pada cluster and/or konsumen produsen:

   - SYS_DATASHARE_CHANGE_LOG
   - SYS_DATASHARE_PENGGUNAAN_KONSUMEN
   - SYS_DATASHARE_PENGGUNAAN_PRODUSEN

   Gunakan tampilan sistem berikut untuk melihat objek datashare dan informasi konsumen data untuk berbagi data keluar:

   - SVV_DATASHARES
   - SVV_DATASHARE_KONSUMEN
   - SVV_DATASHARE_OBJECT
2. Administrator konsumen melihat berbagi data yang diberikan penggunaannya dan meninjau konten setiap datashare dengan melihat data share masuk menggunakan. SVV_DATASHARES

   Untuk menggunakan data bersama, setiap administrator konsumen membuat database Amazon Redshift dari datashare. Administrator kemudian memberikan izin kepada pengguna dan peran yang sesuai di cluster konsumen. Pengguna dan peran dapat mencantumkan objek bersama

sebagai bagian dari kueri metadata standar dengan melihat tampilan sistem metadata berikut dan dapat segera mulai menanyakan data.

- SVV_REDSHIFT_KOLOM
- SVV_REDSHIFT_DATABASE
- SVV_REDSHIFT_FUNCTION
- SVV_REDSHIFT_SKEMA
- SVV_REDSHIFT_TABEL

Untuk melihat objek skema lokal dan bersama Amazon Redshift dan skema eksternal, gunakan tampilan sistem metadata berikut untuk menanyainya.

- SVV_ALL_KOLOM
- SVV_ALL_SCHEMAS
- SVV_ALL_TABEL

Saat Anda terhubung ke database konsumen, penemuan lintas database dinonaktifkan. Tampilan sistem metadata hanya mengembalikan metadata untuk objek bersama di datashare yang terkait dengan database yang terhubung.

Mengintegrasikan berbagi data Amazon Redshift dengan AWS CloudTrail

Berbagi data terintegrasi dengan AWS CloudTrail. CloudTrail adalah layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon Redshift. CloudTrail menangkap semua panggilan API untuk berbagi data sebagai peristiwa. Panggilan yang ditangkap termasuk panggilan dari AWS CloudTrail konsol dan panggilan kode ke operasi berbagi data. Untuk informasi selengkapnya tentang integrasi Amazon Redshift dengan AWS CloudTrail, lihat Login dengan CloudTrail.

Untuk informasi selengkapnya CloudTrail, lihat Cara CloudTrail kerja.

Semi-structured data dalam Amazon Redshift

Dengan menggunakan dukungan data semi-terstruktur di Amazon Redshift, Anda dapat menyerap dan menyimpan data semi-terstruktur di gudang data Amazon Redshift Anda. Menggunakan tipe data SUPER dan bahasa PartiQL, Amazon Redshift memperluas kemampuan data warehouse untuk berintegrasi dengan sumber data SQL dan NoSQL. Dengan cara ini, Amazon Redshift memungkinkan analitik yang efisien pada data tersimpan relasional dan semi-terstruktur seperti JSON. Untuk informasi tentang integrasi Amazon Redshift dengan PartiQL, lihat. PartiQL — bahasa SQL-compatible query untuk Amazon Redshift

Amazon Redshift menawarkan dua bentuk dukungan data semi-terstruktur: tipe data SUPER dan Amazon Redshift Spectrum.

Anda dapat menanyakan data semi-terstruktur dengan menelannya ke Amazon Redshift dan menyimpannya dalam tipe data SUPER atau menggunakan Amazon Redshift Spectrum untuk menanyakan data yang disimpan di Amazon S3.

Praktik terbaik untuk bekerja dengan data semi-terstruktur

Pertimbangkan praktik terbaik berikut saat bekerja dengan data semi-terstruktur.

- Kami menyarankan Anda mengatur opsi `enable_case_sensitive_super_attribute` konfigurasi ke true saat bekerja dengan data SUPER. Untuk informasi selengkapnya, lihat enable_case_sensitive_super_atribut.
- Gunakan perintah COPY untuk memuat data dari bucket Amazon S3 ke kolom Amazon Redshift SUPER.
- Gunakan pengetikan dinamis PartiQL dan semantik yang lemah untuk menjalankan kueri ad hoc pada nilai data SUPER tanpa harus memaksakan skema sebelum melakukan kueri. Untuk informasi tentang pengetikan dinamis, lihatPengetikan dinamis. Untuk informasi tentang semantik yang longgar, lihat. Semantik yang lemah
- Rusak data tanpa skema dan semi-terstruktur ke dalam tampilan terwujud menggunakan PartiQL jika Anda berencana untuk sering menanyakan data. Saat Anda melakukan analitik pada data yang diparut, organisasi kolom tampilan terwujud Amazon Redshift memberikan kinerja yang lebih baik. Selain itu, pengguna dan alat intelijen bisnis (BI) yang memerlukan skema konvensional untuk data yang dicerna dapat menggunakan tampilan (baik terwujud atau virtual) sebagai presentasi skema konvensional dari data.

Setelah tampilan terwujud PartiQL Anda mengekstrak data yang ditemukan di JSON atau SUPER ke dalam tampilan terwujud kolom konvensional, Anda dapat menanyakan tampilan yang terwujud. Untuk informasi tentang pandangan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift. Untuk informasi selengkapnya tentang cara kerja tipe data SUPER dengan tampilan yang terwujud, lihatJenis data SUPER dan tampilan yang terwujud.

Konsep untuk penggunaan tipe data SUPER

Berikut ini, Anda dapat menemukan beberapa konsep tipe data Amazon Redshift SUPER.

Memahami jenis data SUPER di Amazon Redshift — Tipe data SUPER adalah tipe data Amazon Redshift yang memungkinkan penyimpanan array dan struktur tanpa skema yang berisi skalar Amazon Redshift dan kemungkinan array dan struktur bersarang. Tipe data SUPER dapat secara native menyimpan berbagai format data semi-terstruktur, seperti JSON atau data yang berasal dari sumber berorientasi dokumen. Anda dapat menambahkan kolom SUPER baru untuk menyimpan data semi-terstruktur dan menulis kueri yang mengakses kolom SUPER, bersama dengan kolom skalar biasa. Untuk informasi selengkapnya tentang tipe data SUPER, lihatJenis SUPER.

Mengambil JSON tanpa skema ke dalam SUPER — Dengan tipe data SUPER semi-terstruktur yang fleksibel, Amazon Redshift dapat menerima dan menelan JSON tanpa skema ke dalam kolom dengan tipe data SUPER. Misalnya, Anda dapat memasukkan nilai JSON `[10.5, “first”]` ke dalam kolom tipe data SUPER dengan menggunakan perintah COPY. Kolom akan memiliki nilai SUPER`[10.5, ‘first’]`. Anda juga dapat menelan JSON menggunakanFungsi JSON_PARSE. Baik SALIN maupun men `json_parse` elan JSON menggunakan semantik penguraian yang ketat secara default. Anda juga dapat membangun nilai SUPER termasuk array dan struktur, menggunakan data database itu sendiri.

Kolom SUPER tidak memerlukan modifikasi skema saat menelan struktur fleksibel JSON tanpa skema. Misalnya, saat menganalisis aliran klik, Anda awalnya menyimpan di kolom SUPER struktur “klik” dengan atribut “IP” dan “waktu”. Anda dapat menambahkan atribut “id pelanggan” tanpa mengubah skema Anda untuk menerima perubahan tersebut.

Format asli yang digunakan untuk tipe data SUPER adalah format biner yang membutuhkan lebih sedikit ruang daripada nilai JSON dalam bentuk tekstualnya. Ini memungkinkan penyerapan dan pemrosesan runtime nilai SUPER yang lebih cepat pada kueri.

Kueri data SUPER dengan PartiQL — PartiQL adalah ekstensi yang kompatibel ke belakang dari SQL-92 yang digunakan banyak AWS layanan saat ini. Dengan penggunaan PartiQL, konstruksi

SQL yang sudah dikenal dengan mulus menggabungkan akses ke data SQL tabular klasik dan data semi-terstruktur SUPER. Anda dapat melakukan navigasi objek dan array dan array unnest. PartiQL memperluas bahasa SQL standar untuk secara deklaratif mengekspresikan dan memproses data bersarang dan multinilai.

PartiQL adalah perpanjangan dari SQL di mana data bersarang dan tanpa skema dari kolom SUPER adalah warga kelas satu. PartiQL tidak memerlukan semua ekspresi kueri untuk diperiksa tipe selama waktu kompilasi kueri. Pendekatan ini memungkinkan ekspresi kueri yang berisi tipe data SUPER untuk diketik secara dinamis selama eksekusi kueri ketika jenis data aktual di dalam kolom SUPER diakses. Juga, PartiQL beroperasi dalam mode lemah di mana inkonsistensi tipe tidak menyebabkan kegagalan tetapi mengembalikan null. Kombinasi pemrosesan kueri tanpa skema dan lemah membuat PartiQL ideal untuk aplikasi ekstrak, muat, transformasi (ELT) di mana kueri SQL Anda mengevaluasi data JSON yang dicerna di kolom SUPER.

Untuk informasi selengkapnya tentang PartiQL untuk Amazon Redshift, lihat. PartiQL — bahasa SQL-compatible query untuk Amazon Redshift Untuk informasi tentang pengetikan dinamis, lihatPengetikan dinamis. Untuk informasi tentang pemrosesan kueri yang lemah, lihatSemantik yang lemah.

Integrasikan dengan Redshift Spectrum — Amazon Redshift mendukung beberapa aspek PartiQL saat menjalankan kueri Redshift Spectrum melalui JSON, Parquet, dan format lain yang memiliki data bersarang. Redshift Spectrum hanya mendukung data bersarang yang memiliki skema. Misalnya, dengan Redshift Spectrum Anda dapat menyatakan bahwa data JSON Anda memiliki atribut `nested_schemaful_example` dalam skema`ARRAY<STRUCT<a:INTEGER, b:DECIMAL(5,2)>>`. Skema atribut ini menentukan bahwa data selalu berisi array, yang berisi struktur dengan integer `a` dan desimal `b`. Jika data berubah untuk menyertakan lebih banyak atribut, jenisnya juga berubah. Sebaliknya, tipe data SUPER tidak memerlukan skema. Anda dapat menyimpan array dengan elemen struktur yang memiliki atribut atau tipe berbeda. Juga, nilai dapat disimpan di luar array.

Pertimbangan untuk menggunakan data tipe SUPER

Saat bekerja dengan data SUPER, pertimbangkan hal berikut:

- Gunakan driver JDBC versi 2.x, driver ODBC versi 2.x, atau driver Amazon Redshift Python versi 2.0.872 atau yang lebih baru. Driver ODBC versi 1.x tidak didukung.

  Untuk informasi tentang driver JDBC, lihat Mengon figurasi koneksi untuk driver JDBC versi 2.x untuk Amazon Redshift di Panduan Manajemen Amazon Redshift.

Untuk informasi tentang driver ODBC, lihat Mengon figurasi koneksi untuk driver ODBC versi 2.x untuk Amazon Redshift di Panduan Manajemen Amazon Redshift..

Untuk informasi tentang driver Python, lihat konektor Amazon Redshift Python di Panduan Manajemen Amazon Redshift..

Untuk informasi selengkapnya tentang konfigurasi SUPER, lihatKonfigurasi SUPER.

PartiQL — bahasa SQL-compatible query untuk Amazon Redshift

Amazon Redshift mendukung PartiQL, bahasa SQL-compatible kueri, untuk memilih, menyisipkan, memperbarui, dan menghapus data di Amazon Redshift. Menggunakan PartiQL, Anda dapat dengan mudah berinteraksi dengan tabel Amazon Redshift dan menjalankan kueri ad hoc menggunakan Konsol Manajemen AWS, SQL, the Workbench/J AWS Command Line Interface, dan Amazon Redshift Data API untuk PartiQL.

Operasi PartiQL memberikan ketersediaan, latensi, dan kinerja yang sama seperti operasi bidang data Redshift lainnya.

Bagian berikut menjelaskan implementasi Amazon Redshift dari PartiQL.

Apa yang dimaksud dengan PartiQL?

PartiQL menyediakan akses SQL-compatible kueri di beberapa penyimpanan data yang berisi data terstruktur, semi-terstruktur, dan bersarang. Ini banyak digunakan di Amazon dan sekarang tersedia sebagai bagian dari banyak AWS layanan, termasuk Amazon Redshift.

Untuk spesifikasi PartiQL dan tutorial tentang bahasa query inti, lihat ikhtisar PartiQL.

Untuk contoh penggunaan kueri PartiQL di Amazon Redshift dengan data semi-terstruktur, lihat. Contoh penggunaan data semi-terstruktur di Amazon Redshift

> **Note**
>
> - Amazon Redshift mendukung subset bahasa query PartiQL.
> - Amazon Redshift tidak mendukung format data Amazon Ion atau literal Amazon Ion.

PartiQL di Amazon Redshift

Untuk menjalankan kueri PartiQL di Amazon Redshift, Anda dapat menggunakan metode berikut:

- Yang Konsol Manajemen AWS
- SQL Workbench/J
- Yang AWS CLI
- API Data Amazon Redshift untuk PartiQL

Memuat data semi-terstruktur ke Amazon Redshift

Gunakan tipe data SUPER untuk mengurai dan menanyakan data hierarkis dan generik di Amazon Redshift. Amazon Redshift memperkenalkan Fungsi JSON_PARSE untuk mengurai data dalam format JSON dan mengubahnya menjadi representasi SUPER. Amazon Redshift juga mendukung pemuatan kolom SUPER menggunakan perintah COPY. Format file yang didukung adalah JSON, Avro, teks, format nilai yang dipisahkan koma (CSV), Parquet, dan ORC.

Anda dapat menggunakan JSON_PARSE untuk menyisipkan dan memperbarui data JSON ke kolom SUPER, atau menggunakan COPY untuk memuat data JSON ke Amazon Redshift dari sumber luar seperti dari bucket Amazon S3.

Topik

- Menggunakan JSON_PARSE untuk memasukkan data ke kolom SUPER
- Menggunakan COPY untuk memuat data ke kolom SUPER

Menggunakan JSON_PARSE untuk memasukkan data ke kolom SUPER

Anda dapat memasukkan atau memperbarui data JSON ke dalam kolom SUPER menggunakanFungsi JSON_PARSE. Fungsi mengurai data dalam format JSON dan mengubahnya menjadi tipe data SUPER, yang dapat Anda gunakan dalam pernyataan INSERT atau UPDATE.

Contoh berikut menyisipkan data JSON ke dalam kolom SUPER. Jika fungsi JSON_PARSE hilang dalam kueri, Amazon Redshift memperlakukan nilai sebagai string tunggal, bukan string yang harus diurai JSON-formatted .

```
--Drop the table if it exists.
```

```
DROP TABLE IF EXISTS test_json;

--Create the table.
CREATE TABLE test_json (all_data SUPER);

--Populate the table.
INSERT INTO test_json VALUES (JSON_PARSE('
{
    "name": {
        "first_name": "Jake",
        "last_name": "Smith"
    },
    "age": 30,
    "hobby": "Biking"
}'
) );

SELECT * FROM test_json;

 all_data
---------
{"name":{"first_name":"Jake","last_name":"Smith"},"age":30,"hobby":"Biking"}
```

Menggunakan COPY untuk memuat data ke kolom SUPER

Di bagian berikut, Anda dapat mempelajari berbagai cara menggunakan perintah COPY untuk memuat data JSON ke Amazon Redshift. Untuk informasi tentang parameter format data yang digunakan Amazon Redshift untuk mengurai JSON dalam perintah COPY, baca deskripsi JSON format for COPY parameter di. Parameter format data

Topik

- Menyalin data dari JSON dan Avro
- Menyalin data dari teks dan CSV
- Menyalin data dari parket dan ORC format kolom

Menyalin data dari JSON dan Avro

Amazon Redshift menyediakan metode berikut untuk menelan dokumen JSON menggunakan COPY, bahkan dengan struktur JSON yang sepenuhnya atau sebagian tidak diketahui:

- Simpan data yang berasal dari dokumen JSON ke dalam kolom data SUPER tunggal menggunakan `noshred` opsi. Metode ini berguna ketika skema tidak diketahui atau diharapkan berubah. Dengan demikian, metode ini memudahkan untuk menyimpan seluruh tupel dalam satu kolom SUPER.
- Rusak dokumen JSON menjadi beberapa kolom Amazon Redshift menggunakan opsi `auto` atau`jsonpaths`. Atribut dapat berupa skalar Amazon Redshift atau nilai SUPER.

Anda dapat menggunakan opsi ini dengan format JSON atau Avro. Untuk informasi selengkapnya tentang opsi JSON seperti`noshred`,`auto`, dan`jsonpaths`, lihatJSON format for COPY.

Ukuran maksimum untuk objek JSON di Amazon Redshift adalah 4 MB, yang berlaku sebelum penghancuran atau penguraian apa pun.

Metode 1: Menyalin dokumen JSON ke dalam satu kolom data SUPER menggunakan **`noshred`**

Anda dapat menyalin seluruh dokumen JSON ke dalam kolom data SUPER tunggal menggunakan `noshred` opsi dalam perintah COPY. Pertimbangkan contoh berikut:

1. Buat tabel dengan kolom data SUPER tunggal.

   ```
   CREATE TABLE region_nations_noshred (rdata SUPER);
   ```

2. Salin data dari Amazon S3 ke kolom data SUPER tunggal. Untuk menelan data sumber JSON ke dalam kolom data SUPER tunggal, tentukan `noshred` opsi dalam klausa FORMAT JSON.

   ```
   COPY region_nations_noshred FROM 's3://redshift-downloads/semistructured/tpch-
   nested/data/json/region_nation'
   REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxx:role/Redshift-S3'
   FORMAT JSON 'noshred';
   ```

   Setelah COPY berhasil menelan JSON, tabel Anda memiliki kolom data `rdata` SUPER yang berisi data dari seluruh objek JSON. Data yang dicerna mempertahankan semua properti hierarki JSON. Namun, daun dikonversi ke tipe skalar Amazon Redshift untuk pemrosesan kueri yang efisien.

3. Gunakan query berikut untuk mengambil string JSON asli.

   ```
   SELECT rdata FROM region_nations_noshred;
   ```

Saat Amazon Redshift menghasilkan kolom data SUPER, kolom tersebut dapat diakses menggunakan JDBC sebagai string melalui serialisasi JSON. Untuk informasi selengkapnya, lihat Serialisasi JSON bersarang kompleks.

Metode 2: Menyalin dokumen JSON ke beberapa kolom data SUPER

Anda dapat merobek dokumen JSON menjadi beberapa kolom yang dapat berupa kolom data SUPER atau tipe skalar Amazon Redshift. Amazon Redshift menyebarkan bagian yang berbeda dari objek JSON ke kolom yang berbeda. Pertimbangkan contoh berikut:

1. Buat tabel dengan beberapa kolom untuk menampung JSON yang diparut.

```
CREATE TABLE region_nations
(
 r_regionkey smallint
 ,r_name varchar
 ,r_comment varchar
 ,r_nations super
 );
```

2. Untuk menyalin JSON ke dalam `region_nations` tabel, tentukan opsi AUTO dalam klausa FORMAT JSON untuk membagi nilai JSON di beberapa kolom. COPY cocok dengan atribut JSON tingkat atas dengan nama kolom dan memungkinkan nilai bersarang untuk dicerna sebagai nilai SUPER, seperti array dan objek JSON.

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/
json/region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'auto';
```

   Ketika nama atribut JSON dalam campuran huruf besar dan kecil, tentukan `auto ignorecase` opsi dalam klausa FORMAT JSON. Untuk informasi selengkapnya tentang perintah COPY, lihatMuat dari data JSON menggunakan opsi 'auto ignorecase'.

Dalam beberapa kasus, ada ketidakcocokan antara nama kolom dan atribut JSON atau atribut yang akan dimuat bersarang lebih dari kedalaman level. Jika demikian, gunakan `jsonpaths` file untuk memetakan atribut JSON secara manual ke kolom Amazon Redshift. Pertimbangkan contoh berikut:

1. Buat tabel dengan beberapa kolom untuk menampung JSON yang diparut.

```
CREATE TABLE region_nations
(
 r_regionkey smallint
 ,r_name varchar
 ,r_comment varchar
 ,r_nations super
 );
```

2. Dalam hal ini, nama kolom tidak cocok dengan atribut JSON. Anda dapat membuat `jsonpaths` file yang memetakan jalur atribut ke kolom tabel berdasarkan posisinya dalam `jsonpaths` array, seperti berikut ini:

```
{"jsonpaths": [
       "$.r_regionkey",
       "$.r_name",
       "$.r_comment",
       "$.r_nations
    ]
}
```

3. Gunakan lokasi `jsonpaths` file sebagai argumen ke opsi FORMAT JSON di COPY.

```
COPY nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 's3://redshift-downloads/semistructured/tpch-nested/data/jsonpaths/
nations_jsonpaths.json';
```

4. Gunakan kueri berikut untuk mengakses tabel yang menunjukkan penyebaran data ke beberapa kolom. Kolom data SUPER dicetak menggunakan format JSON.

```
SELECT r_regionkey,r_name,r_comment,r_nations[0].n_nationkey FROM region_nations
 ORDER BY 1,2,3 LIMIT 1;
```

`jsonpaths`file memetakan bidang dalam dokumen JSON ke kolom tabel. Anda dapat mengekstrak kolom tambahan, seperti kunci distribusi dan pengurutan, sambil tetap memuat dokumen lengkap sebagai kolom SUPER. Kueri berikut memuat dokumen lengkap ke `nations` kolom. `name`Kolom

adalah kunci pengurutan dan `regionkey` kolom adalah kunci distribusi. Pertimbangkan contoh berikut:

1. Buat tabel dengan beberapa kolom untuk menampung JSON yang diparut.

```
CREATE TABLE nations_sorted (
    regionkey smallint,
    name varchar,
    nations super
) DISTKEY(regionkey) SORTKEY(name);
```

2. Petakan root jsonpath “$” ke root dokumen sebagai berikut:

```
{"jsonpaths": [
       "$.r_regionkey",
       "$.r_name",
       "$"
    ]
}
```

3. Gunakan lokasi `jsonpaths` file sebagai argumen ke opsi FORMAT JSON di COPY.

```
COPY nations_sorted FROM 's3://redshift-downloads/semistructured/tpch-nested/data/
json/region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 's3://redshift-downloads/semistructured/tpch-nested/data/jsonpaths/
nations_sorted_jsonpaths.json';
```

Untuk informasi lebih lanjut tentang penggunaan`jsonpaths`, lihatSALIN dari format JSON.

Menyalin data dari teks dan CSV

Amazon Redshift mewakili kolom SUPER dalam format teks dan CSV sebagai JSON serial. Pemformatan JSON yang valid diperlukan agar kolom SUPER dapat memuat dengan informasi jenis yang benar. Hapus kutipan objek, array, angka, boolean, dan nilai nol. Bungkus nilai string dalam tanda kutip ganda. Kolom SUPER menggunakan aturan pelarian standar untuk format teks dan CSV.

Saat menyalin dari CSV, pembatas lolos sesuai dengan standar CSV. Pertimbangkan contoh berikut:

```
CREATE TABLE region_nations
(
```

```
 r_regionkey smallint
 ,r_name varchar
 ,r_comment varchar
 ,r_nations super
 );

 COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/csv/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT CSV;
```

Saat menyalin dari teks, jika pembatas yang dipilih mungkin juga muncul di bidang SUPER, gunakan opsi ESCAPE selama COPY dan UNLOAD. Pertimbangkan contoh berikut:

```
CREATE TABLE region_nations
(
 r_regionkey smallint
 ,r_name varchar
 ,r_comment varchar
 ,r_nations super
 );

COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/text/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
DELIMITER ','
ESCAPE;
```

Menyalin data dari parket dan ORC format kolom

Jika data semi-terstruktur atau bersarang sudah tersedia dalam format Apache Parquet atau Apache ORC, Anda dapat menggunakan perintah COPY untuk menyerap data ke Amazon Redshift.

Struktur tabel Amazon Redshift harus sesuai dengan jumlah kolom dan tipe data kolom dari file Parquet atau ORC. Dengan menentukan SERIALIZETOJSON dalam perintah COPY, Anda dapat memuat jenis kolom apa pun dalam file yang sejajar dengan kolom SUPER dalam tabel sebagai SUPER. Ini termasuk struktur dan tipe array.

Contoh berikut menggunakan format parket:

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/
parquet/region_nation'
```

```
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT PARQUET SERIALIZETOJSON;
```

Contoh berikut menggunakan format ORC.

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/orc/
region_nation'
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT ORC SERIALIZETOJSON;
```

Ketika atribut tipe data tanggal atau waktu berada di ORC, Amazon Redshift mengubahnya menjadi varchar setelah mengkodekannya di SUPER.

Membongkar data semi-terstruktur

Dengan Amazon Redshift, Anda dapat mengekspor data semi-terstruktur dari cluster Amazon Redshift ke Amazon S3 dalam berbagai format, termasuk teks, Apache Parquet, Apache ORC, dan Avro. Bagian berikut akan memandu Anda melalui proses mengonfigurasi dan menjalankan operasi pembongkaran untuk data semi-terstruktur Anda di Amazon Redshift.

CSV or text formats

Anda dapat membongkar tabel dengan kolom data SUPER ke Amazon S3 dalam format nilai yang dipisahkan koma (CSV) atau teks. Menggunakan kombinasi klausa navigasi dan unnest, Amazon Redshift menurunkan data hierarkis dalam format data SUPER ke Amazon S3 dalam format CSV atau teks. Selanjutnya, Anda dapat membuat tabel eksternal terhadap data yang tidak dimuat dan menanyainya menggunakan Redshift Spectrum. Untuk informasi tentang penggunaan UNLOAD dan izin IAM yang diperlukan, lihat. MEMBONGKAR

Contoh berikut membongkar semua data dari tabel Amazon Redshift ke bucket Amazon S3.

```
UNLOAD ('SELECT * FROM <redshift_table>')
TO '<S3_bucket>'
IAM_ROLE '<iam_role>'
DELIMITER AS '|'
GZIP
ALLOWOVERWRITE;
```

Tidak seperti tipe data lain di mana string yang ditentukan pengguna mewakili nilai nol, Amazon Redshift mengekspor kolom data SUPER menggunakan format JSON dan merepresentasikannya

sebagai null seperti yang ditentukan oleh format JSON. Akibatnya, kolom data SUPER mengabaikan opsi NULL [AS] yang digunakan dalam perintah UNLOAD.

Parquet format

Anda dapat membongkar tabel dengan kolom data SUPER ke Amazon S3 dalam format Parquet. Amazon Redshift mewakili kolom SUPER di Parquet sebagai tipe data JSON. Hal ini memungkinkan data semi-terstruktur untuk direpresentasikan dalam Parquet. Anda dapat menanyakan kolom ini menggunakan Redshift Spectrum atau menelannya kembali ke Amazon Redshift menggunakan perintah COPY. Untuk informasi tentang penggunaan UNLOAD dan izin IAM yang diperlukan, lihat. MEMBONGKAR

Contoh berikut membongkar semua data dari tabel Amazon Redshift ke bucket Amazon S3 dalam format Parquet.

```
UNLOAD ('SELECT * FROM <Amazon Redshift_table>')
TO '<S3_bucket>'
IAM_ROLE '<iam_role>'
FORMAT PARQUET;
```

Mengkueri data semi-terstruktur

Dengan Amazon Redshift, Anda dapat menanyakan dan menganalisis data semi-terstruktur, seperti JSON, Avro, atau Ion, di samping data terstruktur Anda. Semi-structured Data mengacu pada data yang memiliki skema fleksibel, memungkinkan struktur hierarkis atau bersarang. Bagian berikut menunjukkan kueri data semi-terstruktur menggunakan dukungan Amazon Redshift untuk format data terbuka, sehingga Anda dapat membuka informasi berharga dari struktur data yang kompleks.

Amazon Redshift menggunakan bahasa PartiQL untuk menawarkan SQL-compatible akses ke data relasional, semi-terstruktur, dan bersarang.

PartiQL beroperasi dengan tipe dinamis. Pendekatan ini memungkinkan pemfilteran intuitif, penggabungan, dan agregasi pada kombinasi kumpulan data terstruktur, semi-terstruktur, dan bersarang. Sintaks PartiQL menggunakan notasi bertitik dan subskrip array untuk navigasi jalur saat mengakses data bersarang. Ini juga memungkinkan item klausa FROM untuk mengulangi array dan digunakan untuk operasi unnest. Berikut ini, Anda dapat menemukan deskripsi pola kueri yang berbeda yang menggabungkan penggunaan tipe data SUPER dengan navigasi jalur dan array, un, unpivoting, dan join. Untuk informasi lebih lanjut tentang PartiQL, lihatPartiQL — bahasa SQL-compatible query untuk Amazon Redshift.

Navigasi

Amazon Redshift menggunakan PartiQL untuk mengaktifkan navigasi ke array dan struktur menggunakan tanda kurung [...] dan notasi titik masing-masing. Selanjutnya, Anda dapat mencampur navigasi ke dalam struktur menggunakan notasi titik dan array menggunakan notasi braket. Misalnya, pernyataan berikut hanya memilih elemen ketiga dalam array yang bersarang satu tingkat dalam objek SUPER:

```
SELECT super_object.array[2];

 array
---------------
 third_element
```

Anda dapat menggunakan notasi titik dan tanda kurung saat melakukan operasi data seperti pemfilteran, penggabungan, dan agregasi. Anda dapat menggunakan notasi ini di mana saja dalam kueri di mana biasanya ada referensi kolom. Misalnya, pernyataan berikut memilih jumlah peristiwa dengan tipe`UPDATED`.

```
SELECT COUNT(*)
FROM test_json
WHERE all_data.data.pnr.events[0]."eventType" = 'UPDATED';

 eventType | count
-----------+-------
 "UPDATED" | 1
```

Untuk contoh yang lebih mendalam tentang menggunakan navigasi PartiQL, lihatContoh penggunaan data semi-terstruktur di Amazon Redshift.

Kueri tanpa sarang

Untuk mengatasi kueri, Amazon Redshift menyediakan dua cara untuk mengulangi array SUPER: sintaks PartiQL dan operasi UNNEST dalam klausa FROM. Kedua metode unmenghasilkan output yang sama. Untuk informasi tentang operasi UNNEST, lihatKlausa FROM. Untuk contoh penggunaan operasi UNNEST, lihatContoh UNNEST.

Amazon Redshift dapat menavigasi array SUPER menggunakan sintaks PartiQL dalam klausa FROM kueri. Menggunakan contoh sebelumnya, contoh berikut mengulangi nilai atribut untuk`c_orders`.

```
SELECT orders.*, o FROM customer_orders orders, orders.c_orders o;
```

Sintaks PartiQL dari unmenggunakan item klausa FROM x (AS) y berarti bahwa y mengulangi setiap nilai (SUPER) dalam ekspresi array (SUPER) x. Dalam hal ini, x adalah ekspresi SUPER dan y merupakan alias untukx.

Operan kiri juga dapat menggunakan notasi titik dan tanda kurung untuk navigasi reguler. Dalam contoh berikut, customer_orders_lineitem c adalah iterasi di atas tabel customer_order_lineitem dasar dan c.c_orders o merupakan iterasi di atas c.c_orders array. Untuk mengulangi o_lineitems atribut, yang merupakan array dalam array, Anda dapat menambahkan beberapa klausa, seperti:

```
SELECT c.*, o, l FROM customer_orders_lineitem c, c.c_orders o, o.o_lineitems l;
```

Amazon Redshift juga mendukung indeks array saat mengulangi array menggunakan kata kunci AT. Klausa x AS y AT z mengulangi array x dan menghasilkan bidang z, yang merupakan indeks array. Contoh berikut menunjukkan bagaimana indeks array bekerja.

```
SELECT c_name,
       orders.o_orderkey AS orderkey,
       index AS orderkey_index
FROM customer_orders_lineitem c, c.c_orders AS orders AT index
ORDER BY orderkey_index;

c_name             | orderkey | orderkey_index
-------------------+----------+----------------
Customer#000008251 | 3020007  |        0
Customer#000009452 | 4043971  |        0
  (2 rows)
```

Berikut ini adalah contoh iterasi di atas array skalar.

```
CREATE TABLE bar AS SELECT json_parse('{"scalar_array": [1, 2.3, 45000000]}') AS data;

SELECT element, index FROM bar AS b, b.data.scalar_array AS element AT index;

 index | element
-------+----------
     0 | 1
     1 | 2.3
```

```
     2 | 45000000
(3 rows)
```

Contoh berikut berulang-ulang di atas array beberapa level. Contoh menggunakan beberapa klausa unnest untuk mengiterasi ke array terdalam. Array `f.multi_level_array` AS berulang`multi_level_array`. Elemen array AS adalah iterasi atas array di dalamnya. `multi_level_array`

```
CREATE TABLE foo AS SELECT json_parse('[[1.1, 1.2], [2.1, 2.2], [3.1, 3.2]]') AS
 multi_level_array;

SELECT array, element FROM foo AS f, f.multi_level_array AS array, array AS element;

 element | array
---------+---------
 1.1     | [1.1,1.2]
 1.2     | [1.1,1.2]
 2.1     | [2.1,2.2]
 2.2     | [2.1,2.2]
 3.1     | [3.1,3.2]
 3.2     | [3.1,3.2]
(6 rows)
```

Untuk informasi lebih lanjut tentang klausa FROM, lihatKlausa FROM. Untuk contoh lebih lanjut tentang kueri SUPER yang tidak bersarang, lihatContoh penggunaan data semi-terstruktur di Amazon Redshift.

Objek tidak berputar

Untuk melakukan pembongkaran objek, Amazon Redshift menggunakan sintaks PartiQL untuk mengulangi objek SUPER. Ia melakukan ini menggunakan klausa FROM dari kueri bersama dengan kata kunci UNPIVOT. Dalam contoh berikut, ekspresi adalah `c.c_orders[0]` objek. Contoh kueri berulang di atas setiap atribut yang dikembalikan oleh objek.

```
SELECT attr as attribute_name, json_typeof(val) as value_type
FROM customer_orders_lineitem c, UNPIVOT c.c_orders[0] AS val AT attr
WHERE c_custkey = 9451;

 attribute_name  | value_type
-----------------+------------
 o_orderstatus   | string
```

```
 o_clerk         | string
 o_lineitems     | array
 o_orderdate     | string
 o_shippriority  | number
 o_totalprice    | number
 o_orderkey      | number
 o_comment       | string
 o_orderpriority | string
(9 rows)
```

Seperti halnya un, sintaks unpivoting juga merupakan perpanjangan dari klausa FROM. Perbedaannya adalah bahwa sintaks unpivoting menggunakan kata kunci UNPIVOT untuk menunjukkan bahwa itu mengulangi objek alih-alih array. Ini menggunakan AS `value_alias` untuk iterasi pada semua nilai di dalam objek, dan menggunakan AT `attribute_alias` untuk mengiterasi semua atribut. Pertimbangkan sintaks berikut:

```
UNPIVOT expression AS value_alias [ AT attribute_alias ]
```

Amazon Redshift mendukung penggunaan objek unpivoting dan unarray dalam satu klausa FROM sebagai berikut:

```
SELECT attr as attribute_name, val as object_value
FROM customer_orders_lineitem c, c.c_orders AS o, UNPIVOT o AS val AT attr
WHERE c_custkey = 9451;
```

Saat Anda menggunakan objek unpivoting, Amazon Redshift tidak mendukung unpioting berkorelasi. Secara khusus, anggaplah Anda memiliki kasus di mana ada beberapa contoh unpivoting di tingkat kueri yang berbeda dan bagian dalam yang tidak berputar mereferensikan yang luar. Amazon Redshift tidak mendukung jenis multiple unpivoting ini.

Untuk informasi lebih lanjut tentang klausa FROM, lihatKlausa FROM. Untuk contoh penggunaan berputar dengan tipe SUPER, lihatContoh penggunaan data semi-terstruktur di Amazon Redshift.

Pengetikan dinamis

Pengetikan dinamis tidak memerlukan pengecoran eksplisit data yang diekstraksi dari jalur titik dan tanda kurung. Amazon Redshift menggunakan pengetikan dinamis untuk memproses data SUPER tanpa skema tanpa perlu mendeklarasikan tipe data sebelum Anda menggunakannya dalam kueri Anda. Pengetikan dinamis menggunakan hasil navigasi ke kolom data SUPER tanpa harus secara eksplisit menempatkannya ke jenis Amazon Redshift. Pengetikan dinamis paling berguna

dalam klausa join dan GROUP BY. Contoh berikut menggunakan pernyataan SELECT yang tidak memerlukan pengecoran eksplisit ekspresi titik dan tanda kurung ke jenis Amazon Redshift biasa. Untuk informasi tentang kompatibilitas jenis dan konversi, lihatKompatibilitas dan konversi tipe.

Pertimbangkan contoh berikut, yang mencari baris di mana status pesanan adalah`shipped`:

```
SELECT c_orders[0].o_orderkey
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderstatus = 'shipped';
```

Tanda kesetaraan dalam kueri sampel ini mengevaluasi `true` kapan nilai c_orders [0] .o_orderstatus adalah string 'shipped'. Dalam semua kasus lain, tanda kesetaraan mengevaluasi`false`, termasuk kasus-kasus di mana argumen kesetaraan adalah jenis yang berbeda. Misalnya, jika status pesanan adalah bilangan bulat, barisnya tidak akan dipilih.

Pengetikan dinamis dan statis

Tanpa menggunakan pengetikan dinamis, Anda tidak dapat menentukan apakah c_orders [0] .o_orderstatus adalah string, integer, atau struktur. Anda hanya dapat menentukan bahwa c_orders [0] .o_orderstatus adalah tipe data SUPER, yang dapat berupa skalar Amazon Redshift, array, atau struktur. Tipe statis c_orders [0] .o_orderstatus adalah tipe data SUPER. Secara konvensional, tipe secara implisit adalah tipe statis dalam SQL.

Amazon Redshift menggunakan pengetikan dinamis untuk pemrosesan data tanpa skema. Ketika kueri mengevaluasi data, c_orders [0] .o_orderstatus ternyata tipe tertentu. Misalnya, mengevaluasi c_orders [0] .o_orderstatus pada catatan pertama customer_orders_lineitem dapat menghasilkan integer. Mengevaluasi pada catatan kedua dapat menghasilkan string. Ini adalah jenis ekspresi dinamis.

Saat menggunakan operator SQL atau fungsi dengan ekspresi titik dan tanda kurung yang memiliki tipe dinamis, Amazon Redshift menghasilkan hasil yang mirip dengan menggunakan operator atau fungsi SQL standar dengan tipe statis masing-masing. Dalam contoh ini, ketika tipe dinamis dari ekspresi jalur adalah string, perbandingan dengan string 'P' bermakna. Setiap kali tipe dinamis c_orders [0] .o_orderstatus adalah tipe data lain kecuali string, kesetaraan mengembalikan false. Fungsi lain mengembalikan null ketika argumen salah ketik digunakan.

Contoh berikut menulis kueri sebelumnya dengan pengetikan statis:

```
SELECT c_custkey
FROM customer_orders_lineitem
```

```
WHERE CASE WHEN JSON_TYPEOF(c_orders[0].o_orderstatus) = 'string'
           THEN c_orders[0].o_orderstatus::VARCHAR = 'P'
           ELSE FALSE END;
```

Perhatikan perbedaan berikut antara predikat kesetaraan dan predikat perbandingan. Dalam contoh sebelumnya, jika Anda mengganti predikat kesetaraan dengan predikat kurang dari atau sama, semantik menghasilkan nol, bukan false.

```
SELECT c_orders[0]. o_orderkey
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderstatus <= 'P';
```

Dalam contoh ini, jika c_orders [0] .o_orderstatus adalah string, Amazon Redshift mengembalikan true jika menurut abjad sama atau lebih kecil dari 'P'. Amazon Redshift mengembalikan false jika menurut abjad lebih besar dari 'P'. Namun, jika c_orders [0] .o_orderstatus bukan string, Amazon Redshift mengembalikan null karena Amazon Redshift tidak dapat membandingkan nilai dari jenis yang berbeda, seperti yang ditunjukkan dalam kueri berikut:

```
SELECT c_custkey
FROM customer_orders_lineitem
WHERE CASE WHEN JSON_TYPEOF(c_orders[0].o_orderstatus) = 'string'
           THEN c_orders[0].o_orderstatus::VARCHAR <= 'P'
           ELSE NULL END;
```

Pengetikan dinamis tidak mengecualikan dari perbandingan tipe yang minimal sebanding. Misalnya, Anda dapat mengonversi tipe skalar CHAR dan VARCHAR Amazon Redshift menjadi SUPER. Mereka sebanding dengan string, termasuk mengabaikan karakter spasi putih di belakang yang mirip dengan jenis Amazon Redshift CHAR dan VARCHAR. Demikian pula, bilangan bulat, desimal, dan nilai floating-point sebanding dengan nilai SUPER. Khusus untuk kolom desimal, setiap nilai juga dapat memiliki skala yang berbeda. Amazon Redshift masih menganggapnya sebagai tipe dinamis.

Amazon Redshift juga mendukung kesetaraan pada objek dan array yang dievaluasi sebagai sama dalam, seperti mengevaluasi jauh ke dalam objek atau array dan membandingkan semua atribut. Gunakan deep equal dengan hati-hati, karena proses melakukan deep equence bisa memakan waktu.

Menggunakan pengetikan dinamis untuk bergabung

Untuk gabungan, pengetikan dinamis secara otomatis mencocokkan nilai dengan tipe dinamis yang berbeda tanpa melakukan analisis CASE WHEN yang panjang untuk mengetahui jenis data apa yang

mungkin muncul. Misalnya, asumsikan bahwa organisasi Anda mengubah format yang digunakan untuk kunci bagian dari waktu ke waktu.

Kunci bagian integer awal yang dikeluarkan diganti dengan kunci bagian string, seperti 'A55', dan kemudian diganti lagi dengan kunci bagian array, seperti ['X', 10] yang menggabungkan string dan angka. Amazon Redshift tidak harus melakukan analisis kasus yang panjang tentang kunci bagian dan dapat menggunakan join, seperti yang ditunjukkan pada contoh berikut.

```
SELECT c.c_name
    ,l.l_extendedprice
    ,l.l_discount
FROM customer_orders_lineitem c
    ,c.c_orders o
    ,o.o_lineitems l
    ,supplier_partsupp s
    ,s.s_partsupps ps
WHERE l.l_partkey = ps.ps_partkey
AND c.c_nationkey = s.s_nationkey
ORDER BY c.c_name;
```

Contoh berikut menunjukkan betapa kompleks dan tidak efisiennya kueri yang sama tanpa menggunakan pengetikan dinamis:

```
SELECT c.c_name
    ,l.l_extendedprice
    ,l.l_discount
FROM customer_orders_lineitem c
    ,c.c_orders o
    ,o.o_lineitems l
    ,supplier_partsupp s
    ,s.s_partsupps ps
WHERE CASE WHEN IS_INTEGER(l.l_partkey) AND IS_INTEGER(ps.ps_partkey)
           THEN l.l_partkey::integer = ps.ps_partkey::integer
           WHEN IS_VARCHAR(l.l_partkey) AND IS_VARCHAR(ps.ps_partkey)
           THEN l.l_partkey::varchar = ps.ps_partkey::varchar
           WHEN IS_ARRAY(l.l_partkey) AND IS_ARRAY(ps.ps_partkey)
                AND IS_VARCHAR(l.l_partkey[0]) AND IS_VARCHAR(ps.ps_partkey[0])
                AND IS_INTEGER(l.l_partkey[1]) AND IS_INTEGER(ps.ps_partkey[1])
           THEN l.l_partkey[0]::varchar = ps.ps_partkey[0]::varchar
                AND l.l_partkey[1]::integer = ps.ps_partkey[1]::integer
           ELSE FALSE END
AND c.c_nationkey = s.s_nationkey
```

```
ORDER BY c.c_name;
```

Case-insensitive menanyakan

Anda dapat melakukan perbandingan string yang tidak peka huruf besar/kecil pada data SUPER menggunakan fungsi COLLATE atau dengan mengatur kolasi pada tingkat kolom atau database. Untuk informasi selengkapnya tentang pengaturan kolasi pada pembuatan tabel, lihatCREATE TABLE. Untuk informasi tentang perilaku kolasi dengan operator dan fungsi data SUPER, lihatPerilaku pengumpulan.

Contoh berikut menggunakan fungsi COLLATE pada nilai string yang diekstraksi dari data SUPER.

```
CREATE TABLE events (data SUPER);
INSERT INTO events VALUES (JSON_PARSE('{"status": "Active", "name": "Event1"}'));
INSERT INTO events VALUES (JSON_PARSE('{"status": "ACTIVE", "name": "Event2"}'));
INSERT INTO events VALUES (JSON_PARSE('{"status": "active", "name": "Event3"}'));

SELECT data.name FROM events
WHERE COLLATE(data.status::VARCHAR, 'case_insensitive') = 'active';

 name
----------
 "Event1"
 "Event2"
 "Event3"
(3 rows)
```

Anda juga dapat menentukan kolom SUPER dengan kolasi case-insensitive pada pembuatan tabel. Dalam hal ini, semua perbandingan string pada kolom tidak peka huruf besar/kecil.

```
CREATE TABLE events_ci (data SUPER COLLATE CASE_INSENSITIVE);
INSERT INTO events_ci VALUES (JSON_PARSE('{"status": "Active"}'));
INSERT INTO events_ci VALUES (JSON_PARSE('{"status": "ACTIVE"}'));

SELECT * FROM events_ci WHERE data.status::VARCHAR = 'active';

 data
-----------------------
 {"status":"Active"}
 {"status":"ACTIVE"}
(2 rows)
```

Semantik yang lemah

Secara default, operasi navigasi pada nilai SUPER mengembalikan null alih-alih mengembalikan kesalahan saat navigasi tidak valid. Navigasi objek tidak valid jika nilai SUPER bukan objek atau jika nilai SUPER adalah objek tetapi tidak berisi nama atribut yang digunakan dalam kueri. Misalnya, kueri berikut mengakses nama atribut yang tidak valid di kolom data SUPER cdata:

```
SELECT c.c_orders.something FROM customer_orders_lineitem c;
```

Navigasi array mengembalikan null jika nilai SUPER bukan array atau indeks array berada di luar batas. Kueri berikut mengembalikan null karena c_orders [1] [1] berada di luar batas.

```
SELECT c.c_orders[1][1] FROM customer_orders_lineitem c;
```

Semantik lax sangat berguna saat menggunakan pengetikan dinamis untuk memberikan nilai SUPER. Mendapatkan nilai SUPER ke tipe yang salah mengembalikan null alih-alih kesalahan jika cast tidak valid. Misalnya, kueri berikut mengembalikan null karena tidak dapat memberikan nilai string 'Baik' dari atribut objek o_orderstatus ke INTEGER. Amazon Redshift mengembalikan kesalahan untuk pemeran VARCHAR ke INTEGER tetapi tidak untuk pemeran SUPER.

```
SELECT c.c_orders.o_orderstatus::integer FROM customer_orders_lineitem c;
```

Urutkan berdasarkan

Amazon Redshift tidak mendefinisikan perbandingan SUPER di antara nilai-nilai dengan tipe dinamis yang berbeda. Nilai SUPER yang merupakan string tidak lebih kecil atau lebih besar dari nilai SUPER yang merupakan angka. Untuk menggunakan klausa ORDER BY dengan kolom SUPER, Amazon Redshift mendefinisikan urutan total di antara berbagai jenis yang akan diamati saat Amazon Redshift memberi peringkat nilai SUPER menggunakan klausa ORDER BY. Urutan di antara tipe dinamis adalah boolean, angka, string, array, objek.

Untuk contoh penggunaan GROUP BY dan ORDER BY dalam kueri SUPER, lihatMemfilter data semi-terstruktur.

Operator dan fungsi

Dengan Amazon Redshift, Anda dapat melakukan analitik lanjutan pada kumpulan data besar menggunakan data SUPER menggunakan operator dan fungsi. Operator dan fungsi untuk data

SUPER adalah konstruksi SQL yang memungkinkan analisis kompleks dan manipulasi data semi-terstruktur yang disimpan dalam tabel Amazon Redshift.

Bagian berikut akan mencakup sintaks, contoh, dan praktik terbaik untuk menggunakan operator dan fungsi untuk data SUPER di Amazon Redshift untuk membuka potensi penuh data semi-terstruktur Anda.

Operator aritmatika

Nilai SUPER mendukung semua operator aritmatika dasar +, -, *,/,% menggunakan pengetikan dinamis. Jenis operasi yang dihasilkan tetap sebagai SUPER. Untuk semua operator, kecuali untuk operator biner +, operan input harus berupa angka. Jika tidak, Amazon Redshift mengembalikan null. Perbedaan antara nilai desimal dan floating-point dipertahankan saat Amazon Redshift menjalankan operator ini dan tipe dinamis tidak berubah. Namun, skala desimal berubah saat Anda menggunakan perkalian dan pembagian. Luapan aritmatika masih menyebabkan kesalahan kueri, tidak diubah menjadi null. Operator biner + melakukan penambahan jika input adalah angka atau penggabungan jika input adalah string. Jika satu operan adalah string dan operan lainnya adalah angka, hasilnya adalah null. Operator awalan unary + dan - mengembalikan null jika nilai SUPER bukan angka seperti yang ditunjukkan pada contoh berikut:

```
SELECT (c_orders[0]. o_orderkey + 0.5) * c_orders[0]. o_orderkey / 10 AS math FROM
 customer_orders_lineitem;
           math
----------------------------
 1757958232200.1500
(1 row)
```

Pengetikan dinamis memungkinkan nilai desimal di SUPER memiliki skala yang berbeda. Amazon Redshift memperlakukan nilai desimal seolah-olah mereka adalah tipe statis yang berbeda dan memungkinkan semua operasi matematika. Amazon Redshift menghitung skala yang dihasilkan secara dinamis berdasarkan skala operan. Jika salah satu operan adalah bilangan floating-point, maka Amazon Redshift mempromosikan operan lainnya ke bilangan floating-point dan menghasilkan hasilnya sebagai bilangan floating-point.

Fungsi aritmatika

Amazon Redshift mendukung fungsi aritmatika berikut untuk kolom SUPER. Mereka mengembalikan null jika inputnya bukan angka:

- LANTAI. Untuk informasi selengkapnya, lihat Fungsi LANTAI.
- CEIL dan PLAFON. Untuk informasi selengkapnya, lihat Fungsi CEILING (atau CEIL).
- BULAT. Untuk informasi selengkapnya, lihat Fungsi ROUND.
- BATANG. Untuk informasi selengkapnya, lihat Fungsi TRUNC.
- PERUT. Untuk informasi selengkapnya, lihat Fungsi ABS.

Contoh berikut menggunakan fungsi aritmatika untuk menanyakan data:

```
SELECT x, FLOOR(x), CEIL(x), ROUND(x)
FROM (
    SELECT (c_orders[0]. o_orderkey + 0.5) * c_orders[0].o_orderkey / 10 AS x
    FROM customer_orders_lineitem
    );

         x          |     floor     |     ceil      |     round
--------------------+---------------+---------------+---------------
 1389636795898.0500  | 1389636795898  | 1389636795899  | 1389636795898
```

Fungsi ABS mempertahankan skala masukan desimal sementara FLOOR, CEIL. ROUND menghilangkan skala masukan desimal.

Fungsi array

Amazon Redshift mendukung komposisi array dan fungsi utilitas berikut:

- SUSUNAN. Untuk informasi selengkapnya, lihat Fungsi ARRAY.
- ARRAY_CONCAT. Untuk informasi selengkapnya, lihat Fungsi ARRAY_CONCAT.
- ARRAY_BERISI. Untuk informasi selengkapnya, lihat Fungsi ARRAY_CONTAINS.
- ARRAY_DISTINCT. Untuk informasi selengkapnya, lihat Fungsi ARRAY_DISTINCT.
- ARRAY_KECUALI. Untuk informasi selengkapnya, lihat Fungsi ARRAY_EXCEPT.
- ARRAY_RATAKAN. Untuk informasi selengkapnya, lihat Fungsi ARRAY_FLATTEN.
- ARRAY_PERSIMPANGAN. Untuk informasi selengkapnya, lihat Fungsi ARRAY_INTERSECTION.
- POSISI ARRAY. Untuk informasi selengkapnya, lihat Fungsi ARRAY_POSITION.
- POSISI ARRAY. Untuk informasi selengkapnya, lihat Fungsi ARRAY_POSITIONS.
- SUSUN_SUSUNAN. Untuk informasi selengkapnya, lihat Fungsi ARRAY_SORT.
- ARRAY_UNION. Untuk informasi selengkapnya, lihat Fungsi ARRAY_UNION.

- ARRAY_TUMPANG TINDIH. Untuk informasi selengkapnya, lihat Fungsi ARRAYS_OVERLAP.
- GET_ARRAY_PANJANG. Untuk informasi selengkapnya, lihat Fungsi GET_ARRAY_LENGTH.
- PISAH_KE_ARRAY. Untuk informasi selengkapnya, lihat Fungsi SPLIT_TO_ARRAY.
- SUBARRAY. Untuk informasi selengkapnya, lihat Fungsi SUBARRAY.

Anda dapat membuat array SUPER dari nilai dalam tipe data Amazon Redshift menggunakan fungsi ARRAY, termasuk nilai SUPER lainnya. Contoh berikut menggunakan fungsi variadik ARRAY:

```
SELECT ARRAY(1, c.c_custkey, NULL, c.c_name, 'abc') FROM customer_orders_lineitem c;
                              array
 ------------------------------------------------------
[1,8401,null,""Customer#000008401"",""abc""]
[1,9452,null,""Customer#000009452"",""abc""]
[1,9451,null,""Customer#000009451"",""abc""]
[1,8251,null,""Customer#000008251"",""abc""]
[1,5851,null,""Customer#000005851"",""abc""]
(5 rows)
```

Contoh berikut menggunakan penggabungan array dengan fungsi ARRAY_CONCAT:

```
SELECT ARRAY_CONCAT(JSON_PARSE('[10001,10002]'),JSON_PARSE('[10003,10004]'));

             array_concat
-------------------------------------
 [10001,10002,10003,10004]
(1 row)
```

Contoh berikut menggunakan manipulasi array dengan fungsi SUBARRAY yang mengembalikan subset dari array input.

```
SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);

   subarray
---------------
 ["c","d","e"]
(1 row))
```

Contoh berikut menggabungkan beberapa level array ke dalam satu array menggunakan ARRAY_FLATTEN:

```
SELECT x, ARRAY_FLATTEN(x) FROM (SELECT ARRAY(1, ARRAY(2, ARRAY(3, ARRAY()))) AS x);

     x           | array_flatten
 ----------------+---------------
 [1,[2,[3,[]]]]  | [1,2,3]
(1 row)
```

Fungsi array ARRAY_CONCAT dan ARRAY_FLATTEN menggunakan aturan pengetikan dinamis. Mereka mengembalikan null alih-alih kesalahan jika inputnya bukan array. Fungsi GET_ARRAY_LENGTH mengembalikan panjang array SUPER yang diberikan objek atau jalur array.

```
SELECT c_name
FROM customer_orders_lineitem
WHERE GET_ARRAY_LENGTH(c_orders) = (
    SELECT MAX(GET_ARRAY_LENGTH(c_orders))
    FROM customer_orders_lineitem
    );
```

Contoh berikut membagi string ke array string menggunakan SPLIT_TO_ARRAY. Fungsi ini menggunakan pembatas sebagai parameter opsional. Jika tidak ada pembatas yang tidak ada, maka defaultnya adalah koma.

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');

   split_to_array
---------------------
 ["12","345","6789"]
(1 row)
```

Perilaku pengumpulan

Dengan SUPER, Anda dapat mengatur kolasi kolom dengan CREATE TABLE, mengambil set kolasi default dengan CREATE DATABASE, dan mengatur kolasi ekspresi dengan fungsi COLLATE. https://docs.aws.amazon.com/redshift/latest/dg/r_COLLATE.html

Pengaturan kolasi berlaku untuk semua operator perbandingan dan nilai string yang disimpan di SUPER, apakah itu nilai string, string di dalam array SUPER, atau nilai objek SUPER. Untuk objek SUPER, perilaku kolasi hanya berlaku untuk nilai dan bukan atribut. Misalnya, perbandingan nilai yang memiliki kolasi case insensitive `{"attribute": "a"} = {"attribute":`

`"A"}` mengembalikan true, sementara `{"attribute": "a"} = {"ATTRIBUTE": "a"}` mengembalikan false.

Fungsi informasi

Kolom data SUPER mendukung fungsi inspeksi yang mengembalikan tipe dinamis dan informasi tipe lainnya tentang nilai SUPER. Contoh yang paling umum adalah fungsi skalar JSON_TYPEOF yang mengembalikan VARCHAR dengan nilai boolean, number, string, object, array, atau null, tergantung pada tipe dinamis dari nilai SUPER. Amazon Redshift mendukung fungsi boolean berikut untuk kolom data SUPER:

- Fungsi IS_ARRAY
- Fungsi IS_BIGINT
- Fungsi IS_CHAR
- Fungsi IS_DECIMAL
- Fungsi IS_FLOAT
- Fungsi IS_INTEGER
- Fungsi IS_OBJECT
- Fungsi IS_SCALAR
- Fungsi IS_SMALLINT
- Fungsi IS_VARCHAR

Untuk informasi selengkapnya tentang fungsi informasi tipe SUPER, lihatFungsi informasi tipe SUPER.

Fungsi objek

Berikut ini adalah fungsi objek SQL yang didukung Amazon Redshift untuk membuat dan mengoperasikan objek tipe SUPER:

- Fungsi GET_NUMBER_ATTRIBUTES
- Fungsi LOWER_ATTRIBUTE_NAMES
- Fungsi OBJEK
- Fungsi OBJECT_TRANSFORM

- Fungsi UPPER_ATTRIBUTE_NAMES

Untuk informasi selengkapnya tentang fungsi objek, lihatFungsi objek.

Fungsi string

Untuk menggunakan fungsi string dengan literal string dalam tipe data SUPER, Anda harus mengonversi string literal ke tipe string sebelum menerapkan fungsi. Fungsi mengembalikan null jika input bukan string literal.

Fungsi stringsekarang mendukung hingga 16.000.000 byte.

Contoh berikut menggunakan SUBSTRING untuk mengekstrak pratinjau string dengan ukuran 5.000.000 byte yang disimpan dalam objek SUPER JSON

```
CREATE TABLE customer_data (
   customer_id INT,
   profile SUPER
);

INSERT INTO customer_data VALUES (
   1,
   JSON_PARSE('{"name": "John Doe", "description": "' || REPEAT('A', 5000000) || '"}')
);

SELECT
   customer_id,
   profile.name::VARCHAR AS name,
   SUBSTRING(profile.description::VARCHAR, 1, 50) AS description_preview
FROM customer_data;

 customer_id | name     | description_preview
-------------+----------+----------------------------------------------------
         1 | John Doe | AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Contoh berikut menunjukkan fungsi LEFT, RIGHT, dan CONCAT pada literal string dari array SUPER:

```
CREATE TABLE documents (
   doc_id INT,
```

```
   chapters SUPER
);

INSERT INTO documents VALUES (
   1,
   JSON_PARSE('["' || REPEAT('hello', 400000) || '", "' || REPEAT('world', 600000) ||
 '"]')
);

SELECT
   doc_id,
   LEFT(chapters[0]::VARCHAR, 20) AS chapter1_start,
   RIGHT(chapters[1]::VARCHAR, 20) AS chapter2_end,
   LEN(CONCAT(chapters[0]::VARCHAR, chapters[1]::VARCHAR)) AS concat_size
FROM documents;

 doc_id |    chapter1_start    |     chapter2_end     | concat_size
--------+----------------------+----------------------+-------------
      1 | hellohellohellohello | worldworldworldworld |     5000000
```

Contoh berikut menyimpan string mandiri di SUPER:

```
CREATE TABLE text_storage (
   text_id INT,
   content SUPER
);

INSERT INTO text_storage VALUES
   (1, REPEAT('A', 8000000)),
   (2, REPEAT('B', 16000000));

SELECT
   text_id,
   LEN(content::VARCHAR) AS content_length
FROM text_storage;

 text_id | content_length
---------+----------------
      1 |        8000000
      2 |       16000000
```

Konfigurasi SUPER

Anda dapat mengonfigurasi data SUPER Anda untuk skenario tertentu. Bagian berikut memberikan rincian tentang memilih dan menerapkan konfigurasi SUPER yang sesuai berdasarkan persyaratan format data Anda.

Topik

- Mode longgar dan ketat untuk SUPER
- Mengakses bidang JSON dengan nama atau atribut bidang huruf besar dan huruf campuran
- Opsi penguraian untuk SUPER

Mode longgar dan ketat untuk SUPER

Saat Anda menanyakan data SUPER, ekspresi jalur mungkin tidak cocok dengan struktur data SUPER yang sebenarnya. Jika Anda mencoba mengakses anggota objek atau elemen array yang tidak ada, Amazon Redshift mengembalikan nilai NULL jika kueri Anda dijalankan dalam mode lax default. Jika Anda menjalankan kueri dalam mode ketat, Amazon Redshift mengembalikan kesalahan. Parameter sesi berikut dapat diatur untuk mengaktifkan atau menonaktifkan mode lax.

Contoh berikut menggunakan parameter sesi untuk mengaktifkan mode lax.

```
SET navigate_super_null_on_error=ON;  --default lax mode for navigation

SET cast_super_null_on_error=ON;  --default lax mode for casting

SET parse_super_null_on_error=OFF;  --default strict mode for ingestion
```

Mengakses bidang JSON dengan nama atau atribut bidang huruf besar dan huruf campuran

Ketika nama atribut JSON Anda menggunakan huruf besar atau huruf campuran, Anda harus dapat menavigasi struktur tipe SUPER dengan cara yang peka huruf besar/kecil. Untuk melakukan itu, Anda dapat mengonfigurasi `enable_case_sensitive_super_attribute` ke TRUE dan menggunakan nama atribut huruf besar dan huruf campuran langsung dalam kueri Anda tanpa membungkusnya dalam tanda kutip ganda.

Contoh berikut menggambarkan cara `enable_case_sensitive_super_attribute` mengatur data query.

```
SET enable_case_sensitive_super_attribute to TRUE;

-- Accessing JSON attribute names with uppercase and mixed-case names
SELECT json_table.data.ITEMS.Name,
       json_table.data.price
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;

 Name | price
------+-------
 "TV" | 345
(1 row)

RESET enable_case_sensitive_super_attribute;

-- After resetting the above configuration, the following query accessing JSON
 attribute names with uppercase and mixed-case names should return null (if in lax
 mode).
SELECT json_table.data.ITEMS.Name,
       json_table.data.price
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;

 name | price
------+-------
      | 345
(1 row)
```

Atau, Anda dapat mengonfigurasi `enable_case_sensitive_identifier` ke TRUE dan membungkus nama atribut huruf besar dan huruf campuran dengan tanda kutip ganda. Untuk informasi selengkapnya, lihat enable_case_sensitive_identifier.

Opsi penguraian untuk SUPER

Saat Anda menggunakan fungsi JSON_PARSE untuk mengurai string JSON menjadi nilai SUPER, batasan tertentu berlaku:

- Nama atribut yang sama tidak dapat muncul di objek yang sama, tetapi dapat muncul di objek bersarang. Opsi `json_parse_dedup_attributes` konfigurasi memungkinkan JSON_PARSE untuk menyimpan hanya kejadian terakhir dari atribut duplikat alih-alih mengembalikan kesalahan.

- Literal string individu dalam objek SUPER memiliki batas ukuran 16.000.000 byte. Opsi `json_parse_truncate_strings` konfigurasi memungkinkan JSON_PARSE () untuk secara otomatis memotong string yang lebih panjang dari batas ini tanpa mengembalikan kesalahan. Perilaku ini mempengaruhi nilai string saja dan bukan nama atribut.

Untuk informasi selengkapnya tentang fungsi JSON_PARSE, lihat. Fungsi JSON_PARSE

Contoh berikut menunjukkan cara mengatur opsi `json_parse_dedup_attributes` konfigurasi ke perilaku default mengembalikan kesalahan untuk atribut duplikat.

```
SET json_parse_dedup_attributes=OFF;  --default behavior of returning error instead of
 de-duplicating attributes
```

Contoh berikut menunjukkan cara mengatur opsi `json_parse_truncate_strings` konfigurasi untuk perilaku default mengembalikan kesalahan untuk string yang lebih panjang dari batas ini.

```
SET json_parse_truncate_strings=OFF;  --default behavior of returning error instead of
 truncating strings
```

Batasan

Dengan Amazon Redshift, Anda dapat bekerja dengan tipe data SUPER untuk menyimpan dan menanyakan data semi-terstruktur seperti JSON, Avro, atau Ion. Batasan tipe data SUPER mengacu pada batasan dan batas saat menggunakan tipe data ini di Amazon Redshift. Bagian berikut memberikan rincian tentang batasan spesifik tipe data SUPER, seperti ukuran maksimum, tingkat sarang, dan tipe data yang didukung dalam data semi-terstruktur.

- Anda tidak dapat mendefinisikan kolom SUPER sebagai kunci distribusi atau pengurutan.
- Objek SUPER individu dapat menampung hingga 16 MB data.
- Kedalaman sarang maksimum untuk array dan struktur tipe data SUPER adalah 1.000.
- Setiap string literal yang disimpan dalam objek SUPER tunggal dibatasi hingga 16.000.000 byte.
- Jika tidak, nilai individual dalam objek SUPER dibatasi hingga panjang maksimum jenis Amazon Redshift yang sesuai.
- Saat Anda menggunakan perintah COPY untuk memuat data ke kolom SUPER dari format berbasis baris (TEXT, CSV, JSON, AVRO), seluruh baris harus muat dalam buffer pemindai COPY, yang secara default berukuran 64 MB. Batas pemindai 64 MB tidak berlaku untuk format

kolumnar (PARQUET, ORC). Dalam semua kasus, setiap objek SUPER dibatasi hingga 16 MB terlepas dari format sumber.

- Opsi TRUNCATECOLUMNS dari perintah COPY, jika didukung, hanya berlaku untuk kolom VARCHAR dan tidak memotong nilai string di kolom SUPER.
- Anda tidak dapat melakukan operasi pembaruan atau transformasi sebagian pada kolom SUPER.
- Anda tidak dapat menggunakan tipe data SUPER dan aliasnya di join kanan atau gabungan luar penuh.
- Tipe data SUPER tidak mendukung XML sebagai format serialisasi masuk atau keluar.
- Dalam klausa FROM dari subkueri (yang berkorelasi atau tidak) yang mereferensikan variabel tabel untuk un, query hanya dapat merujuk ke tabel induknya dan bukan tabel lain.
- Anda tidak dapat melakukan unatau unpivoting objek pada kolom SUPER yang dihasilkan oleh ekspresi tabel umum rekursif (). `WITH RECURSIVE` Untuk informasi lebih lanjut tentang CTE rekursif, lihat. Ekspresi tabel umum rekursif
- Batasan casting

  Nilai SUPER dapat ditransfer ke dan dari tipe data lain, dengan pengecualian berikut:

  - Amazon Redshift tidak membedakan bilangan bulat dan desimal skala 0.
  - Jika skala tidak nol, tipe data SUPER memiliki perilaku yang sama dengan tipe data Amazon Redshift lainnya, kecuali Amazon Redshift mengonversi SUPER-related kesalahan menjadi nol, seperti yang ditunjukkan pada contoh berikut.

```
SELECT 5::bool;
 bool
-------
 True
(1 row)

SELECT 5::decimal::bool;
ERROR:  cannot cast type numeric to boolean

SELECT 5::super::bool;
 bool
-------
 True
(1 row)

SELECT 5.0::bool;
ERROR:  cannot cast type numeric to boolean
```

```
SELECT 5.0::super::bool;
 bool
-------
(1 row)
```

- Amazon Redshift tidak mentransmisikan tipe tanggal dan waktu ke tipe data SUPER. Amazon Redshift hanya dapat menampilkan tipe data tanggal dan waktu dari tipe data SUPER, seperti yang ditunjukkan pada contoh berikut.

```
SELECT o.o_orderdate FROM customer_orders_lineitem c,c.c_orders o;
  order_date
------------------
 "2001-09-08"
(1 row)


SELECT JSON_TYPEOF(o.o_orderdate) FROM customer_orders_lineitem c,c.c_orders o;
 json_typeof
-----------------
 string
(1 row)


SELECT o.o_orderdate::date FROM customer_orders_lineitem c,c.c_orders o;
 order_date
----------------
 2001-09-08
(1 row)


--date/time cannot be cast to super
SELECT '2019-09-09'::date::super;
ERROR:  cannot cast type date to super
```

- Cast dari nilai non-skalar (objek dan array) ke string mengembalikan NULL. Untuk membuat serial nilai-nilai non-skalar ini dengan benar, jangan melemparkannya. Sebagai gantinya, gunakan `json_serialize` untuk memberikan nilai non-skalar. `json_serialize`Fungsi mengembalikan varchar. Biasanya, Anda tidak perlu mentransmisikan nilai non-skalar ke varchar karena Amazon Redshift secara implisit melakukan serialisasi seperti yang ditunjukkan pada contoh pertama berikut.

```
SELECT r_nations FROM region_nations WHERE r_regionkey=300;
   r_nations
----------------
 [1,"abc",null]
(1 row)

SELECT r_nations::varchar FROM region_nations WHERE r_regionkey=300;
 r_nations
-------------
(1 row)

SELECT JSON_SERIALIZE(r_nations) FROM region_nations WHERE r_regionkey=300;
 json_serialize
-----------------
 [1,"abc",null]
(1 row)
```

- Amazon Redshift tidak mendukung fungsi volatile, seperti RANDOM () atau TIMEOFDAY (), dalam subkueri yang mengunci tabel luar atau sisi kiri (LHS) fungsi IN dengan subkueri tersebut.

Jenis data SUPER dan tampilan yang terwujud

Dengan Amazon Redshift, Anda dapat menggunakan tampilan yang terwujud untuk meningkatkan kinerja dan fleksibilitas kueri yang dijalankan terhadap tipe data SUPER. Dengan tipe data SUPER, Anda dapat menyimpan superset kolom dari tabel dasar dalam tampilan terwujud, memungkinkan Anda menanyakan tampilan terwujud secara langsung tanpa bergabung dengan tabel dasar. Bagian berikut menunjukkan cara membuat dan menggunakan tampilan terwujud dengan tipe data SUPER di Amazon Redshift.

Amazon Redshift mendukung tampilan terwujud yang menggabungkan kolom tipe data SUPER dan kueri PartiQL. Tampilan terwujud dapat menyegarkan secara bertahap, sedangkan Amazon Redshift hanya memperbarui data yang telah berubah di tabel dasar sejak operasi penyegaran terakhir. Pendekatan pembaruan selektif ini membuat proses penyegaran lebih efisien daripada perhitungan ulang penuh. Untuk informasi selengkapnya tentang tampilan yang terwujud, lihat[Tampilan yang terwujud di Amazon Redshift](#).

Mempercepat kueri PartiQL

Anda dapat menggunakan tampilan terwujud untuk mempercepat kueri PartiQL yang menavigasi data hierarkis and/or yang tidak ada di kolom SUPER. Dengan membuat satu atau lebih tampilan terwujud untuk mencabik-cabik nilai SUPER menjadi beberapa kolom dan memanfaatkan organisasi kolom kueri analitis Amazon Redshift, Anda pada dasarnya dapat mengekstrak dan menormalkan data bersarang. Tingkat normalisasi tergantung pada seberapa banyak upaya yang Anda lakukan untuk mengubah data SUPER menjadi data kolumnar konvensional.

Topik berikut menampilkan contoh memecah, atau merobek, data kompleks menjadi kolom yang lebih kecil, serta membuat kolom skalar dari data yang diparut untuk meningkatkan kinerja.

Topik

- Menghancurkan data semi-terstruktur menjadi kolom SUPER dengan tampilan yang terwujud
- Membuat kolom skalar Amazon Redshift dari data yang diparut

Menghancurkan data semi-terstruktur menjadi kolom SUPER dengan tampilan yang terwujud

Dengan Amazon Redshift, Anda dapat meningkatkan kinerja kueri dengan merobek data menjadi kolom SUPER menggunakan tampilan yang terwujud. Shredding mengacu pada proses memecah tipe data kompleks seperti JSON atau XML semi-terstruktur menjadi kolom yang lebih kecil dan lebih datar. Kolom SUPER adalah bentuk khusus penyimpanan kolom yang dioptimalkan untuk memindai data parut dengan cepat.

Bagian berikut menjelaskan langkah-langkah dan pertimbangan untuk merobek data menjadi kolom SUPER menggunakan tampilan terwujud di Amazon Redshift.

Contoh berikut menunjukkan definisi tampilan terwujud yang mencabik-cabik data bersarang dengan kolom yang dihasilkan masih menjadi tipe data SUPER.

```
SELECT c.c_name, o.o_orderstatus
FROM customer_orders_lineitem c, c.c_orders o;
```

Contoh berikut menunjukkan definisi tampilan terwujud yang membuat kolom skalar Amazon Redshift konvensional dari data yang diparut.

```
SELECT c.c_name, c.c_orders[0].o_totalprice
```

```
FROM customer_orders_lineitem c;
```

Anda dapat membuat satu tampilan terwujud super_mv untuk mempercepat kedua kueri.

Untuk menjawab kueri pertama, Anda harus mewujudkan atribut o_orderstatus. Anda dapat menghilangkan atribut c_name karena tidak melibatkan navigasi bersarang atau tidak bersarang. Anda juga harus menyertakan dalam tampilan terwujud atribut c_custkey dari customer_orders_lineitem untuk dapat menggabungkan tabel dasar dengan tampilan terwujud.

Untuk menjawab kueri kedua, Anda juga harus mewujudkan atribut o_totalprice dan indeks array o_idx dari c_orders. Oleh karena itu, Anda dapat mengakses indeks 0 dari c_orders.

```
CREATE MATERIALIZED VIEW super_mv distkey(c_custkey) sortkey(c_custkey) AS (
  SELECT c_custkey, o.o_orderstatus, o.o_totalprice, o_idx
  FROM customer_orders_lineitem c, c.c_orders o AT o_idx
);
```

Atribut o_orderstatus dan o_totalprice dari tampilan terwujud super_mv adalah SUPER.

Tampilan terwujud super_mv akan diperbarui secara bertahap setelah perubahan pada tabel dasar customer_orders_lineitem.

```
REFRESH MATERIALIZED VIEW super_mv;
INFO: Materialized view super_mv was incrementally updated successfully.
```

Untuk menulis ulang kueri PartiQL pertama sebagai kueri SQL biasa, gabungkan customer_orders_lineitem dengan super_mv sebagai berikut.

```
SELECT c.c_name, v.o_orderstatus
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey;
```

Demikian pula, Anda dapat menulis ulang kueri PartiQL kedua. Contoh berikut menggunakan filter pada o_idx = 0.

```
SELECT c.c_name, v.o_totalprice
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey
WHERE v.o_idx = 0;
```

Dalam perintah CREATE MATERIALIZED VIEW, tentukan c_custkey sebagai kunci distribusi dan kunci pengurutan untuk super_mv. Amazon Redshift melakukan penggabungan gabungan yang efisien, dengan asumsi bahwa c_custkey juga merupakan kunci distribusi dan kunci pengurutan customer_orders_lineitem. Jika bukan itu masalahnya, Anda dapat menentukan c_custkey sebagai kunci pengurutan dan kunci distribusi customer_orders_lineitem sebagai berikut.

```
ALTER TABLE customer_orders_lineitem
ALTER DISTKEY c_custkey, ALTER SORTKEY (c_custkey);
```

Gunakan pernyataan EXPLAIN untuk memverifikasi bahwa Amazon Redshift melakukan gabungan gabungan pada kueri yang ditulis ulang.

```
EXPLAIN
      SELECT c.c_name, v.o_orderstatus
      FROM customer_orders_lineitem c JOIN super_mv v ON c.c_custkey = v.c_custkey;

      QUERY PLAN

 ------------------------------------------------------------------------------------------------
      XN Merge Join DS_DIST_NONE  (cost=0.00..34701.82 rows=1470776 width=27)
      Merge Cond: ("outer".c_custkey = "inner".c_custkey)
      ->  XN Seq Scan on mv_tbl__super_mv__0 derived_table2  (cost=0.00..14999.86
 rows=1499986 width=13)
      ->  XN Seq Scan on customer_orders_lineitem c  (cost=0.00..999.96 rows=99996
 width=30)
      (4 rows)
```

Membuat kolom skalar Amazon Redshift dari data yang diparut

Data tanpa skema yang disimpan di SUPER dapat memengaruhi kinerja Amazon Redshift. Misalnya, filter predikat atau gabungkan kondisi karena pemindaian terbatas jangkauan tidak dapat menggunakan peta zona secara efektif. Pengguna dan alat BI dapat menggunakan tampilan terwujud sebagai presentasi konvensional data dan meningkatkan kinerja kueri analitis.

Kueri berikut memindai tampilan terwujud `super_mv` dan memfilter. `o_orderstatus`

```
SELECT c.c_name, v.o_totalprice
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey
WHERE v.o_orderstatus = 'F';
```

Periksa `stl_scan` untuk memverifikasi bahwa Amazon Redshift tidak dapat menggunakan peta zona secara efektif pada pemindaian terbatas jangkauan. `o_orderstatus`

```
SELECT slice, is_rrscan FROM stl_scan
WHERE query = pg_last_query_id() AND perm_table_name LIKE '%super_mv%';

 slice | is_rrscan
-------+-----------
     0 | f
     1 | f
     5 | f
     4 | f
     2 | f
     3 | f
(6 rows)
```

Contoh berikut menyesuaikan tampilan terwujud `super_mv` untuk membuat kolom skalar dari data yang diparut. Dalam hal ini, Amazon Redshift melakukan pemindahan `o_orderstatus` dari SUPER ke VARCHAR. Selain itu, tent `o_orderstatus` ukan sebagai kunci pengurutan untuk`super_mv`.

```
CREATE MATERIALIZED VIEW super_mv distkey(c_custkey) sortkey(c_custkey, o_orderstatus)
 AS (
  SELECT c_custkey, o.o_orderstatus::VARCHAR AS o_orderstatus, o.o_totalprice, o_idx
  FROM customer_orders_lineitem c, c.c_orders o AT o_idx
);
```

Setelah menjalankan kembali kueri, verifikasi bahwa Amazon Redshift sekarang dapat menggunakan peta zona.

```
SELECT v.o_totalprice
FROM super_mv v
WHERE v.o_orderstatus = 'F';
```

Anda dapat memverifikasi bahwa pemindaian terbatas jangkauan sekarang menggunakan peta zona sebagai berikut.

```
SELECT slice, is_rrscan FROM stl_scan
WHERE query = pg_last_query_id() AND perm_table_name LIKE '%super_mv%';

 slice | is_rrscan
-------+-----------
```

```
     0 | t
     1 | t
     2 | t
     3 | t
     4 | t
     5 | t
(6 rows)
```

Contoh penggunaan data semi-terstruktur di Amazon Redshift

Contoh berikut menunjukkan cara bekerja dengan data semi-terstruktur di Amazon Redshift menggunakan sintaks PartiQL. Anda akan membuat tabel sampel untuk memuat kumpulan sampel data semi-terstruktur, kemudian menanyakan objek data semi-terstruktur dalam berbagai kasus penggunaan.

> **Note**
>
> Kami menyarankan Anda mengatur opsi `enable_case_sensitive_super_attribute` konfigurasi ke true sebelum bekerja dengan tipe data SUPER. Untuk informasi selengkapnya, lihat enable_case_sensitive_super_atribut.

Memuat data semi-terstruktur

Pernyataan berikut membuat tabel sampel dan memuat objek JSON sampel ke kolom `all_data` SUPER.

```
DROP TABLE IF EXISTS test_json;

SET enable_case_sensitive_super_attribute TO true;

CREATE TABLE test_json (all_data SUPER);

INSERT INTO test_json VALUES (JSON_PARSE('
{
   "data":{
      "pnr":{
         "type":"pnr",
         "pnrid":"123PQRS-2024-09-20",
         "bookingIdentifier":"123PQRS",
```

```
         "version":"5",
         "triggerType":"",
         "events":[
            {
               "eventType":"UPDATED",
               "type":"PART",
               "id":"123PQRS-2024-09-20-HO-1"
            },
            {
               "eventType":"CREATED",
               "type":"ABC",
               "id":"123PQRS-2024-09-20-OT-38"
            }
         ],
         "create":{
            "pnrCreateDate":"2024-09-20T16:56:00Z",
            "officeID":"OFFCID1234",
            "officeIDCategory":"Email"
         },
         "lastModification":{
            "dateTime":"2024-09-20T17:09:00Z"
         },
         "PARTDetails":[
            {
               "path":"1",
               "TrainPARTs":[
                  {
                     "PARTID":"123PQRS-2024-09-20-HO-1",
                     "departure":{
                        "departureStation":"XYZ",
                        "departureTimeLocal":"2024-10-03T06:30:00",
                        "departureTimeGMT":"2024-10-03T10:30:00Z"
                     },
                     "arrival":{
                        "arrivalStation":"ABC",
                        "arrivalTimeLocal":"2024-10-03T08:20:00",
                        "arrivalTimeGMT":"2024-10-03T15:20:00Z"
                     },
                     "marketing":{
                        "carrierCode":"XX",
                        "TrainNumber":"100"
                     },
                     "operating":{
                        "carrierCode":"YY",
```

```
                  "TrainNumber":"100-A"
               },
               "status":"ON",
               "aircraft":{
                  "code":"222"
               },
               "class":"WC",
               "first":"Y",
               "seating":[

               ]
            }
         ]
      }
   ],
   "commuterInformation":[
      {
         "commuterID":"2",
         "commuterPNR":"123PQRS-2024-09-20-RO-2",
         "commuterTypeCode":"DOM",
         "firstName":"JOHN",
         "lastName":"MILLER"
      }
   ],
   "contactDetail":[
      {
         "emailContacts":[
            {
               "id":"123PQRS-2024-09-20-OT-4",
               "contact":"JOHNMILLER@EXAMPLE.COM",
               "purpose":[
                  "BUSINESS"
               ],
               "commuter":[
                  "123PQRS-2024-09-20-RO-2"
               ],
               "language":"EN"
            },
            {
               "id":"123PQRS-2024-09-20-OT-5",
               "contact":"HARVEYCORMIER@EXAMPLE.COM",
               "purpose":[
                  "NOTIFICATION"
               ],
```

```
                    "commuter":[
                       "123PQRS-2024-09-20-RO-2"
                    ],
                    "language":"EN"
                 }
              ]
           },
           {
              "phoneContacts":[
                 {
                    "id":"123PQRS-2024-09-20-OT-3",
                    "contact":"1234567890",
                    "purpose":[
                       "NOTIFICATION"
                    ],
                    "commuter":[
                       "123PQRS-2024-09-20-RO-2"
                    ],
                    "language":""
                 }
              ]
           },
           {
              "addressInfo":[
                 {
                    "id":"123PQRS-2024-09-20-OT-6",
                    "addressline":[
                       "112 PORT STREET"
                    ],
                    "provinceState":"CA",
                    "cityName":"SAN JOSE",
                    "postalCode":"12345",
                    "countryCode":"USA",
                    "purpose":[
                       "MAILING"
                    ],
                    "commuter":[
                       "123PQRS-2024-09-20-RO-2"
                    ]
                 }
              ]
           }
        ],
        "PendingService":[
```

```
            {
               "id":"123PQRS-2024-09-20-OT-26",
               "code":"MONO",
               "status":"",
               "text":"Broken Seat at Coach-No XYZ123 Seat-No 567",
               "trainCode":"WC-1",
               "TrainsArray":[
                  "123PQRS-2024-09-20-HO-1"
               ],
               "commuter":[
                  "123PQRS-2024-09-20-RO-2"
               ]
            },
            {
               "id":"123PQRS-2024-09-20-OT-27",
               "code":"OTHS",
               "status":"",
               "text":"Broken Seat at Coach-No XYZ567 Seat-No 111",
               "trainCode":"WC-1",
               "TrainsArray":[
                  "123PQRS-2024-09-20-HO-1"
               ],
               "commuter":[
                  "123PQRS-2024-09-20-RO-2"
               ]
            },
            {
               "id":"123PQRS-2024-09-20-OT-28",
               "code":"OTHS",
               "status":"",
               "text":"Broken Seat at Coach-No XYZ890 Seat-No 123",
               "trainCode":"WC-1",
               "TrainsArray":[
                  "123PQRS-2024-09-20-HO-1"
               ],
               "commuter":[
                  "123PQRS-2024-09-20-RO-2"
               ]
            },
            {
               "id":"123PQRS-2024-09-20-OT-29",
               "code":"OTHS",
               "status":"",
               "text":"Broken Seat at Coach-No XYZ111 Seat-No 333",
```

```
            "trainCode":"WC-1",
            "TrainsArray":[
               "123PQRS-2024-09-20-HO-1"
            ],
            "commuter":[
               "123PQRS-2024-09-20-RO-2"
            ]
         }
      ],
      "parts": [
         {
            "partname": "prop",
            "manufacturer": "local parts co",
            "quality": 2,
            "price": 10.00
         },
         {
            "partname": "prop",
            "manufacturer": "big parts co",
            "quality": null,
            "price": 9.00
         },
         {
            "partname": "prop",
            "manufacturer": "small parts co",
            "quality": 1,
            "price": 12.00
         },
         {
            "partname": "rudder",
            "manufacturer": "local parts co",
            "quality": 1,
            "price": 2.50
         },
         {
            "partname": "rudder",
            "manufacturer": "big parts co",
            "quality": 2,
            "price": 3.75
         },
         {
            "partname": "rudder",
            "manufacturer": "small parts co",
            "quality": null,
```

```
            "price": 1.90
         },
         {
            "partname": "wing",
            "manufacturer": "local parts co",
            "quality": null,
            "price": 7.50
         },
         {
            "partname": "wing",
            "manufacturer": "big parts co",
            "quality": 1,
            "price": 15.20
         },
         {
            "partname": "wing",
            "manufacturer": "small parts co",
            "quality": null,
            "price": 11.80
         }
      ],
      "count_by_color": [
         {
            "quality": "high",
            "red": 15,
            "green": 20,
            "blue": 7
         },
         {
            "quality": "normal",
            "red": 35,
            "green": null,
            "blue": 40
         },
         {
            "quality": "low",
            "red": 10,
            "green": 23,
            "blue": null
         }
      ]
   }
},
"id":"abcdefgh-ijklmnop-qrstuvwxyz123",
```

```
   "mainIds":[
      {
         "ID":"pqrstuvwxyz-aabbcc123",
         "Source":"NYC"
      }
   ]
}
'));
```

Mengkueri data semi-terstruktur bersarang

Pernyataan berikut menggunakan notasi titik PartiQL untuk mengekstrak `pnrid` bidang, yang bersarang tiga tingkat jauh di dalam objek tingkat atas`all_data`.

```
select all_data.data.pnr.pnrid::varchar from test_json;

 pnrid
--------------------
 123PQRS-2024-09-20
```

Pernyataan berikut menggunakan notasi braket PartiQL untuk menentukan dan mengekstrak hanya elemen pertama dari `events` array yang bersarang di dalam objek tingkat atas.

```
SELECT
    all_data.data.pnr.events[0]
FROM test_json;

 events
--------------------------------
{
   "eventType":"UPDATED",
   "type":"PART",
   "id":"123PQRS-2024-09-20-HO-1"
}
```

Pernyataan berikut mengekstrak `eventType` properti hanya elemen yang ditentukan dari `events` array.

```
SELECT
    all_data.data.pnr.events[0].eventType
FROM test_json;
```

```
 eventtype
-----------
 "UPDATED"
```

Pernyataan berikut

Menggunakan **enable_case_sensitive_super_atribut dengan data semi-terstruktur**

Contoh berikut menunjukkan bagaimana opsi enable_case_sensitive_super_atribut konfigurasi mempengaruhi kueri data semi-terstruktur. Untuk informasi selengkapnya, lihat Mengakses bidang JSON dengan nama atau atribut bidang huruf besar dan huruf campuran.

Dalam pernyataan berikut, mengatur ulang opsi konfigurasi ke default false membuat kueri mengembalikan NULL untuk nama atribut kasus campuran.

```
RESET enable_case_sensitive_super_attribute;

SELECT
    all_data.data.pnr.events[0].eventType
FROM test_json;

 eventtype
-----------
NULL
```

Dalam contoh berikut, kueri sampel mengembalikan hasil yang diinginkan setelah Anda menyetel `enable_case_sensitive_super_attribute` ke true. Tidak diperlukan tanda kutip ganda di sekitar nama atribut.

```
SET enable_case_sensitive_super_attribute TO true;

SELECT
    all_data.data.pnr.events[0].eventType
FROM test_json;

 eventtype
-----------
 "UPDATED"
```

Memfilter data semi-terstruktur

Pernyataan berikut menggunakan sintaks PartiQL dalam klausa WHERE dari pernyataan yang menghitung peristiwa jenis UPDATED untuk mengambil data atribut tertentu dari dalam array. Anda dapat menggunakan sintaks ini di bagian mana pun dari kueri di mana Anda biasanya akan merujuk kolom.

```
SELECT COUNT(*)
FROM test_json
WHERE all_data.data.pnr.events[0].eventType = 'UPDATED';

 count
------
 1
```

Contoh berikut menggunakan sintaks tanda kurung dan titik PartiQL di klausa GROUP BY dan ORDER BY.

```
SELECT all_data.data.pnr.events[0].eventType::varchar,
       COUNT(*)
FROM test_json
WHERE all_data.data.pnr.events[0].eventType IS NOT NULL
GROUP BY all_data.data.pnr.events[0].eventType
ORDER BY all_data.data.pnr.events[0].eventType;

 eventtype | count
-----------+-------
 "UPDATED" | 1
```

Data semi-terstruktur yang tidak bersarang

Pernyataan berikut menggunakan gabungan PartiQL untuk menguji array. `events` Perhatikan bahwa gabungan ini berfungsi bahkan ketika jumlah indeks untuk array tidak statis.

Untuk contoh data semi-terstruktur yang tidak bersarang menggunakan UNNEST dalam klausa FROM, lihat. Contoh UNNEST

```
SELECT
a.all_data.data.pnr.type::varchar type_info,
a.all_data.data.pnr.pnrid::varchar pnr_id ,
a.all_data.data.pnr.bookingIdentifier::varchar booking_id,
```

```
a.all_data.data.pnr.version::varchar version_info,
b.eventType::varchar event_type,
b.id::varchar event_id
FROM test_json a,
  a.all_data.data.pnr.events b;

 type_info | pnr_id              | booking_id | version_info | event_type | event_id
-----------+---------------------+------------+--------------+------------
+-------------------------
 pnr       | 123PQRS-2024-09-20  | 123PQRS    | 5            | UPDATED    |
 123PQRS-2024-09-20-HO-1
 pnr       | 123PQRS-2024-09-20  | 123PQRS    | 5            | CREATED    |
 123PQRS-2024-09-20-OT-38
```

Array bersarang yang tidak bersarang

Pernyataan berikut menggunakan gabungan PartiQL untuk menguji array yang bersarang di dalam array lain.

Untuk contoh data semi-terstruktur yang tidak bersarang menggunakan UNNEST dalam klausa FROM, lihat. Contoh UNNEST

```
SELECT
a.all_data.data.pnr.type::varchar type_info,
a.all_data.data.pnr.pnrid::varchar pnr_id ,
a.all_data.data.pnr.bookingIdentifier::varchar booking_id,
a.all_data.data.pnr.version::varchar version_info,
d.id::varchar email_record_id,
d.contact::varchar email_contact,
e::varchar email_purpose,
f::varchar email_commuter
FROM test_json a,
  a.all_data.data.pnr.contactDetail c,
  c.emailContacts d,
  d.purpose e,
  d.commuter f;

 type_info | pnr_id              | booking_id | version_info | email_record_id
 | email_contact             | email_purpose | email_commuter
-----------+---------------------+------------+--------------+-------------------------
+--------------------------+---------------+-------------------------
 pnr       | 123PQRS-2024-09-20  | 123PQRS    | 5            | 123PQRS-2024-09-20-OT-4
 | JOHNMILLER@EXAMPLE.COM    | BUSINESS      | 123PQRS-2024-09-20-RO-2
```

```
 pnr       | 123PQRS-2024-09-20  | 123PQRS    | 5            | 123PQRS-2024-09-20-OT-5
 | HARVEYCORMIER@EXAMPLE.COM | NOTIFICATION  | 123PQRS-2024-09-20-RO-2
```

Menggunakan data semi-terstruktur dalam subquery

Pernyataan berikut menggunakan subquery dalam klausa WHERE untuk mengembalikan hanya subbagian dari hasil dari contoh sebelumnya.

```
SELECT
a.all_data.data.pnr.type::varchar type_info,
a.all_data.data.pnr.pnrid::varchar pnr_id ,
a.all_data.data.pnr.bookingIdentifier::varchar booking_id,
a.all_data.data.pnr.version::varchar version_info,
d.id::varchar email_record_id,
d.contact::varchar email_contact
FROM test_json a,
a.all_data.data.pnr.contactDetail c,
c.emailContacts d
WHERE (SELECT COUNT(*) FROM d.purpose e WHERE e = 'BUSINESS') > 0;

 type_info | pnr_id              | booking_id | version_info | email_record_id
 | email_contact             | email_purpose | email_commuter
-----------+---------------------+------------+--------------+-------------------------
+---------------------------+---------------+-------------------------
 pnr       | 123PQRS-2024-09-20  | 123PQRS    | 5            | 123PQRS-2024-09-20-OT-4
 | JOHNMILLER@EXAMPLE.COM    | BUSINESS      | 123PQRS-2024-09-20-RO-2
```

Menggabungkan kueri menggunakan data semi-terstruktur

Pernyataan berikut menggunakan fungsi COUNT untuk menggabungkan jumlah elemen dalam `PendingService` array.

```
SELECT
a.all_data.data.pnr.type::varchar type_info,
a.all_data.data.pnr.pnrid::varchar pnr_id ,
a.all_data.data.pnr.bookingIdentifier::varchar booking_id,
a.all_data.data.pnr.version::varchar version_info,
COUNT(*) AS total_pending_service
FROM test_json a,
  a.all_data.data.pnr.PendingService c
GROUP BY 1,2,3,4;
```

```
 type_info | pnr_id             | booking_id | version_info | total_pending_service
-----------+--------------------+------------+--------------+----------------------
 pnr       | 123PQRS-2024-09-20 | 123PQRS    | 5            | 4
```

Menggunakan data semi-terstruktur dalam tampilan terwujud

Pernyataan berikut menggunakan pernyataan dari contoh sebelumnya untuk membuat tampilan yang terwujud. Tampilan terwujud secara otomatis menyegarkan jumlah layanan yang tertunda ketika tabel dasar mendapatkan data baru.

```
CREATE MATERIALIZED VIEW mv_total_pending_service
AUTO REFRESH YES
AS
SELECT
a.all_data.data.pnr.type::varchar type_info,
a.all_data.data.pnr.pnrid::varchar pnr_id ,
a.all_data.data.pnr.bookingIdentifier::varchar booking_id,
a.all_data.data.pnr.version::varchar version_info,
COUNT(*) AS total_pending_service
FROM test_json a,
  a.all_data.data.pnr.PendingService c
GROUP BY 1,2,3,4;
```

Menggunakan PIVOT dan UNPIVOT dengan data semi-terstruktur

Pernyataan berikut menggunakan PIVOT pada `partname` kolom untuk mengembalikan harga rata-rata setiap bagian.

```
SELECT *
FROM
(
SELECT
c.partname::varchar, c.price
FROM test_json a,
  a.all_data.data.pnr.parts c)
PIVOT (AVG(price) for partname IN ('prop', 'rudder', 'wing'));


 prop       | rudder             |  wing
------------+--------------------+--------
 10.33      | 2.71               |  11.50
```

Pada contoh sebelumnya, hasilnya diubah menjadi kolom. Contoh berikut menunjukkan kueri GROUP BY yang mengembalikan harga rata-rata dalam baris, bukan dalam kolom.

```
SELECT partname, avg(price)
FROM (
SELECT
c.partname::varchar, c.price
FROM test_json a,
  a.all_data.data.pnr.parts c)
WHERE partname IN ('prop', 'rudder', 'wing')
GROUP BY partname;

 partname |  avg
----------+-------
 prop     | 10.33
 rudder   |  2.71
 wing     | 11.50
```

Berikut ini adalah contoh PIVOT `manufacturer` dengan kolom implisit.

```
SELECT *
FROM (
SELECT
c.quality, c.manufacturer::varchar
FROM test_json a,
  a.all_data.data.pnr.parts c) PIVOT (
count(*) FOR quality IN (1, 2, NULL)
);

 manufacturer      | 1  | 2  | null
-------------------+----+----+------
 local parts co    | 1  | 1  |  1
 big parts co      | 1  | 1  |  1
 small parts co    | 1  | 0  |  2
```

Berikut ini adalah contoh UNPIVOT pada `quality` kolom.

```
SELECT *
FROM
(
SELECT
c.quality as quality
```

```
FROM test_json a,
  a.all_data.data.pnr.parts c)
UNPIVOT (cnt FOR column_header IN (quality));

 column_header   | cnt
-----------------+----
 quality         | 2
 quality         | 1
 quality         | 1
 quality         | 2
 quality         | 1
```

Machine learning

Amazon Redshift machine learning (Amazon Redshift ML) adalah layanan berbasis cloud yang kuat yang memudahkan analis dan ilmuwan data dari semua tingkat keahlian untuk menggunakan teknologi pembelajaran mesin. Amazon Redshift ML menggunakan model untuk menghasilkan hasil. Anda dapat menggunakan model dengan cara berikut:

- Anda dapat memberikan data yang ingin Anda latih model, dan metadata yang terkait dengan input data ke Amazon Redshift. Kemudian Amazon Redshift ML membuat model di Amazon SageMaker AI yang menangkap pola dalam data input. Dengan menggunakan data Anda sendiri untuk model, Anda dapat menggunakan Amazon Redshift ML untuk mengidentifikasi tren dalam data, seperti prediksi churn, nilai seumur hidup pelanggan, atau prediksi pendapatan. Anda dapat menggunakan model ini untuk menghasilkan prediksi untuk data input baru tanpa menimbulkan biaya tambahan.
- Anda dapat menggunakan salah satu Model Foundation (FM) yang disediakan oleh Amazon Bedrock, seperti Claude atau Amazon Titan. Menggunakan Amazon Bedrock, Anda dapat menggabungkan kekuatan model bahasa besar (LLM) dengan data analitik Anda di Amazon Redshift dalam beberapa langkah. Dengan menggunakan Large Language Model (LLM) eksternal, Anda dapat menggunakan Amazon Redshift untuk melakukan Natural Language Processing (NLP) pada data Anda. Anda dapat menggunakan NLP untuk aplikasi seperti pembuatan teks, analisis sentimen, atau terjemahan. Untuk informasi tentang menggunakan Amazon Bedrock dengan Amazon Redshift lihat. Integrasi Amazon Redshift ML dengan Amazon Bedrock

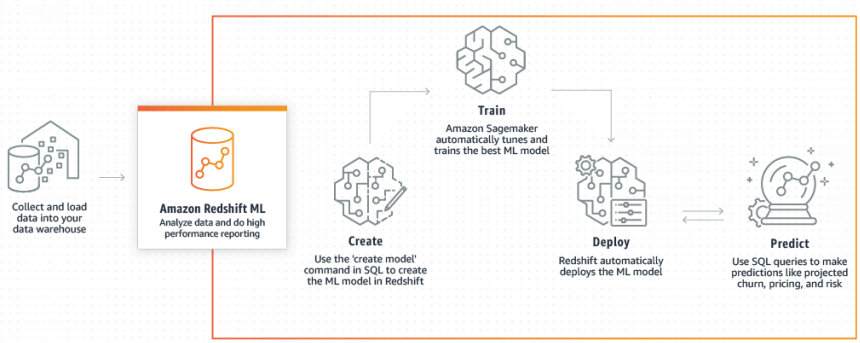
> **Note**
>
> Memilih tidak menggunakan data Anda untuk peningkatan layanan
> Jika Anda menggunakan model Amazon Bedrock, kami mendorong Anda untuk membaca AWS kebijakan tentang cara layanan Amazon Bedrock menangani data Anda. Anda harus menentukan apakah Anda perlu menggunakan kebijakan opt-out untuk mencegah layanan menggunakan data Anda untuk peningkatan model atau layanan, jika Amazon Bedrock menerapkan fungsionalitas tersebut di masa mendatang. Untuk memastikan bahwa layanan tidak menggunakan data Anda untuk tujuan tersebut, gunakan kebijakan AWS opt-out umum. Untuk informasi selengkapnya, lihat berikut ini:
>
> - Kebijakan opt-out layanan AI
> - Amazon Bedrock FAQ

> **Note**
>
> LLM dapat menghasilkan informasi yang tidak akurat atau tidak lengkap. Kami merekomendasikan memverifikasi informasi yang dihasilkan LLM untuk memastikannya akurat dan lengkap.

Cara kerja Amazon Redshift ML dengan Amazon AI SageMaker

Amazon Redshift bekerja dengan Amazon SageMaker AI Autopilot untuk secara otomatis mendapatkan model terbaik dan membuat fungsi prediksi tersedia di Amazon Redshift.

Diagram berikut menggambarkan cara kerja Amazon Redshift ML.



Alur kerja umum adalah sebagai berikut:

1. Amazon Redshift mengekspor data pelatihan ke Amazon S3.
2. Amazon SageMaker AI Autopilot memproses data pelatihan terlebih dahulu. Preprocessing melakukan fungsi-fungsi penting, seperti memperhitungkan nilai yang hilang. Ini mengakui bahwa kolom tertentu bersifat kategoris (seperti kode pos), memformatnya dengan benar untuk pelatihan, dan melakukan banyak tugas lainnya. Memilih preprosesor terbaik untuk diterapkan pada dataset pelatihan adalah masalah tersendiri, dan Amazon SageMaker AI Autopilot mengotomatiskan solusinya.
3. Amazon SageMaker AI Autopilot menemukan algoritma dan algoritma hyperparameter yang memberikan model dengan prediksi paling akurat.
4. Amazon Redshift mendaftarkan fungsi prediksi sebagai fungsi SQL di cluster Amazon Redshift Anda.
5. Saat Anda menjalankan pernyataan CREATE MODEL, Amazon Redshift menggunakan Amazon SageMaker AI untuk pelatihan. Oleh karena itu, ada biaya terkait untuk melatih model Anda. Ini adalah item baris terpisah untuk Amazon SageMaker AI di AWS tagihan Anda. Anda juga

membayar penyimpanan yang digunakan di Amazon S3 untuk menyimpan data pelatihan Anda. Inferensi menggunakan model yang dibuat dengan CREATE MODEL yang dapat Anda kompilasi dan jalankan di cluster Redshift Anda tidak dikenakan biaya. Tidak ada biaya tambahan Amazon Redshift untuk menggunakan Amazon Redshift ML.

Topik

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Biaya untuk menggunakan Amazon Redshift ML
- Memulai dengan Amazon Redshift ML
- Tutorial untuk Amazon Redshift ML
- Integrasi Amazon Redshift ML dengan Amazon Bedrock

Ikhtisar pembelajaran mesin

Dengan Amazon Redshift, Anda dapat memanfaatkan kemampuan pembelajaran mesin untuk mendapatkan wawasan berharga dari data Anda. Ikhtisar Machine Learning (ML) ini menunjukkan cara menjelajahi, memvisualisasikan, dan mempersiapkan data Anda untuk pelatihan dan penerapan model ML. Bagian berikut akan memandu Anda melalui proses memanfaatkan Amazon Redshift ML untuk membuka potensi data Anda melalui pembelajaran mesin.

Dengan menggunakan Amazon Redshift ML, Anda dapat melatih model pembelajaran mesin menggunakan pernyataan SQL dan memanggilnya dalam kueri SQL untuk prediksi.

Untuk membantu Anda mempelajari cara menggunakan Amazon Redshift ML, Anda dapat menonton video berikut: Amazon Redshift ML.

Untuk informasi tentang prasyarat untuk menyiapkan cluster Redshift atau kelompok kerja Tanpa Server, izin, dan kepemilikan untuk menggunakan Amazon Redshift ML, baca bagian berikut. Bagian ini juga menjelaskan cara kerja pelatihan dan prediksi sederhana di Amazon Redshift ML.

Bagaimana pembelajaran mesin dapat memecahkan masalah

Model pembelajaran mesin menghasilkan prediksi dengan menemukan pola dalam data pelatihan Anda dan kemudian menerapkan pola ini ke data baru. Dalam pembelajaran mesin, Anda melatih model-model ini dengan mempelajari pola-pola yang paling baik menjelaskan data Anda. Kemudian Anda menggunakan model untuk membuat prediksi (juga disebut kesimpulan) pada data baru.

Pembelajaran mesin biasanya merupakan proses berulang di mana Anda dapat terus meningkatkan akurasi prediksi dengan mengubah parameter dan meningkatkan data pelatihan Anda. Jika data berubah, pelatihan ulang model baru dengan dataset baru terjadi.

Untuk mengatasi berbagai tujuan bisnis, ada pendekatan pembelajaran mesin fundamental yang berbeda.

Pembelajaran yang diawasi di Amazon Redshift ML

Amazon Redshift mendukung pembelajaran terawasi, yang merupakan pendekatan paling umum untuk analitik perusahaan tingkat lanjut. Pembelajaran yang diawasi adalah pendekatan pembelajaran mesin yang disukai ketika Anda memiliki kumpulan data yang mapan dan pemahaman tentang bagaimana data input spesifik memprediksi berbagai hasil bisnis. Hasil ini kadang-kadang disebut label. Secara khusus, dataset Anda adalah tabel dengan atribut yang terdiri dari fitur (input) dan target (output). Misalnya, Anda memiliki tabel yang menyediakan usia dan kode pos untuk pelanggan masa lalu dan sekarang. Misalkan Anda juga memiliki bidang “aktif” yang benar untuk pelanggan saat ini dan salah untuk pelanggan yang telah menangguhkan keanggotaan mereka. Tujuan pembelajaran mesin yang diawasi adalah untuk menemukan pola usia dan kode pos yang mengarah ke churn pelanggan, seperti yang diwakili oleh pelanggan yang targetnya “Salah.” Anda dapat menggunakan model ini untuk memprediksi pelanggan yang cenderung mengalami churn, seperti menangguhkan keanggotaan mereka, dan berpotensi menawarkan insentif retensi.

Amazon Redshift mendukung pembelajaran terawasi yang mencakup regresi, klasifikasi biner, dan klasifikasi multiclass. Regresi mengacu pada masalah memprediksi nilai berkelanjutan, seperti total pengeluaran pelanggan. Klasifikasi biner mengacu pada masalah memprediksi salah satu dari dua hasil, seperti memprediksi apakah pelanggan berubah atau tidak. Klasifikasi multiclass mengacu pada masalah memprediksi salah satu dari banyak hasil, seperti memprediksi item yang mungkin diminati pelanggan. Analis data dan ilmuwan data dapat menggunakannya untuk melakukan pembelajaran yang diawasi untuk mengatasi masalah mulai dari peramalan, personalisasi, atau prediksi churn pelanggan. Anda juga dapat menggunakan pembelajaran terawasi dalam masalah seperti prediksi penjualan mana yang akan ditutup, prediksi pendapatan, deteksi penipuan, dan prediksi nilai seumur hidup pelanggan.

Pembelajaran tanpa pengawasan di Amazon Redshift ML

Pembelajaran tanpa pengawasan menggunakan algoritma pembelajaran mesin untuk menganalisis dan mengelompokkan data pelatihan tanpa label. Algoritma menemukan pola atau pengelompokan tersembunyi. Tujuannya adalah untuk memodelkan struktur atau distribusi yang mendasari dalam data untuk mempelajari lebih lanjut tentang data.

Amazon Redshift mendukung algoritma K-Means pengelompokan untuk memecahkan masalah pembelajaran tanpa pengawasan. Algoritma ini memecahkan masalah pengelompokan di mana Anda ingin menemukan pengelompokan dalam data. K-Means Algoritma mencoba untuk menemukan pengelompokan diskrit dalam data. Data yang tidak diklasifikasikan dikelompokkan dan dipartisi berdasarkan persamaan dan perbedaannya. Dengan mengelompokkan, K-Means algoritma secara iteratif menentukan sentroid terbaik dan menetapkan setiap anggota ke sentroid terdekat. Anggota terdekat dengan centroid yang sama termasuk dalam kelompok yang sama. Anggota kelompok semirip mungkin dengan anggota lain dalam kelompok yang sama, dan seberbeda mungkin dari anggota kelompok lain. Misalnya, algoritma K-Means pengelompokan dapat digunakan untuk mengklasifikasikan kota yang terkena dampak pandemi atau mengklasifikasikan kota berdasarkan popularitas produk konsumen.

Saat menggunakan K-Means algoritma, Anda menentukan input k yang menentukan jumlah cluster yang akan ditemukan dalam data. Output dari algoritma ini adalah satu set k centroid. Setiap titik data milik salah satu kluster k yang paling dekat dengannya. Setiap cluster dijelaskan oleh sentroidnya. Centroid dapat dianggap sebagai rata-rata multi-dimensi cluster. K-Means Algoritma membandingkan jarak untuk melihat seberapa berbeda cluster satu sama lain. Jarak yang lebih besar umumnya menunjukkan perbedaan yang lebih besar antara cluster.

Pra-pemrosesan data penting K-Means, karena memastikan bahwa fitur model tetap pada skala yang sama dan menghasilkan hasil yang dapat diandalkan. Amazon Redshift mendukung beberapa K-Means preprosesor untuk pernyataan CREATE MODEL, seperti, StandardScaler MinMax, dan. NumericPassthrough Jika Anda tidak ingin menerapkan prapemrosesan apa pun K-means, pilih NumericPassthrough secara eksplisit sebagai transformator. Untuk informasi selengkapnya tentang K-Means parameter, lihatBUAT MODEL dengan K-MEANS parameter.

Untuk membantu Anda mempelajari cara melakukan pelatihan tanpa pengawasan dengan K-Means pengelompokan, Anda dapat menonton video berikut: Pelatihan tanpa peng awasan dengan pengelompokan. K-Means

Syarat dan konsep untuk Amazon Redshift ML

Istilah-istilah berikut digunakan untuk menggambarkan beberapa konsep Amazon Redshift ML:

- Pembelajaran mesin di Amazon Redshift melatih model dengan satu perintah SQL. Amazon Redshift ML dan Amazon SageMaker AI mengelola semua konversi data, izin, penggunaan sumber daya, dan penemuan model yang tepat.

- Pelatihan adalah fase ketika Amazon Redshift membuat model pembelajaran mesin dengan menjalankan subset data tertentu ke dalam model. Amazon Redshift secara otomatis meluncurkan pekerjaan pelatihan di Amazon SageMaker AI dan menghasilkan model.
- Prediksi (juga disebut inferensi) adalah penggunaan model dalam kueri Amazon Redshift SQL untuk memprediksi hasil. Pada waktu inferensi, Amazon Redshift menggunakan fungsi prediksi berbasis model sebagai bagian dari kueri yang lebih besar untuk menghasilkan prediksi. Prediksi dihitung secara lokal, di cluster Redshift, sehingga memberikan throughput tinggi, latensi rendah, dan nol biaya tambahan.
- Dengan membawa model Anda sendiri (BYOM), Anda dapat menggunakan model yang dilatih di luar Amazon Redshift dengan Amazon SageMaker AI untuk inferensi dalam database secara lokal di Amazon Redshift. Amazon Redshift ML mendukung penggunaan BYOM dalam inferensi lokal.
- Inferensi lokal digunakan ketika model dilatih sebelumnya di Amazon SageMaker AI, dikompilasi oleh Amazon SageMaker AI Neo, dan dilokalkan di Amazon Redshift ML. Untuk mengimpor model yang didukung untuk inferensi lokal ke Amazon Redshift, gunakan perintah CREATE MODEL. Amazon Redshift mengimpor model SageMaker AI yang telah dilatih sebelumnya dengan memanggil Amazon SageMaker AI Neo. Anda mengkompilasi model di sana dan mengimpor model yang dikompilasi ke Amazon Redshift. Gunakan inferensi lokal untuk kecepatan yang lebih cepat dan biaya yang lebih rendah.
- Inferensi jarak jauh digunakan saat Amazon Redshift memanggil titik akhir model yang digunakan di AI. SageMaker Inferensi jarak jauh memberikan fleksibilitas untuk memanggil semua jenis model khusus dan model pembelajaran mendalam, seperti TensorFlow model yang Anda buat dan gunakan di Amazon SageMaker AI.

Yang juga penting adalah sebagai berikut:

- Amazon SageMaker AI adalah layanan pembelajaran mesin yang dikelola sepenuhnya. Dengan Amazon SageMaker AI, ilmuwan dan pengembang data dapat dengan mudah membangun, melatih, dan langsung menerapkan model ke lingkungan yang siap produksi yang dihosting. Untuk informasi tentang Amazon SageMaker AI, lihat Apa itu Amazon SageMaker AI di Panduan Pengembang SageMaker AI Amazon.
- Amazon SageMaker AI Autopilot adalah kumpulan fitur yang secara otomatis melatih dan menyetel model pembelajaran mesin terbaik untuk klasifikasi atau regresi, berdasarkan data Anda. Anda mempertahankan kontrol penuh dan visibilitas. Amazon SageMaker AI Autopilot mendukung data input dalam format tabular. Amazon SageMaker AI Autopilot menyediakan pembersihan dan prapemrosesan data otomatis, pemilihan algoritma otomatis untuk regresi linier, klasifikasi

biner, dan klasifikasi multiclass. Ini juga mendukung optimasi hyperparameter otomatis (HPO), pelatihan terdistribusi, instance otomatis, dan pemilihan ukuran cluster. Untuk informasi tentang Amazon SageMaker AI Autopilot, lihat Men gotomatiskan pengembangan model dengan Amazon SageMaker AI Autopilot di Panduan Pengembang AI Amazon SageMaker .

- Amazon Bedrock adalah layanan yang dikelola sepenuhnya yang menawarkan pilihan model dasar (FM) berkinerja tinggi dari perusahaan AI terkemuka seperti AI21 Labs, Anthropic, Cohere, Meta, Mistral AI, Stability AI, dan Amazon melalui satu API, bersama dengan serangkaian kemampuan yang diperlukan untuk membangun aplikasi AI generatif.

Pembelajaran mesin untuk pemula dan ahli

Dengan Amazon Redshift, Anda dapat memanfaatkan kemampuan Machine Learning (ML) untuk mendapatkan wawasan dari data Anda, baik Anda seorang pemula atau ahli dalam ML. Machine Learning adalah fitur Amazon Redshift yang Anda gunakan untuk membuat, melatih, dan menerapkan model ML menggunakan perintah SQL, tanpa memerlukan keahlian ML yang luas atau rekayasa data yang kompleks.

Bagian berikut memandu Anda melalui proses memanfaatkan Machine Learning, memberdayakan Anda untuk membuka potensi penuh data Anda dengan Amazon Redshift.

Dengan Amazon Redshift ML, Anda dapat melatih model dengan satu perintah SQL CREATE MODEL tunggal. Perintah CREATE MODEL membuat model yang digunakan Amazon Redshift untuk menghasilkan prediksi berbasis model dengan konstruksi SQL yang sudah dikenal.

Amazon Redshift ML sangat berguna ketika Anda tidak memiliki keahlian dalam pembelajaran mesin, alat, bahasa, algoritma, dan API. Dengan Amazon Redshift ML, Anda tidak perlu melakukan pekerjaan berat tanpa perbedaan yang diperlukan untuk berintegrasi dengan layanan pembelajaran mesin eksternal. Amazon Redshift menghemat waktu Anda untuk memformat dan memindahkan data, mengelola kontrol izin, atau membuat integrasi kustom, alur kerja, dan skrip. Anda dapat dengan mudah menggunakan algoritma pembelajaran mesin populer dan menyederhanakan kebutuhan pelatihan yang membutuhkan iterasi yang sering dari pelatihan hingga prediksi. Amazon Redshift secara otomatis menemukan algoritma terbaik dan menyesuaikan model terbaik untuk masalah Anda. Anda dapat membuat prediksi dari dalam cluster Amazon Redshift tanpa perlu memindahkan data dari Amazon Redshift atau berinteraksi dengan dan membayar layanan lain.

Amazon Redshift ML mendukung analis data dan ilmuwan data dalam menggunakan pembelajaran mesin. Hal ini juga memungkinkan para ahli pembelajaran mesin untuk menggunakan pengetahuan mereka untuk memandu pernyataan CREATE MODEL untuk hanya menggunakan aspek yang

mereka tentukan. Dengan demikian, Anda dapat mempercepat waktu yang dibutuhkan CREATE MODEL untuk menemukan kandidat terbaik, meningkatkan akurasi model, atau keduanya.

Pernyataan CREATE MODEL menawarkan fleksibilitas dalam bagaimana Anda dapat menentukan parameter untuk pekerjaan pelatihan. Dengan menggunakan fleksibilitas ini, baik pemula atau ahli pembelajaran mesin dapat memilih preprosesor, algoritma, jenis masalah, dan hyperparameter pilihan mereka. Misalnya, pengguna yang tertarik dengan customer churn mungkin menentukan untuk pernyataan CREATE MODEL bahwa jenis masalah adalah klasifikasi biner, yang berfungsi dengan baik untuk customer churn. Kemudian pernyataan CREATE MODEL mempersempit pencariannya untuk model terbaik menjadi model klasifikasi biner. Bahkan dengan pilihan pengguna dari jenis masalah, masih ada banyak opsi yang dapat digunakan oleh pernyataan CREATE MODEL. Misalnya, CREATE MODEL menemukan dan menerapkan transformasi preprocessing terbaik dan menemukan pengaturan hyperparameter terbaik.

Amazon Redshift ML mempermudah pelatihan dengan secara otomatis menemukan model terbaik menggunakan Amazon SageMaker AI Autopilot. Di balik layar, Amazon SageMaker AI Autopilot secara otomatis melatih dan menyetel model pembelajaran mesin terbaik berdasarkan data yang Anda berikan. Amazon SageMaker AI Neo kemudian menyusun model pelatihan dan membuatnya tersedia untuk prediksi di cluster Redshift Anda. Saat Anda menjalankan kueri inferensi pembelajaran mesin menggunakan model terlatih, kueri dapat menggunakan kemampuan pemrosesan paralel besar-besaran Amazon Redshift. Pada saat yang sama, kueri dapat menggunakan prediksi berbasis pembelajaran mesin.

- Sebagai pemula pembelajaran mesin, dengan pengetahuan umum tentang berbagai aspek pembelajaran mesin seperti preprosesor, algoritma, dan hyperparameter, gunakan pernyataan CREATE MODEL hanya untuk aspek yang Anda tentukan. Kemudian Anda dapat mempersingkat waktu yang dibutuhkan CREATE MODEL untuk menemukan kandidat terbaik atau meningkatkan akurasi model. Selain itu, Anda dapat meningkatkan nilai bisnis dari prediksi dengan memperkenalkan pengetahuan domain tambahan seperti jenis masalah atau tujuan. Misalnya, dalam skenario churn pelanggan, jika hasil “pelanggan tidak aktif” jarang terjadi, maka tujuan F1 sering lebih disukai daripada tujuan Akurasi. Karena model Akurasi tinggi mungkin memprediksi “pelanggan aktif” sepanjang waktu, ini menghasilkan akurasi tinggi tetapi nilai bisnis kecil. Untuk informasi tentang tujuan F1, lihat AutoMLJobObjective di Referensi API Amazon SageMaker AI.

  Untuk informasi selengkapnya tentang opsi dasar untuk pernyataan CREATE MODEL, lihatCIPTAKAN MODEL Sederhana.
- Sebagai praktisi tingkat lanjut pembelajaran mesin, Anda dapat menentukan jenis masalah dan preprosesor untuk fitur tertentu (tetapi tidak semua). Kemudian CREATE MODEL mengikuti

saran Anda pada aspek yang ditentukan. Pada saat yang sama, CREATE MODEL masih menemukan preprosesor terbaik untuk fitur yang tersisa dan hyperparameter terbaik. Untuk informasi selengkapnya tentang cara membatasi satu atau lebih aspek dari pipeline pelatihan, lihatBUAT MODEL dengan panduan pengguna.

- Sebagai ahli pembelajaran mesin, Anda dapat mengambil kendali penuh atas pelatihan dan penyetelan hyperparameter. Kemudian pernyataan CREATE MODEL tidak mencoba menemukan preprosesor, algoritma, dan hyperparameter yang optimal karena Anda membuat semua pilihan. Untuk informasi selengkapnya tentang cara menggunakan CREATE MODEL dengan AUTO OFF, lihatBUAT model XGBoost dengan AUTO OFF.
- Sebagai insinyur data, Anda dapat membawa model XGBoost yang telah dilatih sebelumnya di Amazon SageMaker AI dan mengimpornya ke Amazon Redshift untuk inferensi lokal. Dengan membawa model Anda sendiri (BYOM), Anda dapat menggunakan model yang dilatih di luar Amazon Redshift dengan Amazon SageMaker AI untuk inferensi dalam database secara lokal di Amazon Redshift. Amazon Redshift ML mendukung penggunaan BYOM baik dalam inferensi lokal maupun jarak jauh.

  Untuk informasi selengkapnya tentang cara menggunakan pernyataan CREATE MODEL untuk inferensi lokal atau jarak jauh, lihatBawa model Anda sendiri (BYOM) - inferensi lokal.

Sebagai pengguna Amazon Redshift ML, Anda dapat memilih salah satu opsi berikut untuk melatih dan menerapkan model Anda:

- Jenis masalah, lihatBUAT MODEL dengan panduan pengguna.
- Tujuan, lihat BUAT MODEL dengan panduan pengguna ataubUAT model XGBoost dengan AUTO OFF.
- Jenis model, lihatBUAT model XGBoost dengan AUTO OFF.
- Preprosesor, lihatBUAT MODEL dengan panduan pengguna.
- Hyperparameter, lihatBUAT model XGBoost dengan AUTO OFF.
- Bawa model Anda sendiri (BYOM), lihatBawa model Anda sendiri (BYOM) - inferensi lokal.

Biaya untuk menggunakan Amazon Redshift ML

Dengan Amazon Redshift, Anda dapat memanfaatkan kemampuan pembelajaran mesin untuk mendapatkan wawasan dari data Anda tanpa memerlukan keahlian rekayasa data atau pembelajaran mesin yang ekstensif. Bagian berikut menjelaskan biaya yang terkait dengan penggunaan Amazon

Redshift ML, membantu Anda merencanakan dan mengoptimalkan pengeluaran Anda sambil memanfaatkan integrasi pembelajaran mesin yang kuat ini.

Biaya untuk menggunakan Amazon Redshift ML dengan AI SageMaker

Amazon Redshift ML untuk SageMaker AI menggunakan sumber daya cluster yang ada untuk prediksi sehingga Anda dapat menghindari biaya Amazon Redshift tambahan. Tidak ada biaya tambahan Amazon Redshift untuk membuat atau menggunakan model. Prediksi terjadi secara lokal di cluster Redshift Anda, jadi Anda tidak perlu membayar ekstra kecuali Anda perlu mengubah ukuran cluster Anda. Amazon Redshift ML menggunakan Amazon SageMaker AI untuk melatih model Anda, yang memang memiliki biaya terkait tambahan.

Tidak ada biaya tambahan untuk fungsi prediksi yang berjalan di dalam cluster Amazon Redshift Anda. Pernyataan CREATE MODEL menggunakan Amazon SageMaker AI dan menimbulkan biaya tambahan. Biaya meningkat dengan jumlah sel dalam data pelatihan Anda. Jumlah sel adalah produk dari jumlah catatan (dalam kueri pelatihan atau waktu tabel) dikalikan jumlah kolom. Misalnya, ketika kueri SELECT dari pernyataan CREATE MODEL membuat 10.000 catatan dan 5 kolom, maka jumlah sel yang dibuatnya adalah 50.000.

Dalam beberapa kasus, data pelatihan yang dihasilkan oleh kueri SELECT dari CREATE MODEL melebihi batas MAX_CELLS yang Anda berikan (atau default 1 juta jika Anda tidak memberikan batas). Dalam kasus ini, CREATE MODEL secara acak memilih kira-kira MAX_CELLS (yaitu catatan “jumlah kolom” dari kumpulan data pelatihan). CREATE MODEL kemudian melakukan pelatihan menggunakan tupel yang dipilih secara acak ini. Pengambilan sampel acak memastikan bahwa kumpulan data pelatihan yang dikurangi tidak memiliki bias apa pun. Dengan demikian, dengan mengatur MAX_CELLS, Anda dapat mengontrol biaya pelatihan Anda.

Saat menggunakan pernyataan CREATE MODEL, Anda dapat menggunakan opsi MAX_CELLS dan MAX_RUNTIME untuk mengontrol biaya, waktu, dan akurasi model potensial.

MAX_RUNTIME menentukan jumlah waktu maksimum yang dapat dilakukan pelatihan dalam SageMaker AI saat opsi AUTO ON atau OFF digunakan. Pekerjaan pelatihan sering selesai lebih cepat dari MAX_RUNTIME, tergantung pada ukuran dataset. Setelah model dilatih, Amazon Redshift melakukan pekerjaan tambahan di latar belakang untuk mengkompilasi dan menginstal model Anda di cluster Anda. Dengan demikian, CREATE MODEL dapat memakan waktu lebih lama dari MAX_RUNTIME untuk diselesaikan. Namun, MAX_RUNTIME membatasi jumlah komputasi dan waktu yang digunakan dalam SageMaker AI untuk melatih model Anda. Anda dapat memeriksa status model Anda kapan saja menggunakan SHOW MODEL.

Saat Anda menjalankan CREATE MODEL dengan AUTO ON, Amazon Redshift ML menggunakan SageMaker AI Autopilot untuk secara otomatis dan cerdas menjelajahi berbagai model (atau kandidat) untuk menemukan yang terbaik. MAX_RUNTIME membatasi jumlah waktu dan perhitungan yang dihabiskan. Jika MAX_RUNTIME diatur terlalu rendah, mungkin tidak ada cukup waktu untuk menjelajahi bahkan satu kandidat. Jika Anda melihat kesalahan “Kandidat autopilot tidak memiliki model,” jalankan kembali CREATE MODEL dengan nilai MAX_RUNTIME yang lebih besar. Untuk informasi selengkapnya tentang parameter ini, lihat MaxAutoMLJobRuntimeInSeconds di Referensi API Amazon SageMaker AI.

Saat Anda menjalankan CREATE MODEL dengan AUTO OFF, MAX_RUNTIME sesuai dengan batas berapa lama pekerjaan pelatihan dijalankan di SageMaker AI. Pekerjaan pelatihan sering selesai lebih cepat, tergantung pada ukuran dataset dan parameter lain yang digunakan, seperti num_rounds di MODEL_TYPE XGBOOST.

Anda juga dapat mengontrol biaya atau mengurangi waktu pelatihan dengan menentukan nilai MAX_CELLS yang lebih kecil saat menjalankan CREATE MODEL. S el adalah entri dalam database. Setiap baris sesuai dengan sel sebanyak kolom, yang dapat memiliki lebar tetap atau bervariasi. MAX_CELLS membatasi jumlah sel, dan dengan demikian jumlah contoh pelatihan yang digunakan untuk melatih model Anda. Secara default, MAX_CELLS diatur ke 1 juta sel. Mengurangi MAX_CELLS mengurangi jumlah baris dari hasil kueri SELECT di CREATE MODEL yang diekspor dan dikirim Amazon Redshift ke SageMaker AI untuk melatih model. Mengurangi MAX_CELLS dengan demikian mengurangi ukuran dataset yang digunakan untuk melatih model baik dengan AUTO ON dan AUTO OFF. Pendekatan ini membantu mengurangi biaya dan waktu untuk melatih model. Untuk melihat informasi tentang pelatihan dan waktu penagihan untuk pekerjaan pelatihan tertentu, pilih Pekerjaan pelatihan di Amazon SageMaker AI.

Meningkatkan MAX_RUNTIME dan MAX_CELLS sering meningkatkan kualitas model dengan memungkinkan SageMaker AI untuk menjelajahi lebih banyak kandidat. Dengan cara ini, SageMaker AI dapat mengambil lebih banyak waktu untuk melatih setiap kandidat dan menggunakan lebih banyak data untuk melatih model yang lebih baik. Jika Anda ingin iterasi atau eksplorasi dataset yang lebih cepat, gunakan MAX_RUNTIME dan MAX_CELLS yang lebih rendah. Jika Anda ingin meningkatkan akurasi model, gunakan MAX_RUNTIME dan MAX_CELLS yang lebih tinggi.

Untuk informasi selengkapnya tentang biaya yang terkait dengan berbagai nomor ponsel dan detail uji coba gratis, lihat harga https://aws.amazon.com/redshift/pricing Amazon Redshift.

Biaya untuk menggunakan Amazon Redshift ML dengan Amazon Bedrock

Menggunakan Amazon Redshift ML dengan Amazon Bedrock menimbulkan biaya tambahan. Untuk informasi selengkapnya, lihat Harga Amazon Bedrock.

Memulai dengan Amazon Redshift ML

Amazon Redshift ML memudahkan pengguna SQL untuk membuat, melatih, dan menerapkan model pembelajaran mesin menggunakan perintah SQL yang sudah dikenal. Dengan Amazon Redshift ML, Anda dapat menggunakan data Anda di cluster Redshift Anda untuk melatih model dengan Amazon SageMaker AI. Kemudian, model dilokalkan dan prediksi dapat dibuat dalam database Amazon Redshift. Amazon Redshift ML saat ini mendukung algoritma pembelajaran mesin: XGBoost (AUTO ON dan OFF) dan perceptron multilayer (AUTO ON), K-Means (AUTO OFF), dan Linear Learner.

Topik

- Pengaturan pengelompokan dan konfigurasikan untuk administrasi Amazon Redshift ML
- Menggunakan penjelasan model dengan Amazon Redshift ML
- Metrik probabilitas Amazon Redshift ML

Pengaturan pengelompokan dan konfigurasikan untuk administrasi Amazon Redshift ML

Sebelum Anda bekerja dengan Amazon Redshift ML, selesaikan penyiapan cluster dan konfigurasikan izin untuk menggunakan Amazon Redshift ML.

Pengaturan cluster untuk menggunakan Amazon Redshift ML

Sebelum Anda bekerja dengan Amazon Redshift ML, lengkapi prasyarat berikut.

Sebagai administrator Amazon Redshift, lakukan pengaturan satu kali berikut untuk menggunakan cluster yang disediakan Amazon Redshift. Untuk menggunakan Amazon Redshift ML dengan Amazon Redshift Serverless, lihat Mem ulai dengan Amazon Redshift Serverless.

Untuk melakukan pengaturan cluster satu kali untuk Amazon Redshift ML

1. Buat cluster Redshift menggunakan Konsol Manajemen AWS atau AWS Command Line Interface (AWS CLI). Pastikan untuk melampirkan kebijakan AWS Identity and Access

Management (IAM) saat membuat cluster. Untuk informasi selengkapnya tentang izin yang diperlukan untuk menggunakan Amazon Redshift ML dengan Amazon SageMaker AI, lihat Izin yang diperlukan untuk menggunakan pembelajaran mesin Amazon Redshift (ML)

2. Buat peran IAM yang diperlukan untuk menggunakan Amazon Redshift ML dengan salah satu cara berikut:

   - Untuk menggunakan SageMaker AI dengan Amazon Redshift ML, buat peran IAM dengan `AmazonS3FullAccess` dan `AmazonSageMakerFullAccess` kebijakan. Jika Anda berencana juga membuat model Prakiraan, lampirkan `AmazonForecastFullAccess` kebijakan ke peran Anda juga.
   - Untuk menggunakan Amazon Bedrock dengan Amazon Redshift ML, buat peran IAM dengan `AmazonS3FullAccess` dan kebijakan. `AmazonBedrockFullAccess`
   - Sebaiknya buat peran IAM melalui konsol Amazon Redshift yang memiliki `AmazonRedshiftAllCommandsFullAccess` kebijakan dengan izin untuk menjalankan perintah SQL, seperti CREATE MODEL. Amazon Redshift menggunakan API-based mekanisme yang mulus untuk membuat peran IAM secara terprogram atas nama Anda Akun AWS . Amazon Redshift secara otomatis melampirkan kebijakan ter AWS kelola yang ada ke peran IAM. Pendekatan ini berarti Anda dapat tetap berada di dalam konsol Amazon Redshift dan tidak perlu beralih ke konsol IAM untuk pembuatan peran. Untuk informasi selengkapnya, lihat Membuat peran IAM sebagai default untuk Amazon Red shift.

     Saat peran IAM dibuat sebagai default untuk cluster Anda, sertakan `redshift` sebagai bagian dari nama sumber daya atau gunakan Redshift-specific tag untuk menandai sumber daya tersebut.

     Untuk menggunakan model pondasi Amazon Bedrock, tambahkan bagian berikut:

```
// Required section if you use Bedrock models.
{
   "Effect": "Allow",
   "Action": "bedrock:InvokeModel",
   "Resource": [
       "arn:aws:bedrock:<region>::foundation-model/*"
   ]
}
```

- Jika Anda ingin membuat peran IAM dengan kebijakan yang lebih ketat, Anda dapat menggunakan kebijakan berikut ini. Anda juga dapat mengubah kebijakan ini untuk memenuhi kebutuhan Anda.

  Bucket Amazon S3 `redshift-downloads/redshift-ml/` adalah lokasi di mana data sampel yang digunakan untuk langkah dan contoh lain disimpan. Anda dapat menghapusnya jika Anda tidak perlu memuat data dari Amazon S3. Atau, ganti dengan bucket Amazon S3 lain yang Anda gunakan untuk memuat data ke Amazon Redshift.

  Nilai*`your-account-id`*,*`your-role`*, dan adalah `amzn-s3-demo-bucket` nilai yang Anda tentukan sebagai bagian dari perintah CREATE MODEL Anda.

  (Opsional) Gunakan bagian AWS KMS kunci dari kebijakan sampel jika Anda menentukan AWS KMS kunci saat menggunakan Amazon Redshift ML. *`your-kms-key`*Nilai adalah kunci yang Anda gunakan sebagai bagian dari perintah CREATE MODEL Anda.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "sagemaker:*Job*",
                "sagemaker:AddTags",
                "sagemaker:CreateModel",
                "sagemaker:CreateEndpoint",
                "sagemaker:CreateEndpointConfig",
                "sagemaker:DeleteEndpoint",
                "sagemaker:DeleteEndpointConfig",
                "sagemaker:DeleteModel"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "s3:AbortMultipartUpload",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/<your-role>",
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::redshift-downloads/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::redshift-downloads"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:Encrypt",
                "kms:GenerateDataKey*"
             ],
             "Resource": [
                "arn:aws:kms:us-east-1:111122223333:key/<your-kms-key>"
             ]
        }
    ]
}
```

3. Untuk mengizinkan Amazon Redshift dan SageMaker AI mengambil peran untuk berinteraksi dengan layanan lain, tambahkan kebijakan kepercayaan berikut ke peran IAM.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "redshift.amazonaws.com",
          "sagemaker.amazonaws.com",
          "forecast.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. (Opsional) Buat bucket Amazon S3 dan AWS KMS kunci. Ini untuk digunakan Amazon Redshift untuk menyimpan data pelatihan yang dikirim ke Amazon SageMaker AI dan menerima model terlatih dari Amazon SageMaker AI.
5. (Opsional) Buat kombinasi yang berbeda dari peran IAM dan bucket Amazon S3 untuk mengontrol akses ke grup pengguna yang berbeda.
6. Saat Anda mengaktifkan perutean VPC yang disempurnakan, lalu lintas antara Redshift ML dan bucket S3 Anda melewati VPC pribadi Anda. Untuk informasi selengkapnya tentang perutean VPC, lihat Perutean VPC yang ditingkatkan di Amazon Redshift.

   Untuk informasi selengkapnya tentang izin yang diperlukan untuk menentukan VPC pribadi untuk pekerjaan penyetelan hyperparameter Anda, lihat Izin yang diperlukan untuk menggunakan Amazon Redshift ML dengan Amazon AI. SageMaker

   > **Note**
   >
   > Panggilan inferensi yang dilakukan ke model SageMaker AI jarak jauh tidak melalui VPC Anda.

Untuk informasi tentang cara menggunakan pernyataan CREATE MODEL untuk mulai membuat model untuk kasus penggunaan yang berbeda, lihatBUAT MODEL.

Mengelola izin dan kepemilikan

Sama seperti objek database lainnya, seperti tabel atau fungsi, Amazon Redshift mengikat pembuatan dan penggunaan model ML untuk mengakses mekanisme kontrol. Ada izin terpisah untuk membuat model yang menjalankan fungsi prediksi.

Contoh berikut menggunakan dua grup pengguna, `retention_analyst_grp` (pembuat model) dan `marketing_analyst_grp` (pengguna model) untuk menggambarkan bagaimana Amazon Redshift mengelola kontrol akses. Analis retensi membuat model pembelajaran mesin yang dapat digunakan oleh kumpulan pengguna lain melalui izin yang diperoleh.

Seorang superuser dapat MEMBERIKAN IZIN PENGGUNA atau GRUP untuk membuat model pembelajaran mesin menggunakan pernyataan berikut.

```
GRANT CREATE MODEL TO GROUP retention_analyst_grp;
```

Pengguna atau grup dengan izin ini dapat membuat model dalam skema apa pun di cluster jika pengguna memiliki izin CREATE biasa pada SKEMA. Model pembelajaran mesin adalah bagian dari hierarki skema dengan cara yang mirip dengan tabel, tampilan, prosedur, dan fungsi yang ditentukan pengguna.

Dengan asumsi skema `demo_ml` sudah ada, berikan dua grup pengguna izin pada skema sebagai berikut.

```
GRANT CREATE, USAGE ON SCHEMA demo_ml TO GROUP retention_analyst_grp;
```

```
GRANT USAGE ON SCHEMA demo_ml TO GROUP marketing_analyst_grp;
```

Untuk mengizinkan pengguna lain menggunakan fungsi inferensi pembelajaran mesin Anda, berikan izin EXECUTE. Contoh berikut menggunakan izin EXECUTE untuk memberikan izin kepada marketing_analist_grp GROUP untuk menggunakan model.

```
GRANT EXECUTE ON MODEL demo_ml.customer_churn_auto_model TO GROUP
 marketing_analyst_grp;
```

Gunakan pernyataan REVOKE dengan CREATE MODEL dan EXECUTE untuk mencabut izin tersebut dari pengguna atau grup. Untuk informasi selengkapnya tentang perintah kontrol izin, lihat HIBAH danMENCABUT.

Menggunakan penjelasan model dengan Amazon Redshift ML

Dengan penjelasan model di Amazon Redshift ML, Anda menggunakan nilai kepentingan fitur untuk membantu memahami bagaimana setiap atribut dalam data pelatihan Anda berkontribusi pada hasil yang diprediksi.

Penjelasan model membantu meningkatkan model pembelajaran mesin (ML) Anda dengan menjelaskan prediksi yang dibuat model Anda. Penjelasan model membantu menjelaskan bagaimana model ini membuat prediksi menggunakan pendekatan atribusi fitur.

Amazon Redshift ML menggabungkan keterangan model untuk menyediakan fungsionalitas penjelasan model kepada pengguna Amazon Redshift ML. Untuk informasi lebih lanjut tentang keterangan model, lihat Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin? dalam Panduan Pengembang Amazon SageMaker AI.

Penjelasan model juga memantau kesimpulan yang dibuat model dalam produksi untuk penyimpangan atribusi fitur. Ini juga menyediakan alat untuk membantu Anda menghasilkan laporan tata kelola model yang dapat Anda gunakan untuk menginformasikan tim risiko dan kepatuhan, dan regulator eksternal.

Saat Anda menentukan opsi AUTO ON atau AUTO OFF saat menggunakan pernyataan CREATE MODEL, setelah pekerjaan pelatihan model selesai, SageMaker AI membuat output penjelasan. Anda dapat menggunakan fungsi EXPLAIN_MODEL untuk menanyakan laporan keterjelasan dalam format JSON. Untuk informasi selengkapnya, lihat Fungsi pembelajaran mesin.

Metrik probabilitas Amazon Redshift ML

Dalam masalah pembelajaran yang diawasi, label kelas adalah hasil prediksi yang menggunakan data input. Misalnya, jika Anda menggunakan model untuk memprediksi apakah pelanggan akan berlangganan kembali ke layanan streaming, kemungkinan label mungkin dan tidak mungkin. Redshift ML menyediakan kemampuan metrik probabilitas, yang menetapkan probabilitas untuk setiap label untuk menunjukkan kemungkinannya. Ini membantu Anda membuat keputusan yang lebih tepat berdasarkan hasil yang diprediksi. Di Amazon Redshift ML, metrik probabilitas tersedia saat membuat model AUTO ON dengan tipe masalah baik klasifikasi biner atau klasifikasi multiclass.

Jika Anda menghilangkan parameter AUTO ON, Redshift ML mengasumsikan bahwa model harus memiliki AUTO ON.

Buat model

Saat membuat model, Amazon Redshift secara otomatis mendeteksi jenis model dan jenis masalah. Jika ini adalah masalah klasifikasi, Redshift secara otomatis membuat fungsi inferensi kedua yang dapat Anda gunakan untuk menampilkan probabilitas relatif terhadap setiap label. Nama fungsi inferensi kedua ini adalah nama fungsi inferensi yang Anda tentukan diikuti oleh string. `_probabilities` Misalnya, jika Anda menamai fungsi inferensi Anda sebagai`customer_churn_predict`, maka nama fungsi inferensi kedua adalah. `customer_churn_predict_probabilities` Anda kemudian dapat menanyakan fungsi ini untuk mendapatkan probabilitas setiap label.

```
CREATE MODEL customer_churn_model
FROM customer_activity
    PROBLEM_TYPE BINARY_CLASSIFICATION
TARGET churn
FUNCTION customer_churn_predict
IAM_ROLE {default}
AUTO ON
SETTINGS ( S3_BUCKET 'amzn-s3-demo-bucket'
```

Dapatkan probabilitas

Setelah fungsi probabilitas siap, menjalankan perintah mengembalikan tipe SUPER yang berisi array probabilitas yang dikembalikan dan label terkait. Misalnya, hasilnya `"probabilities" : [0.7, 0.3], "labels" : ["False.", "True."]` berarti bahwa label False memiliki probabilitas 0,7, dan label True memiliki probabilitas 0,3.

```
SELECT customer_churn_predict_probabilities(Account_length, Area_code,
            VMail_message, Day_mins, Day_calls, Day_charge,Eve_mins, Eve_calls,
            Eve_charge, Night_mins, Night_calls, Night_charge,Intl_mins, Intl_calls,
            Intl_charge, Cust_serv_calls)
FROM customer_activity;

customer_churn_predict_probabilities
 --------------------
 {"probabilities" : [0.7, 0.3], "labels" : ["False.", "True."]}
 {"probabilities" : [0.8, 0.2], "labels" : ["False.", "True."]}
```

```
 {"probabilities" : [0.75, 0.25], "labels" : ["True.", "False"]}
```

Array probabilitas dan label selalu diurutkan berdasarkan probabilitasnya dalam urutan menurun. Anda dapat menulis kueri untuk mengembalikan hanya label yang diprediksi dengan probabilitas tertinggi dengan menghapus sarang hasil yang dikembalikan SUPER dari fungsi probabilitas.

```
SELECT prediction.labels[0], prediction.probabilities[0]
            FROM (SELECT customer_churn_predict_probabilities(Account_length,
 Area_code,
            VMail_message, Day_mins, Day_calls, Day_charge,Eve_mins, Eve_calls,
            Eve_charge, Night_mins, Night_calls, Night_charge,Intl_mins, Intl_calls,
            Intl_charge, Cust_serv_calls) AS prediction
FROM customer_activity);

  labels   | probabilities
-----------+--------------
 "False."  | 0.7
 "False."  | 0.8
 "True."   | 0.75
```

Untuk membuat kueri lebih sederhana, Anda dapat menyimpan hasil fungsi prediksi dalam tabel.

```
CREATE TABLE churn_auto_predict_probabilities AS
            (SELECT customer_churn_predict_probabilities(Account_length, Area_code,
            VMail_message, Day_mins, Day_calls, Day_charge,Eve_mins, Eve_calls,
            Eve_charge, Night_mins, Night_calls, Night_charge,Intl_mins,
            Intl_calls, Intl_charge, Cust_serv_calls) AS prediction
FROM customer_activity);
```

Anda dapat menanyakan tabel dengan hasil untuk mengembalikan hanya prediksi yang memiliki probabilitas lebih tinggi dari 0,7.

```
SELECT prediction.labels[0], prediction.probabilities[0]
FROM churn_auto_predict_probabilities
WHERE prediction.probabilities[0] > 0.7;

  labels   | probabilities
-----------+--------------
 "False."  | 0.8
 "True."   | 0.75
```

Menggunakan notasi indeks, Anda bisa mendapatkan probabilitas label tertentu. Contoh berikut mengembalikan probabilitas semua `True.` label.

```
SELECT label, index, p.prediction.probabilities[index]
FROM churn_auto_predict_probabilities p, p.prediction.labels AS label AT index
WHERE label='True.';

  label  | index | probabilities
---------+-------+---------------
 "True." |     0 | 0.3
 "True." |     0 | 0.2
 "True." |     0 | 0.75
```

Contoh berikut mengembalikan semua baris yang memiliki label`True.` dengan probabilitas lebih besar dari 0,7, menunjukkan bahwa pelanggan kemungkinan besar akan churn.

```
SELECT prediction.labels[0], prediction.probabilities[0]
FROM churn_auto_predict_probabilities
WHERE prediction.probabilities[0] > 0.7 AND prediction.labels[0] = "True.";

labels     | probabilities
-----------+--------------
 "True."   | 0.75
```

Tutorial untuk Amazon Redshift ML

Anda dapat menggunakan Amazon Redshift ML untuk melatih model pembelajaran mesin menggunakan pernyataan SQL, lalu memanggil model dalam kueri SQL untuk prediksi. Pembelajaran mesin di Amazon Redshift melatih model dengan satu perintah SQL. Amazon Redshift secara otomatis meluncurkan pekerjaan pelatihan di Amazon SageMaker AI dan menghasilkan model. Setelah model dibuat, Anda dapat melakukan prediksi di Amazon Redshift menggunakan fungsi prediksi model.

Ikuti langkah-langkah dalam tutorial ini untuk mempelajari fitur Amazon Redshift ML:

- Tutorial: Membangun model churn pelanggan- Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat model churn pelanggan dengan perintah CREATE MODEL, dan menjalankan kueri prediksi untuk skenario pengguna. Kemudian, Anda mengimplementasikan query menggunakan fungsi SQL yang dihasilkan perintah CREATE MODEL.

- Tutorial: Membangun model K-means pengelompokan- Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat, melatih, dan menerapkan model pembelajaran mesin berdasarkan K-means algorit ma.
- Tutorial: Membangun model klasifikasi multi-kelas- Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat model pembelajaran mesin yang memecahkan masalah klasifikasi multi-kelas. Algoritma klasifikasi multi-kelas mengklasifikasikan titik data menjadi salah satu dari tiga kelas atau lebih. Kemudian, Anda mengimplementasikan query menggunakan fungsi SQL yang dihasilkan perintah CREATE MODEL.
- Tutorial: Membangun model XGBoost- Dalam tutorial ini, Anda membuat model dengan data dari Amazon S3 dan menjalankan kueri prediksi dengan model menggunakan Amazon Redshift ML. Algoritma XGBoost adalah implementasi yang dioptimalkan dari algoritma pohon yang didorong gradien.
- Tutorial: Membangun model regresi- Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat model regresi pembelajaran mesin dan menjalankan kueri prediksi pada model. Dengan model regresi, Anda dapat memprediksi hasil numerik, seperti harga rumah, atau berapa banyak orang yang akan menggunakan layanan penyewaan sepeda kota.
- Tutorial: Membangun model regresi dengan pembelajar linier- Dalam tutorial ini, Anda membuat model pembelajar linier dengan data dari Amazon S3 dan menjalankan kueri prediksi dengan model menggunakan Amazon Redshift ML. Algoritma pembelajar linier SageMaker AI memecahkan masalah regresi atau klasifikasi multi-kelas.
- Tutorial: Membangun model klasifikasi multi-kelas dengan pembelajar linier- Dalam tutorial ini, Anda membuat model pembelajar linier dengan data dari Amazon S3, dan kemudian menjalankan kueri prediksi dengan model menggunakan Amazon Redshift ML. Algoritma pembelajar linier SageMaker AI memecahkan masalah regresi atau klasifikasi.

Tutorial: Membangun model churn pelanggan

Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat model churn pelanggan dengan perintah CREATE MODEL, dan menjalankan kueri prediksi untuk skenario pengguna. Kemudian, Anda mengimplementasikan query menggunakan fungsi SQL yang dihasilkan perintah CREATE MODEL.

Anda dapat menggunakan perintah CREATE MODEL sederhana untuk mengekspor data pelatihan, melatih model, mengimpor model, dan menyiapkan fungsi prediksi Amazon Redshift. Gunakan pernyataan CREATE MODEL untuk menentukan data pelatihan baik sebagai tabel atau pernyataan SELECT.

Contoh ini menggunakan informasi historis untuk membangun model pembelajaran mesin dari churn pelanggan operator seluler. Pertama, SageMaker AI melatih model pembelajaran mesin Anda dan kemudian menguji model Anda menggunakan informasi profil pelanggan sewenang-wenang. Setelah model divalidasi, Amazon SageMaker AI menerapkan model dan fungsi prediksi ke Amazon Redshift. Anda dapat menggunakan fungsi prediksi untuk memprediksi apakah pelanggan akan churn atau tidak.

Contoh kasus penggunaan

Anda dapat memecahkan masalah klasifikasi biner lainnya menggunakan Amazon Redshift ML, seperti memprediksi apakah prospek penjualan akan ditutup atau tidak. Anda juga dapat memprediksi apakah transaksi keuangan adalah penipuan atau tidak.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Lakukan prediksi dengan model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus memiliki prasyarat berikut:

- Anda harus menyiapkan cluster Amazon Redshift untuk Amazon Redshift ML. Untuk melakukannya, gunakan dokumentasi untuk Cluster dan konfigurasikan pengaturan untuk administrasi Amazon Redshift ML.
- Cluster Amazon Redshift yang Anda gunakan untuk membuat model, dan bucket Amazon S3 yang Anda gunakan untuk mementaskan data pelatihan dan menyimpan artefak model harus berada di Wilayah yang sama AWS .
- Untuk mengunduh perintah SQL dan dataset sampel yang digunakan dalam dokumentasi ini, lakukan salah satu hal berikut:
  - Unduh perintah SQL, file aktivitas pelanggan, dan file Abalone.
  - Menggunakan AWS CLI untuk Amazon S3, jalankan perintah berikut. Anda dapat menggunakan jalur target Anda sendiri.

```
aws s3 cp s3://redshift-downloads/redshift-ml/tutorial-scripts/redshift-ml-
tutorial.sql </target/path>
aws s3 cp s3://redshift-downloads/redshift-ml/customer_activity/
customer_activity.csv </target/path>
aws s3 cp s3://redshift-downloads/redshift-ml/abalone_xgb/abalone_xgb.csv </target/
path>
```

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

Gunakan editor kueri Amazon Redshift v2 untuk mengedit dan menjalankan kueri serta memvisualisasikan hasil.

Menjalankan kueri berikut membuat tabel bernama `customer_activity` dan mengambil kumpulan data sampel dari Amazon S3.

```
DROP TABLE IF EXISTS customer_activity;

CREATE TABLE customer_activity (
state varchar(2),
account_length int,
area_code int,
phone varchar(8),
intl_plan varchar(3),
vMail_plan varchar(3),
vMail_message int,
day_mins float,
day_calls int,
day_charge float,
total_charge float,
eve_mins float,
eve_calls int,
eve_charge float,
night_mins float,
night_calls int,
night_charge float,
intl_mins float,
intl_calls int,
intl_charge float,
cust_serv_calls int,
churn varchar(6),
record_date date
```

```
);

COPY customer_activity
FROM 's3://redshift-downloads/redshift-ml/customer_activity/'
REGION 'us-east-1' IAM_ROLE default
FORMAT AS CSV IGNOREHEADER 1;
```

Langkah 2: Buat model pembelajaran mesin

Churn adalah input target kami dalam model ini. Semua input lain untuk model adalah atribut yang membantu membuat fungsi untuk memprediksi churn.

Contoh berikut menggunakan operasi CREATE MODEL untuk memberikan model yang memprediksi apakah pelanggan akan aktif, menggunakan input seperti usia pelanggan, kode pos, pengeluaran, dan kasus. Dalam contoh berikut, ganti amzn-s3-demo-bucket dengan bucket Amazon S3 Anda sendiri.

```
CREATE MODEL customer_churn_auto_model
FROM
    (
      SELECT state,
             account_length,
             area_code,
             total_charge/account_length AS average_daily_spend,
             cust_serv_calls/account_length AS average_daily_cases,
             churn
      FROM customer_activity
      WHERE  record_date < '2020-01-01'
     )
TARGET churn FUNCTION ml_fn_customer_churn_auto
IAM_ROLE default SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket'
);
```

Kueri SELECT dalam contoh sebelumnya membuat data pelatihan. Klausa TARGET menentukan kolom mana yang merupakan label pembelajaran mesin yang digunakan operasi CREATE MODEL untuk mempelajari cara memprediksi. Kolom target “churn” menunjukkan apakah pelanggan masih memiliki keanggotaan aktif atau telah menangguhkan keanggotaan. Bidang S3_BUCKET adalah nama bucket Amazon S3 yang Anda buat sebelumnya. Bucket Amazon S3 digunakan untuk berbagi data pelatihan dan artefak antara Amazon Redshift dan Amazon SageMaker AI. Kolom yang tersisa adalah fitur yang digunakan untuk prediksi.

Untuk ringkasan sintaks dan fitur kasus penggunaan dasar perintah CREATE MODEL, lihat Sim ple CREATE MODEL.

Tambahkan izin untuk enkripsi sisi server (opsional)

Amazon Redshift secara default menggunakan Amazon SageMaker AI Autopilot untuk pelatihan. Secara khusus, Amazon Redshift mengekspor data pelatihan dengan aman ke bucket Amazon S3 yang ditentukan pelanggan. Jika Anda tidak menentukan a`KMS_KEY_ID`, maka data dienkripsi menggunakan enkripsi sisi server secara SSE-S3 default.

Saat mengenkripsi input menggunakan enkripsi sisi server dengan kunci ter AWS KMS kelola (SSE-MMS), tambahkan izin berikut:

```
{
    "Effect": "Allow",
    "Action": [
    "kms:Encrypt"
    "kms:Decrypt"
    ]
}
```

Untuk informasi selengkapnya tentang peran Amazon SageMaker AI, lihat peran Amazon SageMaker AI di Panduan Pengembang SageMaker AI Amazon.

Periksa status pelatihan model (opsional)

Anda dapat menggunakan perintah SHOW MODEL untuk mengetahui kapan model Anda siap.

Gunakan operasi berikut untuk memeriksa status model.

```
SHOW MODEL customer_churn_auto_model;
```

Berikut ini adalah contoh output dari operasi sebelumnya.

```
+-------------------------
+-----------------------------------------------------------------------------------------------------------
+
|           Key             |
                    Value
              |
```

```
+--------------------------
+-----------------------------------------------------------------------------------------------------------------
+
|        Model Name        |
        customer_churn_auto_model
             |
|       Schema Name        |
                public
             |
|          Owner           |
                awsuser
             |
|      Creation Time       |
        Tue, 14.06.2022 17:15:52
             |
|       Model State        |
               TRAINING
             |
|                          |

             |
|      TRAINING DATA:      |

             |
|          Query           | SELECT STATE, ACCOUNT_LENGTH, AREA_CODE, TOTAL_CHARGE /
 ACCOUNT_LENGTH AS AVERAGE_DAILY_SPEND, CUST_SERV_CALLS / ACCOUNT_LENGTH AS
 AVERAGE_DAILY_CASES, CHURN |
|                          |
         FROM CUSTOMER_ACTIVITY
             |
|                          |
    WHERE RECORD_DATE < '2020-01-01'
             |
|      Target Column       |
                 CHURN
             |
|                          |

             |
|       PARAMETERS:        |

             |
```

```
|        Model Type        |
                  auto
              |
|       Problem Type       |

              |
|        Objective         |

              |
|     AutoML Job Name      |
     redshiftml-20220614171552640901
              |
|      Function Name       |
        ml_fn_customer_churn_auto
              |
|   Function Parameters    |                                                 state
 account_length area_code average_daily_spend average_daily_cases
                       |
| Function Parameter Types |
      varchar int4 int4 float8 int4
              |
|         IAM Role         |
          default-aws-iam-role
              |
|        S3 Bucket         |
              amzn-s3-demo-bucket
               |
|       Max Runtime        |
                   5400
              |
+--------------------------
+-----------------------------------------------------------------------------------------------------
+
```

Ketika pelatihan model selesai, `model_state` variabel menjadi`Model is Ready`, dan fungsi prediksi menjadi tersedia.

Langkah 3: Lakukan prediksi dengan model

Anda dapat menggunakan pernyataan SQL untuk melihat prediksi yang dibuat oleh model prediksi. Dalam contoh ini, fungsi prediksi yang dibuat oleh operasi CREATE MODEL diberi nama`ml_fn_customer_churn_auto`. Argumen input untuk fungsi prediksi sesuai dengan jenis

fitur, seperti varchar untuk `state` dan integer untuk`account_length`. Output dari fungsi prediksi adalah tipe yang sama dengan kolom TARGET dari pernyataan CREATE MODEL.

1. Anda melatih model pada data dari sebelum 2020-01-01, jadi sekarang Anda menggunakan fungsi prediksi pada set pengujian. Kueri berikut menampilkan prediksi apakah pelanggan yang mendaftar setelah 2020-01-01 akan mengalami churn atau tidak.

```
SELECT
    phone,
    ml_fn_customer_churn_auto(
        state,
        account_length,
        area_code,
        total_charge / account_length,
        cust_serv_calls / account_length
    ) AS active
FROM
    customer_activity
WHERE
    record_date > '2020-01-01';
```

2. Contoh berikut menggunakan fungsi prediksi yang sama untuk kasus penggunaan yang berbeda. Dalam hal ini, Amazon Redshift memprediksi proporsi churner dan non-churner di antara pelanggan dari berbagai negara bagian di mana tanggal rekor lebih besar dari 2020-01-01.

```
WITH predicted AS (
    SELECT
        state,
        ml_fn_customer_churn_auto(
            state,
            account_length,
            area_code,
            total_charge / account_length,
            cust_serv_calls / account_length
        ) :: varchar(6) AS active
    FROM
        customer_activity
    WHERE
        record_date > '2020-01-01'
)
SELECT
    state,
```

```
    SUM(
        CASE
            WHEN active = 'True.' THEN 1
            ELSE 0
        END
    ) AS churners,
    SUM(
        CASE
            WHEN active = 'False.' THEN 1
            ELSE 0
        END
    ) AS nonchurners,
    COUNT(*) AS total_per_state
FROM
    predicted
GROUP BY
    state
ORDER BY
    state;
```

3. Contoh berikut menggunakan fungsi prediksi untuk kasus penggunaan memprediksi persentase pelanggan yang churn dalam suatu negara. Dalam hal ini, Amazon Redshift memprediksi persentase churn di mana tanggal rekor lebih besar dari 2020-01-01.

```
WITH predicted AS (
    SELECT
        state,
        ml_fn_customer_churn_auto(
            state,
            account_length,
            area_code,
            total_charge / account_length,
            cust_serv_calls / account_length
        ) :: varchar(6) AS active
    FROM
        customer_activity
    WHERE
        record_date > '2020-01-01'
)
SELECT
    state,
    CAST((CAST((SUM(
        CASE
```

```
            WHEN active = 'True.' THEN 1
            ELSE 0
        END
    )) AS FLOAT) / CAST(COUNT(*) AS FLOAT)) AS DECIMAL (3, 2)) AS pct_churn,
    COUNT(*) AS total_customers_per_state
FROM
    predicted
GROUP BY
    state
ORDER BY
    3 DESC;
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon RedShiftML
- perintah CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Tutorial: Membangun model K-means pengelompokan

Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat, melatih, dan menerapkan model pembelajaran mesin berdasarkan K-means algorit ma. Algoritma ini memecahkan masalah pengelompokan di mana Anda ingin menemukan pengelompokan dalam data. K-means membantu dalam mengelompokkan data yang belum diberi label. Untuk mempelajari lebih lanjut tentang K-means pengelompokan, lihat Cara K-means Kerja Clustering di Panduan Pengembang SageMaker AI Amazon.

Anda akan menggunakan operasi CREATE MODEL untuk membuat K-means model dari cluster Amazon Redshift. Anda dapat menggunakan perintah CREATE MODEL untuk mengekspor data pelatihan, melatih model, mengimpor model, dan menyiapkan fungsi prediksi Amazon Redshift.

Gunakan operasi CREATE MODEL untuk menentukan data pelatihan baik sebagai tabel atau pernyataan SELECT.

Dalam tutorial ini, Anda menggunakan K-means dataset Global Database of Events, Language, and Tone (GDELT), yang memantau berita dunia di seluruh dunia, dan data disimpan setiap detik setiap hari. K-means akan mengelompokkan acara yang memiliki nada, aktor, atau lokasi yang sama. Data disimpan sebagai beberapa file di Amazon Simple Storage Service, dalam dua folder berbeda. Folder bersejarah, yang mencakup tahun 1979—2013, dan pembaruan harian, yang mencakup tahun 2013 dan setelahnya. Contoh ini menggunakan format historis dan membawa data 1979.

Contoh kasus penggunaan

Anda dapat memecahkan masalah pengelompokan lainnya dengan Amazon Redshift ML, seperti mengelompokkan pelanggan yang memiliki kebiasaan menonton serupa di layanan streaming. Anda juga dapat menggunakan Redshift ML untuk memprediksi jumlah optimal pusat pengiriman untuk layanan pengiriman.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Lakukan prediksi dengan model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus menyelesaikan pengaturan Administratif untuk Amazon Redshift ML.

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

1. Gunakan editor kueri Amazon Redshift v2 untuk menjalankan kueri berikut. Kueri menjatuhkan `gdelt_data` tabel dalam skema publik jika ada dan membuat tabel dengan nama yang sama di skema publik.

```
DROP TABLE IF EXISTS gdelt_data CASCADE;

CREATE TABLE gdelt_data (
    GlobalEventId bigint,
```

```
    SqlDate bigint,
    MonthYear bigint,
    Year bigint,
    FractionDate double precision,
    Actor1Code varchar(256),
    Actor1Name varchar(256),
    Actor1CountryCode varchar(256),
    Actor1KnownGroupCode varchar(256),
    Actor1EthnicCode varchar(256),
    Actor1Religion1Code varchar(256),
    Actor1Religion2Code varchar(256),
    Actor1Type1Code varchar(256),
    Actor1Type2Code varchar(256),
    Actor1Type3Code varchar(256),
    Actor2Code varchar(256),
    Actor2Name varchar(256),
    Actor2CountryCode varchar(256),
    Actor2KnownGroupCode varchar(256),
    Actor2EthnicCode varchar(256),
    Actor2Religion1Code varchar(256),
    Actor2Religion2Code varchar(256),
    Actor2Type1Code varchar(256),
    Actor2Type2Code varchar(256),
    Actor2Type3Code varchar(256),
    IsRootEvent bigint,
    EventCode bigint,
    EventBaseCode bigint,
    EventRootCode bigint,
    QuadClass bigint,
    GoldsteinScale double precision,
    NumMentions bigint,
    NumSources bigint,
    NumArticles bigint,
    AvgTone double precision,
    Actor1Geo_Type bigint,
    Actor1Geo_FullName varchar(256),
    Actor1Geo_CountryCode varchar(256),
    Actor1Geo_ADM1Code varchar(256),
    Actor1Geo_Lat double precision,
    Actor1Geo_Long double precision,
    Actor1Geo_FeatureID bigint,
    Actor2Geo_Type bigint,
    Actor2Geo_FullName varchar(256),
    Actor2Geo_CountryCode varchar(256),
```

```
    Actor2Geo_ADM1Code varchar(256),
    Actor2Geo_Lat double precision,
    Actor2Geo_Long double precision,
    Actor2Geo_FeatureID bigint,
    ActionGeo_Type bigint,
    ActionGeo_FullName varchar(256),
    ActionGeo_CountryCode varchar(256),
    ActionGeo_ADM1Code varchar(256),
    ActionGeo_Lat double precision,
    ActionGeo_Long double precision,
    ActionGeo_FeatureID bigint,
    DATEADDED bigint
);
```

2. Kueri berikut memuat data sampel ke dalam `gdelt_data` tabel.

```
COPY gdelt_data
FROM 's3://gdelt-open-data/events/1979.csv'
REGION 'us-east-1'
IAM_ROLE default
CSV
DELIMITER '\t';
```

Periksa data pelatihan (opsional)

Untuk melihat data apa model Anda akan dilatih, gunakan kueri berikut.

```
SELECT
    AvgTone,
    EventCode,
    NumArticles,
    Actor1Geo_Lat,
    Actor1Geo_Long,
    Actor2Geo_Lat,
    Actor2Geo_Long
FROM
    gdelt_data LIMIT 100;
```

Langkah 2: Buat model pembelajaran mesin

Contoh berikut menggunakan perintah CREATE MODEL untuk membuat model yang mengelompokkan data menjadi tujuh cluster. Nilai K adalah jumlah cluster yang dibagi menjadi

titik data Anda. Model mengklasifikasikan titik data Anda ke dalam cluster di mana titik data lebih mirip satu sama lain. Dengan mengelompokkan titik data ke dalam kelompok, K-Means algoritma secara berulang menentukan pusat cluster terbaik. Algoritma kemudian menetapkan setiap titik data ke pusat cluster terdekat. Anggota terdekat pusat cluster yang sama termasuk dalam kelompok yang sama. Anggota kelompok semirip mungkin dengan anggota lain dalam kelompok yang sama, dan seberbeda mungkin dari anggota kelompok lain. Nilai K bersifat subjektif dan tergantung pada metode yang mengukur kesamaan di antara titik data. Anda dapat mengubah nilai K untuk memperlancar ukuran cluster jika cluster tidak merata.

Dalam contoh berikut, ganti amzn-s3-demo-bucket dengan bucket Amazon S3 Anda sendiri.

```
CREATE MODEL news_data_clusters
FROM
    (
        SELECT
            AvgTone,
            EventCode,
            NumArticles,
            Actor1Geo_Lat,
            Actor1Geo_Long,
            Actor2Geo_Lat,
            Actor2Geo_Long
        FROM
            gdelt_data
    ) FUNCTION news_monitoring_cluster
    IAM_ROLE default
    AUTO OFF
    MODEL_TYPE KMEANS
    PREPROCESSORS 'none'
    HYPERPARAMETERS DEFAULT
    EXCEPT
    (K '7')
    SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');
```

Periksa status pelatihan model (opsional)

Anda dapat menggunakan perintah SHOW MODEL untuk mengetahui kapan model Anda siap.

Untuk memeriksa status model, gunakan operasi SHOW MODEL berikut dan cari `Model State` apakah ya`Ready`.

```
SHOW MODEL NEWS_DATA_CLUSTERS;
```

Ketika model siap, output dari operasi sebelumnya harus menunjukkan bahwa itu `Model State` adalah`Ready`. Berikut ini adalah contoh output dari operasi SHOW MODEL.

```
+--------------------------
+-----------------------------------------------------------------------------------------------
+
|        Model Name        |
 news_data_clusters                                    |
+--------------------------
+-----------------------------------------------------------------------------------------------
+
|       Schema Name        |                                                 public
                                        |
|          Owner           |                                                awsuser
                                        |
|      Creation Time       |                                      Fri, 17.06.2022
 16:32:19                                    |
|       Model State        |                                                 READY
                                        |
|        train:msd         |                                              2973.822754
                                        |
|      train:progress      |                                               100.000000
                                        |
|     train:throughput     |                                             237114.875000
                                        |
|      Estimated Cost      |                                                0.004983
                                        |
|                          |
                                        |
|      TRAINING DATA:      |
                                        |
|          Query           | SELECT AVGTONE, EVENTCODE, NUMARTICLES, ACTOR1GEO_LAT,
 ACTOR1GEO_LONG, ACTOR2GEO_LAT, ACTOR2GEO_LONG |
|                          |                                                FROM GDELT_DATA
                                        |
|                          |
                                        |
|       PARAMETERS:        |
                                        |
|        Model Type        |                                                 kmeans
                                        |
```

```
|    Training Job Name     |
 redshiftml-20220617163219978978-kmeans                                  |
|      Function Name       |
 news_monitoring_cluster                                          |
|   Function Parameters    |       avgtone eventcode numarticles actor1geo_lat
 actor1geo_long actor2geo_lat actor2geo_long        |
| Function Parameter Types |                              float8 int8 int8 float8 float8
 float8 float8                             |
|         IAM Role         |                                        default-aws-iam-
role                                        |
|        S3 Bucket         |                                           amzn-s3-demo-
bucket                                      |
|       Max Runtime        |                                                5400
                                          |
|                          |
                                          |
|     HYPERPARAMETERS:     |
                                          |
|       feature_dim        |                                                 7
                                          |
|            k             |                                                 7
                                          |
+--------------------------
+-----------------------------------------------------------------------------------------------
+
```

Langkah 3: Lakukan prediksi dengan model

Identifikasi cluster

Anda dapat menemukan pengelompokan diskrit yang diidentifikasi dalam data oleh model Anda, atau dikenal sebagai cluster. Cluster adalah kumpulan titik data yang lebih dekat ke pusat klaster daripada pusat cluster lainnya. Karena nilai K mewakili jumlah cluster dalam model, itu juga mewakili jumlah pusat cluster. Kueri berikut mengidentifikasi cluster dengan menunjukkan cluster yang terkait dengan masing-masing`globaleventid`.

```
SELECT
    globaleventid,
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
```

```
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS cluster
FROM
    gdelt_data;
```

Periksa distribusi data

Anda dapat memeriksa distribusi data di seluruh cluster untuk melihat apakah nilai K yang Anda pilih menyebabkan data agak merata. Gunakan kueri berikut untuk menentukan apakah data didistribusikan secara merata di seluruh cluster Anda.

```
SELECT
    events_cluster,
    COUNT(*) AS nbr_events
FROM
    (
        SELECT
            globaleventid,
            news_monitoring_cluster(
                AvgTone,
                EventCode,
                NumArticles,
                Actor1Geo_Lat,
                Actor1Geo_Long,
                Actor2Geo_Lat,
                Actor2Geo_Long
            ) AS events_cluster
        FROM
            gdelt_data
    )
GROUP BY
    1;
```

Perhatikan bahwa Anda dapat mengubah nilai K untuk menghaluskan ukuran cluster jika cluster tidak merata.

Tentukan pusat cluster

Titik data lebih dekat ke pusat klusternya daripada ke pusat cluster lainnya. Dengan demikian, menemukan pusat cluster membantu Anda mendefinisikan cluster.

Jalankan query berikut untuk menentukan pusat cluster berdasarkan jumlah artikel berdasarkan kode acara.

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    eventcode,
    SUM(numArticles) AS numArticles
FROM
    gdelt_data
GROUP BY
    1,
    2;
```

Menampilkan informasi tentang titik data dalam cluster

Gunakan query berikut untuk mengembalikan data untuk titik-titik yang ditetapkan ke cluster kelima. Artikel yang dipilih harus memiliki dua aktor.

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    eventcode,
    actor1name,
    actor2name,
    SUM(numarticles) AS totalarticles
FROM
    gdelt_data
WHERE
```

```
    events_cluster = 5
    AND actor1name <> ' '
    AND actor2name <> ' '
GROUP BY
    1,
    2,
    3,
    4
ORDER BY
    5 desc;
```

Tampilkan data tentang peristiwa dengan aktor dari kode etnis yang sama

Kueri berikut menghitung jumlah artikel yang ditulis tentang peristiwa dengan nada positif. Kueri juga mengharuskan kedua aktor memiliki kode etnis yang sama dan mengembalikan cluster mana yang ditugaskan setiap peristiwa.

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    SUM(numarticles) AS total_articles,
    eventcode AS event_code,
    Actor1EthnicCode AS ethnic_code
FROM
    gdelt_data
WHERE
    Actor1EthnicCode = Actor2EthnicCode
    AND Actor1EthnicCode <> ' '
    AND Actor2EthnicCode <> ' '
    AND AvgTone > 0
GROUP BY
    1,
    3,
    4
HAVING
    (total_articles) > 4
```

```
ORDER BY
    1,
    2 ASC;
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon Redshift ML
- Operasi CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Tutorial: Membangun model klasifikasi multi-kelas

Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat model pembelajaran mesin yang memecahkan masalah klasifikasi multi-kelas. Algoritma klasifikasi multi-kelas mengklasifikasikan titik data menjadi salah satu dari tiga kelas atau lebih. Kemudian, Anda mengimplementasikan query menggunakan fungsi SQL yang dihasilkan perintah CREATE MODEL.

Anda dapat menggunakan perintah CREATE MODEL untuk mengekspor data pelatihan, melatih model, mengimpor model, dan menyiapkan fungsi prediksi Amazon Redshift. Gunakan operasi CREATE MODEL untuk menentukan data pelatihan baik sebagai tabel atau pernyataan SELECT.

Untuk mengikuti tutorial ini, Anda menggunakan kumpulan data publik Prakir E-Commerce aan Pen jualan, yang mencakup data penjualan pengecer online Inggris. Model yang Anda hasilkan akan menargetkan pelanggan paling aktif untuk program loyalitas pelanggan khusus. Dengan klasifikasi multi-kelas, Anda dapat menggunakan model untuk memprediksi berapa bulan pelanggan akan aktif selama periode 13 bulan. Fungsi prediksi menunjuk pelanggan yang diprediksi aktif selama 7 bulan atau lebih untuk masuk ke program.

Contoh kasus penggunaan

Anda dapat memecahkan masalah klasifikasi multi-kelas lainnya dengan Amazon Redshift ML, seperti memprediksi produk terlaris dari lini produk. Anda juga dapat memprediksi buah mana yang berisi gambar, seperti memilih apel atau pir atau jeruk.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Lakukan prediksi dengan model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus menyelesaikan pengaturan Administratif untuk Amazon Redshift ML.

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

Gunakan editor kueri Amazon Redshift v2 untuk menjalankan kueri berikut. Kueri ini memuat data sampel ke Amazon Redshift.

1. Query berikut membuat tabel bernama`ecommerce_sales`.

```
CREATE TABLE IF NOT EXISTS ecommerce_sales (
    invoiceno VARCHAR(30),
    stockcode VARCHAR(30),
    description VARCHAR(60),
    quantity DOUBLE PRECISION,
    invoicedate VARCHAR(30),
    unitprice DOUBLE PRECISION,
    customerid BIGINT,
    country VARCHAR(25)
);
```

2. Kueri berikut menyalin data sampel dari dataset Prak E-Commerce iraan Penjualan ke dalam `ecommerce_sales` tabel.

```
COPY ecommerce_sales
```

```
FROM
    's3://redshift-ml-multiclass/ecommerce_data.txt'
IAM_ROLE default
DELIMITER '\t'
IGNOREHEADER 1
REGION 'us-east-1'
MAXERROR 100;
```

Pisahkan data

Saat Anda membuat model di Amazon Redshift ML, SageMaker AI secara otomatis membagi data Anda menjadi set pelatihan dan pengujian, sehingga SageMaker AI dapat menentukan akurasi model. Dengan memisahkan data secara manual pada langkah ini, Anda akan dapat memverifikasi keakuratan model dengan mengalokasikan set prediksi tambahan.

Gunakan pernyataan SQL berikut untuk membagi data menjadi tiga set untuk pelatihan, validasi, dan prediksi.

```
--creates table with all data
CREATE TABLE ecommerce_sales_data AS (
    SELECT
        t1.stockcode,
        t1.description,
        t1.invoicedate,
        t1.customerid,
        t1.country,
        t1.sales_amt,
        CAST(RANDOM() * 100 AS INT) AS data_group_id
    FROM
        (
            SELECT
                stockcode,
                description,
                invoicedate,
                customerid,
                country,
                SUM(quantity * unitprice) AS sales_amt
            FROM
                ecommerce_sales
            GROUP BY
                1,
                2,
```

```
                3,
                4,
                5
        ) t1
);

--creates training set
CREATE TABLE ecommerce_sales_training AS (
    SELECT
        a.customerid,
        a.country,
        a.stockcode,
        a.description,
        a.invoicedate,
        a.sales_amt,
        (b.nbr_months_active) AS nbr_months_active
    FROM
        ecommerce_sales_data a
        INNER JOIN (
            SELECT
                customerid,
                COUNT(
                    DISTINCT(
                        DATE_PART(y, CAST(invoicedate AS DATE)) || '-' || LPAD(
                            DATE_PART(mon, CAST(invoicedate AS DATE)),
                            2,
                            '00'
                        )
                    )
                ) AS nbr_months_active
            FROM
                ecommerce_sales_data
            GROUP BY
                1
        ) b ON a.customerid = b.customerid
    WHERE
        a.data_group_id < 80
);

--creates validation set
CREATE TABLE ecommerce_sales_validation AS (
    SELECT
        a.customerid,
        a.country,
```

```
        a.stockcode,
        a.description,
        a.invoicedate,
        a.sales_amt,
        (b.nbr_months_active) AS nbr_months_active
    FROM
        ecommerce_sales_data a
        INNER JOIN (
            SELECT
                customerid,
                COUNT(
                    DISTINCT(
                        DATE_PART(y, CAST(invoicedate AS DATE)) || '-' || LPAD(
                            DATE_PART(mon, CAST(invoicedate AS DATE)),
                            2,
                            '00'
                        )
                    )
                ) AS nbr_months_active
            FROM
                ecommerce_sales_data
            GROUP BY
                1
        ) b ON a.customerid = b.customerid
    WHERE
        a.data_group_id BETWEEN 80
        AND 90
);

--creates prediction set
CREATE TABLE ecommerce_sales_prediction AS (
    SELECT
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    FROM
        ecommerce_sales_data
    WHERE
        data_group_id > 90);
```

Langkah 2: Buat model pembelajaran mesin

Pada langkah ini, Anda menggunakan pernyataan CREATE MODEL untuk membuat model pembelajaran mesin Anda menggunakan klasifikasi multi-kelas.

Kueri berikut membuat model klasifikasi multi-kelas dengan set pelatihan menggunakan operasi CREATE MODEL. Ganti amzn-s3-demo-bucket dengan bucket Amazon S3 Anda sendiri.

```
CREATE MODEL ecommerce_customer_activity
FROM
    (
        SELECT
            customerid,
            country,
            stockcode,
            description,
            invoicedate,
            sales_amt,
            nbr_months_active
        FROM
            ecommerce_sales_training
    ) TARGET nbr_months_active FUNCTION predict_customer_activity IAM_ROLE default
 PROBLEM_TYPE MULTICLASS_CLASSIFICATION SETTINGS (
        S3_BUCKET 'amzn-s3-demo-bucket',
        S3_GARBAGE_COLLECT OFF
    );
```

Dalam kueri ini, Anda menentukan jenis masalah sebagai`Multiclass_Classification`. Target yang Anda prediksi untuk model adalah`nbr_months_active`. Ketika SageMaker AI selesai melatih model, ia menciptakan fungsi`predict_customer_activity`, yang akan Anda gunakan untuk membuat prediksi di Amazon Redshift.

Tampilkan status pelatihan model (opsional)

Anda dapat menggunakan perintah SHOW MODEL untuk mengetahui kapan model Anda siap.

Gunakan kueri berikut untuk mengembalikan berbagai metrik model, termasuk status model dan akurasi.

```
SHOW MODEL ecommerce_customer_activity;
```

Ketika model siap, output dari operasi sebelumnya harus menunjukkan bahwa itu `Model State` adalah`Ready`. Berikut ini adalah contoh output dari operasi SHOW MODEL.

```
+--------------------------
+-----------------------------------------------------------------------------------------------
+
|        Model Name        |
 ecommerce_customer_activity                                |
+--------------------------
+-----------------------------------------------------------------------------------------------
+
|       Schema Name        |                                                 public
                                    |
|          Owner           |                                                 awsuser
                                    |
|      Creation Time       |                                  Fri, 17.06.2022 19:02:15
                                    |
|       Model State        |                                                  READY
                                    |
|   Training Job Status    |
 MaxAutoMLJobRuntimeReached                                 |
|   validation:accuracy    |                                           0.991280
                                    |
|      Estimated Cost      |                                           7.897689
                                    |
|                          |
                                    |
|      TRAINING DATA:      |
                                    |
|          Query           | SELECT CUSTOMERID, COUNTRY, STOCKCODE, DESCRIPTION,
 INVOICEDATE, SALES_AMT, NBR_MONTHS_ACTIVE |
|                          |                                    FROM
 ECOMMERCE_SALES_TRAINING                                  |
|      Target Column       |                                         NBR_MONTHS_ACTIVE
                                    |
|                          |
                                    |
|       PARAMETERS:        |
                                    |
|        Model Type        |                                             xgboost
                                    |
|       Problem Type       |                                  MulticlassClassification
                                    |
```

```
|        Objective         |                                        Accuracy
                                  |
|     AutoML Job Name      |
 redshiftml-20220617190215268770                              |
|      Function Name       |
 predict_customer_activity                                    |
|   Function Parameters    |               customerid country stockcode description
 invoicedate sales_amt               |
| Function Parameter Types |                          int8 varchar varchar varchar
 varchar float8                        |
|         IAM Role         |                                    default-aws-iam-role
                                  |
|        S3 Bucket         |                                        amzn-s3-demo-
bucket                                    |
|       Max Runtime        |                                           5400
                                  |
+--------------------------
+-----------------------------------------------------------------------------------------------
+
```

Langkah 3: Lakukan prediksi dengan model

Kueri berikut menunjukkan pelanggan mana yang memenuhi syarat untuk program loyalitas pelanggan Anda. Jika model memprediksi bahwa pelanggan akan aktif setidaknya selama tujuh bulan, maka model memilih pelanggan untuk program loyalitas.

```
SELECT
    customerid,
    predict_customer_activity(
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    ) AS predicted_months_active
FROM
    ecommerce_sales_prediction
WHERE
    predicted_months_active >= 7
GROUP BY
    1,
    2
```

```
LIMIT
    10;
```

Jalankan kueri prediksi terhadap data validasi (opsional)

Jalankan kueri prediksi berikut terhadap data validasi untuk melihat tingkat akurasi model.

```
SELECT
    CAST(SUM(t1.match) AS decimal(7, 2)) AS predicted_matches,
    CAST(SUM(t1.nonmatch) AS decimal(7, 2)) AS predicted_non_matches,
    CAST(SUM(t1.match + t1.nonmatch) AS decimal(7, 2)) AS total_predictions,
    predicted_matches / total_predictions AS pct_accuracy
FROM
    (
        SELECT
            customerid,
            country,
            stockcode,
            description,
            invoicedate,
            sales_amt,
            nbr_months_active,
            predict_customer_activity(
                customerid,
                country,
                stockcode,
                description,
                invoicedate,
                sales_amt
            ) AS predicted_months_active,
            CASE
                WHEN nbr_months_active = predicted_months_active THEN 1
                ELSE 0
            END AS match,
            CASE
                WHEN nbr_months_active <> predicted_months_active THEN 1
                ELSE 0
            END AS nonmatch
        FROM
            ecommerce_sales_validation
    )t1;
```

Memprediksi berapa banyak pelanggan yang melewatkan entri (opsional)

Kueri berikut membandingkan jumlah pelanggan yang diprediksi aktif hanya selama 5 atau 6 bulan. Model ini memprediksi bahwa pelanggan ini akan kehilangan program loyalitas. Permintaan kemudian membandingkan jumlah yang hampir tidak ketinggalan program dengan jumlah yang diperkirakan memenuhi syarat untuk program loyalitas. Kueri ini dapat digunakan untuk menginformasikan keputusan apakah akan menurunkan ambang batas untuk program loyalitas. Anda juga dapat menentukan apakah ada sejumlah besar pelanggan yang diprediksi hampir tidak ketinggalan program. Anda kemudian dapat mendorong pelanggan tersebut untuk meningkatkan aktivitas mereka untuk mendapatkan keanggotaan program loyalitas.

```
SELECT
    predict_customer_activity(
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    ) AS predicted_months_active,
    COUNT(customerid)
FROM
    ecommerce_sales_prediction
WHERE
    predicted_months_active BETWEEN 5 AND 6
GROUP BY
    1
ORDER BY
    1 ASC
LIMIT
    10)
UNION
(SELECT
      NULL AS predicted_months_active,
    COUNT (customerid)
FROM
    ecommerce_sales_prediction
WHERE
    predict_customer_activity(
        customerid,
        country,
        stockcode,
```

```
        description,
        invoicedate,
        sales_amt
    ) >=7);
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon Redshift ML
- Operasi CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Tutorial: Membangun model XGBoost

Dalam tutorial ini, Anda membuat model dengan data dari Amazon S3 dan menjalankan kueri prediksi dengan model menggunakan Amazon Redshift ML. Algoritma XGBoost adalah implementasi yang dioptimalkan dari algoritma pohon yang didorong gradien. XGBoost menangani lebih banyak tipe data, hubungan, dan distribusi daripada algoritma pohon yang didorong gradien lainnya. Anda dapat menggunakan XGBoost untuk regresi, klasifikasi biner, klasifikasi multi-kelas, dan masalah peringkat. Untuk informasi selengkapnya tentang algoritma XGBoost, lihat algoritma XGBoost di Panduan Pengembang AI Amazon SageMaker .

`CREATE MODEL`Operasi Amazon Redshift ML dengan `AUTO OFF` opsi saat ini mendukung XGBoost sebagai. `MODEL_TYPE` Anda dapat memberikan informasi yang relevan seperti tujuan dan hyperparameter sebagai bagian dari `CREATE MODEL` perintah, berdasarkan kasus penggunaan Anda.

Dalam tutorial ini, Anda menggunakan kumpulan data oten tikasi uang kertas, yang merupakan masalah klasifikasi biner untuk memprediksi apakah uang kertas tertentu asli atau palsu.

Contoh kasus penggunaan

Anda dapat memecahkan masalah klasifikasi biner lainnya menggunakan Amazon Redshift ML, seperti memprediksi apakah pasien sehat atau memiliki penyakit. Anda juga dapat memprediksi apakah email itu spam atau bukan spam.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Lakukan prediksi dengan model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus menyelesaikan pengaturan Administratif untuk Amazon Redshift ML.

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

Gunakan editor kueri Amazon Redshift v2 untuk menjalankan kueri berikut.

Kueri berikut membuat dua tabel, memuat data dari Amazon S3, dan membagi data menjadi set pelatihan dan set pengujian. Anda akan menggunakan set pelatihan untuk melatih model Anda dan membuat fungsi prediksi. Kemudian, Anda akan menguji fungsi prediksi pada set pengujian.

```
--create training set table
CREATE TABLE banknoteauthentication_train(
    variance FLOAT,
    skewness FLOAT,
    curtosis FLOAT,
    entropy FLOAT,
    class INT
);

--Load into training table
COPY banknoteauthentication_train
FROM
    's3://redshiftbucket-ml-sagemaker/banknote_authentication/train_data/' IAM_ROLE
 default REGION 'us-west-2' IGNOREHEADER 1 CSV;
```

```
--create testing set table
CREATE TABLE banknoteauthentication_test(
    variance FLOAT,
    skewness FLOAT,
    curtosis FLOAT,
    entropy FLOAT,
    class INT
);

--Load data into testing table
COPY banknoteauthentication_test
FROM
    's3://redshiftbucket-ml-sagemaker/banknote_authentication/test_data/'
    IAM_ROLE default
    REGION 'us-west-2'
    IGNOREHEADER 1
    CSV;
```

Langkah 2: Buat model pembelajaran mesin

Kueri berikut membuat model XGBoost di Amazon Redshift ML dari set pelatihan yang Anda buat di langkah sebelumnya. Ganti `amzn-s3-demo-bucket` dengan milik Anda sendiri`S3_BUCKET`, yang akan menyimpan kumpulan data input Anda dan artefak Redshift ML lainnya.

```
CREATE MODEL model_banknoteauthentication_xgboost_binary
FROM
    banknoteauthentication_train
    TARGET class
    FUNCTION func_model_banknoteauthentication_xgboost_binary
    IAM_ROLE default
    AUTO OFF
    MODEL_TYPE xgboost
    OBJECTIVE 'binary:logistic'
    PREPROCESSORS 'none'
    HYPERPARAMETERS DEFAULT
EXCEPT(NUM_ROUND '100')
SETTINGS(S3_BUCKET 'amzn-s3-demo-bucket');
```

Tampilkan status pelatihan model (opsional)

Anda dapat menggunakan perintah SHOW MODEL untuk mengetahui kapan model Anda siap.

Gunakan kueri berikut untuk memantau kemajuan pelatihan model.

```
SHOW MODEL model_banknoteauthentication_xgboost_binary;
```

Jika modelnyaREADY, operasi SHOW MODEL juga menyediakan `train:error` metrik, seperti yang ditunjukkan pada contoh output berikut. Met `train:error` rik adalah ukuran akurasi model Anda yang mengukur hingga enam tempat desimal. Nilai 0 adalah yang paling akurat dan nilai 1 paling tidak akurat.

```
+--------------------------+-------------------------------------------------+
|        Model Name        |   model_banknoteauthentication_xgboost_binary   |
+--------------------------+-------------------------------------------------+
| Schema Name              | public                                          |
| Owner                    | awsuser                                         |
| Creation Time            | Tue, 21.06.2022 19:07:35                        |
| Model State              | READY                                           |
| train:error              |                                        0.000000 |
| Estimated Cost           |                                        0.006197 |
|                          |                                                 |
| TRAINING DATA:           |                                                 |
| Query                    | SELECT *                                        |
|                          | FROM "BANKNOTEAUTHENTICATION_TRAIN"             |
| Target Column            | CLASS                                           |
|                          |                                                 |
| PARAMETERS:              |                                                 |
| Model Type               | xgboost                                         |
| Training Job Name        | redshiftml-20220621190735686935-xgboost         |
| Function Name            | func_model_banknoteauthentication_xgboost_binary |
| Function Parameters      | variance skewness curtosis entropy              |
| Function Parameter Types | float8 float8 float8 float8                     |
| IAM Role                 | default-aws-iam-role                            |
| S3 Bucket                | amzn-s3-demo-bucket                             |
| Max Runtime              |                                            5400 |
|                          |                                                 |
| HYPERPARAMETERS:         |                                                 |
| num_round                |                                             100 |
| objective                | binary:logistic                                 |
+--------------------------+-------------------------------------------------+
```

Langkah 3: Lakukan prediksi dengan model

Periksa keakuratan model

Kueri prediksi berikut menggunakan fungsi prediksi yang dibuat pada langkah sebelumnya untuk memeriksa keakuratan model Anda. Jalankan kueri ini pada set pengujian untuk memastikan model tidak sesuai terlalu dekat dengan set pelatihan. Korespondensi yang erat ini juga dikenal sebagai overfitting, dan overfitting dapat menyebabkan model membuat prediksi yang tidak dapat diandalkan.

```
WITH predict_data AS (
    SELECT
        class AS label,
        func_model_banknoteauthentication_xgboost_binary (variance, skewness, curtosis,
 entropy) AS predicted,
        CASE
            WHEN label IS NULL THEN 0
            ELSE label
        END AS actual,
        CASE
            WHEN actual = predicted THEN 1 :: INT
            ELSE 0 :: INT
        END AS correct
    FROM
        banknoteauthentication_test
),
aggr_data AS (
    SELECT
        SUM(correct) AS num_correct,
        COUNT(*) AS total
    FROM
        predict_data
)
SELECT
    (num_correct :: FLOAT / total :: FLOAT) AS accuracy
FROM
    aggr_data;
```

Memprediksi jumlah uang kertas asli dan palsu

Kueri prediksi berikut mengembalikan jumlah perkiraan uang kertas asli dan palsu dalam set pengujian.

```
WITH predict_data AS (
```

```
    SELECT
        func_model_banknoteauthentication_xgboost_binary(variance, skewness, curtosis,
 entropy) AS predicted
    FROM
        banknoteauthentication_test
)
SELECT
    CASE
        WHEN predicted = '0' THEN 'Original banknote'
        WHEN predicted = '1' THEN 'Counterfeit banknote'
        ELSE 'NA'
    END AS banknote_authentication,
    COUNT(1) AS count
FROM
    predict_data
GROUP BY
    1;
```

Temukan pengamatan rata-rata untuk uang kertas asli dan palsu

Kueri prediksi berikut mengembalikan nilai rata-rata setiap fitur untuk uang kertas yang diprediksi asli dan palsu dalam set pengujian.

```
WITH predict_data AS (
    SELECT
        func_model_banknoteauthentication_xgboost_binary(variance, skewness, curtosis,
 entropy) AS predicted,
          variance,
          skewness,
          curtosis,
          entropy
    FROM
        banknoteauthentication_test
)
SELECT
    CASE
        WHEN predicted = '0' THEN 'Original banknote'
        WHEN predicted = '1' THEN 'Counterfeit banknote'
        ELSE 'NA'
    END AS banknote_authentication,
    TRUNC(AVG(variance), 2) AS avg_variance,
    TRUNC(AVG(skewness), 2) AS avg_skewness,
    TRUNC(AVG(curtosis), 2) AS avg_curtosis,
```

```
    TRUNC(AVG(entropy), 2) AS avg_entropy
FROM
    predict_data
GROUP BY
    1
ORDER BY
    2;
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon Redshift ML
- Operasi CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Tutorial: Membangun model regresi

Dalam tutorial ini, Anda menggunakan Amazon Redshift ML untuk membuat model regresi pembelajaran mesin dan menjalankan kueri prediksi pada model. Dengan model regresi, Anda dapat memprediksi hasil numerik, seperti harga rumah, atau berapa banyak orang yang akan menggunakan layanan penyewaan sepeda kota. Anda menggunakan perintah CREATE MODEL di Amazon Redshift dengan data pelatihan Anda. Kemudian, Amazon Redshift ML mengkompilasi model, mengimpor model terlatih ke Redshift, dan menyiapkan fungsi prediksi SQL. Anda dapat menggunakan fungsi prediksi dalam kueri SQL di Amazon Redshift.

Dalam tutorial ini, Anda akan menggunakan Amazon Redshift ML untuk membangun model regresi yang memprediksi jumlah orang yang menggunakan layanan berbagi sepeda kota Toronto pada jam tertentu dalam sehari. Input untuk model termasuk hari libur dan kondisi cuaca. Anda akan menggunakan model regresi, karena Anda menginginkan hasil numerik untuk masalah ini.

Anda dapat menggunakan perintah CREATE MODEL untuk mengekspor data pelatihan, melatih model, dan membuat model tersedia di Amazon Redshift sebagai fungsi SQL. Gunakan operasi CREATE MODEL untuk menentukan data pelatihan baik sebagai tabel atau pernyataan SELECT.

Contoh kasus penggunaan

Anda dapat memecahkan masalah regresi lainnya dengan Amazon Redshift ML, seperti memprediksi nilai seumur hidup pelanggan. Anda juga dapat menggunakan Redshift ML untuk memprediksi harga yang paling menguntungkan dan pendapatan yang dihasilkan dari suatu produk.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Validasi model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus menyelesaikan pengaturan Administratif untuk Amazon Redshift ML.

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

Gunakan editor kueri Amazon Redshift v2 untuk menjalankan kueri berikut.

1. Anda harus membuat tiga tabel untuk memuat tiga kumpulan data publik ke Amazon Redshift. Kumpulan data adalah Data Pen unggang Sepeda Toronto, data cuaca historis, dan data hari libur historis. Jalankan kueri berikut di editor kueri Amazon Redshift untuk membuat tabel bernama`ridership`,`weather`, dan`holiday`.

```
CREATE TABLE IF NOT EXISTS ridership (
    trip_id INT,
    trip_duration_seconds INT,
    trip_start_time timestamp,
    trip_stop_time timestamp,
    from_station_name VARCHAR(50),
    to_station_name VARCHAR(50),
    from_station_id SMALLINT,
    to_station_id SMALLINT,
```

```
    user_type VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS weather (
    longitude_x DECIMAL(5, 2),
    latitude_y DECIMAL(5, 2),
    station_name VARCHAR(20),
    climate_id BIGINT,
    datetime_utc TIMESTAMP,
    weather_year SMALLINT,
    weather_month SMALLINT,
    weather_day SMALLINT,
    time_utc VARCHAR(5),
    temp_c DECIMAL(5, 2),
    temp_flag VARCHAR(1),
    dew_point_temp_c DECIMAL(5, 2),
    dew_point_temp_flag VARCHAR(1),
    rel_hum SMALLINT,
    rel_hum_flag VARCHAR(1),
    precip_amount_mm DECIMAL(5, 2),
    precip_amount_flag VARCHAR(1),
    wind_dir_10s_deg VARCHAR(10),
    wind_dir_flag VARCHAR(1),
    wind_spd_kmh VARCHAR(10),
    wind_spd_flag VARCHAR(1),
    visibility_km VARCHAR(10),
    visibility_flag VARCHAR(1),
    stn_press_kpa DECIMAL(5, 2),
    stn_press_flag VARCHAR(1),
    hmdx SMALLINT,
    hmdx_flag VARCHAR(1),
    wind_chill VARCHAR(10),
    wind_chill_flag VARCHAR(1),
    weather VARCHAR(10)
);

CREATE TABLE IF NOT EXISTS holiday (holiday_date DATE, description VARCHAR(100));
```

2. Kueri berikut memuat data sampel ke dalam tabel yang Anda buat pada langkah sebelumnya.

```
COPY ridership
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/ridership/'
    IAM_ROLE default
```

```
    FORMAT CSV
    IGNOREHEADER 1
    DATEFORMAT 'auto'
    TIMEFORMAT 'auto'
    REGION 'us-west-2'
    gzip;

COPY weather
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/weather/'
    IAM_ROLE default
    FORMAT csv
    IGNOREHEADER 1
    DATEFORMAT 'auto'
    TIMEFORMAT 'auto'
    REGION 'us-west-2'
    gzip;

COPY holiday
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/holiday/'
    IAM_ROLE default
    FORMAT csv
    IGNOREHEADER 1
    DATEFORMAT 'auto'
    TIMEFORMAT 'auto'
    REGION 'us-west-2'
    gzip;
```

3. Kueri berikut melakukan transformasi pada `weather` kumpulan data `ridership` dan untuk menghapus bias atau anomali. Menghapus bias dan anomali menghasilkan peningkatan akurasi model. Kueri menyederhanakan tabel dengan membuat dua tampilan baru yang disebut `ridership_view` dan`weather_view`.

```
CREATE
OR REPLACE VIEW ridership_view AS
SELECT
    trip_time,
    trip_count,
    TO_CHAR(trip_time, 'hh24') :: INT trip_hour,
    TO_CHAR(trip_time, 'dd') :: INT trip_day,
    TO_CHAR(trip_time, 'mm') :: INT trip_month,
    TO_CHAR(trip_time, 'yy') :: INT trip_year,
```

```
    TO_CHAR(trip_time, 'q') :: INT trip_quarter,
    TO_CHAR(trip_time, 'w') :: INT trip_month_week,
    TO_CHAR(trip_time, 'd') :: INT trip_week_day
FROM
    (
        SELECT
            CASE
                WHEN TRUNC(r.trip_start_time) < '2017-07-01' :: DATE THEN
 CONVERT_TIMEZONE(
                    'US/Eastern',
                    DATE_TRUNC('hour', r.trip_start_time)
                )
                ELSE DATE_TRUNC('hour', r.trip_start_time)
            END trip_time,
            COUNT(1) trip_count
        FROM
            ridership r
        WHERE
            r.trip_duration_seconds BETWEEN 60
            AND 60 * 60 * 24
        GROUP BY
            1
    );

CREATE
OR REPLACE VIEW weather_view AS
SELECT
    CONVERT_TIMEZONE(
        'US/Eastern',
        DATE_TRUNC('hour', datetime_utc)
    ) daytime,
    ROUND(AVG(temp_c)) temp_c,
    ROUND(AVG(precip_amount_mm)) precip_amount_mm
FROM
    weather
GROUP BY
    1;
```

4. Kueri berikut membuat tabel yang menggabungkan semua atribut input yang relevan dari `ridership_view` dan `weather_view` ke dalam `trip_data` tabel.

```
CREATE TABLE trip_data AS
SELECT
```

```
    r.trip_time,
    r.trip_count,
    r.trip_hour,
    r.trip_day,
    r.trip_month,
    r.trip_year,
    r.trip_quarter,
    r.trip_month_week,
    r.trip_week_day,
    w.temp_c,
    w.precip_amount_mm,CASE
        WHEN h.holiday_date IS NOT NULL THEN 1
        WHEN TO_CHAR(r.trip_time, 'D') :: INT IN (1, 7) THEN 1
        ELSE 0
    END is_holiday,
    ROW_NUMBER() OVER (
        ORDER BY
            RANDOM()
    ) serial_number
FROM
    ridership_view r
    JOIN weather_view w ON (r.trip_time = w.daytime)
    LEFT OUTER JOIN holiday h ON (TRUNC(r.trip_time) = h.holiday_date);
```

Lihat data sampel (opsional)

Kueri berikut menunjukkan entri dari tabel. Anda dapat menjalankan operasi ini untuk memastikan tabel dibuat dengan benar.

```
SELECT *
FROM trip_data
LIMIT 5;
```

Berikut ini adalah contoh output dari operasi sebelumnya.

```
+--------------------+------------+-----------+----------+------------+-----------
+--------------+-----------------+---------------+--------+-----------------
+------------+---------------+
|      trip_time      | trip_count | trip_hour | trip_day | trip_month | trip_year
 | trip_quarter | trip_month_week | trip_week_day | temp_c | precip_amount_mm |
 is_holiday | serial_number |
```

```
+---------------------+------------+-----------+----------+------------+-----------
+--------------+-----------------+---------------+--------+-----------------
+------------+---------------+
| 2017-03-21 22:00:00 |         47 |        22 |       21 |          3 |        17 |
         1 |               3 |             3 |      1 |                0 |          0 |
             1 |
| 2018-05-04 01:00:00 |         19 |         1 |        4 |          5 |        18 |
         2 |               1 |             6 |     12 |                0 |          0 |
             3 |
| 2018-01-11 10:00:00 |         93 |        10 |       11 |          1 |        18 |
         1 |               2 |             5 |      9 |                0 |          0 |
             5 |
| 2017-10-28 04:00:00 |         20 |         4 |       28 |         10 |        17 |
         4 |               4 |             7 |     11 |                0 |          1 |
             7 |
| 2017-12-31 21:00:00 |         11 |        21 |       31 |         12 |        17 |
         4 |               5 |             1 |    -15 |                0 |          1 |
             9 |
+---------------------+------------+-----------+----------+------------+-----------
+--------------+-----------------+---------------+--------+-----------------
+------------+---------------+
```

Tampilkan korelasi antara atribut (opsional)

Menentukan korelasi membantu Anda mengukur kekuatan hubungan antara atribut. Tingkat asosiasi dapat membantu Anda menentukan apa yang mempengaruhi output target Anda. Dalam tutorial ini, output target adalah`trip_count`.

Kueri berikut membuat atau menggantikan `sp_correlation` prosedur. Anda menggunakan prosedur tersimpan yang di `sp_correlation` panggil untuk menunjukkan korelasi antara atribut dan atribut lainnya dalam tabel di Amazon Redshift.

```
CREATE OR REPLACE PROCEDURE sp_correlation(source_schema_name in varchar(255),
 source_table_name in varchar(255), target_column_name in varchar(255),
 output_temp_table_name inout varchar(255)) AS $$
DECLARE
  v_sql varchar(max);
  v_generated_sql varchar(max);
  v_source_schema_name varchar(255)=lower(source_schema_name);
  v_source_table_name varchar(255)=lower(source_table_name);
  v_target_column_name varchar(255)=lower(target_column_name);
BEGIN
  EXECUTE 'DROP TABLE IF EXISTS ' || output_temp_table_name;
```

```
  v_sql = '
SELECT
  ''CREATE temp table '|| output_temp_table_name||' AS SELECT ''|| outer_calculation||
  '' FROM (SELECT COUNT(1) number_of_items, SUM('||v_target_column_name||')
 sum_target, SUM(POW('||v_target_column_name||',2)) sum_square_target, POW(SUM('||
v_target_column_name||'),2) square_sum_target,''||
  inner_calculation||
  '' FROM (SELECT ''||
  column_name||
  '' FROM '||v_source_table_name||'))''
FROM
  (
  SELECT
    DISTINCT
    LISTAGG(outer_calculation,'','') OVER () outer_calculation
    ,LISTAGG(inner_calculation,'','') OVER () inner_calculation
    ,LISTAGG(column_name,'','') OVER () column_name
  FROM
    (
    SELECT
      CASE WHEN atttypid=16 THEN ''DECODE(''||column_name||'',true,1,0)'' ELSE
 column_name END column_name
      ,atttypid
      ,''CAST(DECODE(number_of_items * sum_square_''||rn||'' - square_sum_''||
rn||'',0,null,(number_of_items*sum_target_''||rn||'' - sum_target * sum_''||rn||
        '')/SQRT((number_of_items * sum_square_target - square_sum_target) *
 (number_of_items * sum_square_''||rn||
        '' - square_sum_''||rn||''))) AS numeric(5,2)) ''||column_name
 outer_calculation
      ,''sum(''||column_name||'') sum_''||rn||'',''||
            ''SUM(trip_count*''||column_name||'') sum_target_''||rn||'',''||
            ''SUM(POW(''||column_name||'',2)) sum_square_''||rn||'',''||
            ''POW(SUM(''||column_name||''),2) square_sum_''||rn inner_calculation
    FROM
      (
      SELECT
        row_number() OVER (order by a.attnum) rn
        ,a.attname::VARCHAR column_name
        ,a.atttypid
      FROM pg_namespace AS n
        INNER JOIN pg_class AS c ON n.oid = c.relnamespace
        INNER JOIN pg_attribute AS a ON c.oid = a.attrelid
      WHERE a.attnum > 0
        AND n.nspname = '''||v_source_schema_name||'''
```

```
        AND c.relname = '''||v_source_table_name||'''
        AND a.atttypid IN (16,20,21,23,700,701,1700)
      )
    )
)';
  EXECUTE v_sql INTO v_generated_sql;
  EXECUTE  v_generated_sql;
END;
$$ LANGUAGE plpgsql;
```

Kueri berikut menunjukkan korelasi antara kolom target,`trip_count`, dan atribut numerik lainnya dalam dataset kami.

```
call sp_correlation(
    'public',
    'trip_data',
    'trip_count',
    'tmp_corr_table'
);

SELECT
    *
FROM
    tmp_corr_table;
```

Contoh berikut menunjukkan output dari `sp_correlation` operasi sebelumnya.

```
+------------+-----------+----------+------------+-----------+--------------
+-----------------+---------------+--------+------------------+------------
+---------------+
| trip_count | trip_hour | trip_day | trip_month | trip_year | trip_quarter
 | trip_month_week | trip_week_day | temp_c | precip_amount_mm | is_holiday |
 serial_number |
+------------+-----------+----------+------------+-----------+--------------
+-----------------+---------------+--------+------------------+------------
+---------------+
|          1 |      0.32 |     0.01 |       0.18 |      0.12 |         0.18 |
     0 |          0.02 |   0.53 |            -0.07 |      -0.13 |             0 |
+------------+-----------+----------+------------+-----------+--------------
+-----------------+---------------+--------+------------------+------------
+---------------+
```

Langkah 2: Buat model pembelajaran mesin

1. Kueri berikut membagi data Anda menjadi satu set pelatihan dan set validasi dengan menunjuk 80% dari dataset untuk pelatihan dan 20% untuk validasi. Set pelatihan adalah masukan untuk model ML untuk mengidentifikasi algoritma terbaik untuk model. Setelah model dibuat, Anda menggunakan set validasi untuk memvalidasi akurasi model.

```
CREATE TABLE training_data AS
SELECT
    trip_count,
    trip_hour,
    trip_day,
    trip_month,
    trip_year,
    trip_quarter,
    trip_month_week,
    trip_week_day,
    temp_c,
    precip_amount_mm,
    is_holiday
FROM
    trip_data
WHERE
    serial_number > (
        SELECT
            COUNT(1) * 0.2
        FROM
            trip_data
    );

CREATE TABLE validation_data AS
SELECT
    trip_count,
    trip_hour,
    trip_day,
    trip_month,
    trip_year,
    trip_quarter,
    trip_month_week,
    trip_week_day,
    temp_c,
    precip_amount_mm,
    is_holiday,
```

```
    trip_time
FROM
    trip_data
WHERE
    serial_number <= (
        SELECT
            COUNT(1) * 0.2
        FROM
            trip_data
    );
```

2. Kueri berikut membuat model regresi untuk memprediksi `trip_count` nilai untuk setiap tanggal dan waktu input. Dalam contoh berikut, ganti amzn-s3-demo-bucket dengan bucket S3 Anda sendiri.

```
CREATE MODEL predict_rental_count
FROM
    training_data TARGET trip_count FUNCTION predict_rental_count
    IAM_ROLE default
    PROBLEM_TYPE regression
    OBJECTIVE 'mse'
    SETTINGS (
        s3_bucket 'amzn-s3-demo-bucket',
        s3_garbage_collect off,
        max_runtime 5000
    );
```

Langkah 3: Validasi model

1. Gunakan kueri berikut untuk menampilkan aspek model, dan temukan metrik kesalahan kuadrat rata-rata dalam output. Kesalahan kuadrat rata-rata adalah metrik akurasi tipikal untuk masalah regresi.

```
show model predict_rental_count;
```

2. Jalankan kueri prediksi berikut terhadap data validasi untuk membandingkan jumlah perjalanan yang diprediksi dengan jumlah perjalanan aktual.

```
SELECT
    trip_time,
    actual_count,
```

```
    predicted_count,
    (actual_count - predicted_count) difference
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
        FROM
            validation_data
    )
LIMIT
    5;
```

3. Kueri berikut menghitung kesalahan kuadrat rata-rata dan kesalahan kuadrat rata-rata akar berdasarkan data validasi Anda. Anda menggunakan kesalahan kuadrat rata-rata dan kesalahan kuadrat rata-rata akar untuk mengukur jarak antara target numerik yang diprediksi dan jawaban numerik yang sebenarnya. Model yang baik memiliki skor rendah di kedua metrik. Kueri berikut mengembalikan nilai kedua metrik.

```
SELECT
    ROUND(
        AVG(POWER((actual_count - predicted_count), 2)),
        2
    ) mse,
    ROUND(
        SQRT(AVG(POWER((actual_count - predicted_count), 2))),
        2
    ) rmse
FROM
    (
        SELECT
```

```
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
        FROM
            validation_data
    );
```

4. Kueri berikut menghitung persentase kesalahan dalam jumlah perjalanan untuk setiap waktu perjalanan pada 01/01/2017. Kueri mengurutkan waktu perjalanan dari waktu dengan kesalahan persen terendah hingga waktu dengan kesalahan persen tertinggi.

```
SELECT
    trip_time,
    CAST(ABS(((actual_count - predicted_count) / actual_count)) * 100 AS DECIMAL
 (7,2)) AS pct_error
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
```

```
        FROM
            validation_data
    )
WHERE
   trip_time LIKE '2017-01-01 %%:%%:%%'
ORDER BY
   2 ASC;
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon Redshift ML
- Operasi CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Tutorial: Membangun model regresi dengan pembelajar linier

Dalam tutorial ini, Anda membuat model pembelajar linier dengan data dari Amazon S3 dan menjalankan kueri prediksi dengan model menggunakan Amazon Redshift ML. Algoritma pembelajar linier SageMaker AI memecahkan masalah regresi atau klasifikasi multi-kelas. Untuk mempelajari selengkapnya tentang masalah regresi dan klasifikasi multi-kelas, lihat Jenis masalah untuk paradigma pembelajaran mesin di Panduan Pengembang SageMaker AI Amazon. Dalam tutorial ini, Anda memecahkan masalah regresi. Algoritma pembelajar linier melatih banyak model secara paralel, dan secara otomatis menentukan model yang paling dioptimalkan. Anda menggunakan operasi CREATE MODEL di Amazon Redshift, yang membuat model pembelajar linier menggunakan SageMaker AI dan mengirimkan fungsi prediksi ke Amazon Redshift. Untuk informasi selengkapnya tentang algoritma pembelajar linier, lihat Algoritma Pembelajar Linear di Panduan Pengembang SageMaker AI Amazon.

Anda dapat menggunakan perintah CREATE MODEL untuk mengekspor data pelatihan, melatih model, mengimpor model, dan menyiapkan fungsi prediksi Amazon Redshift. Gunakan operasi CREATE MODEL untuk menentukan data pelatihan baik sebagai tabel atau pernyataan SELECT.

Model pembelajar linier mengoptimalkan tujuan berkelanjutan atau tujuan diskrit. Tujuan kontinu digunakan untuk regresi, sedangkan variabel diskrit digunakan untuk klasifikasi. Beberapa metode memberikan solusi hanya untuk tujuan berkelanjutan, seperti metode regresi. Algoritma pembelajar linier memberikan peningkatan kecepatan dibandingkan teknik optimasi hyperparameter naif, seperti teknik Naive Bayes. Teknik optimasi naif mengasumsikan bahwa setiap variabel input independen. Untuk menggunakan algoritma pembelajar linier, Anda harus menyediakan kolom yang mewakili dimensi input, dan baris yang mewakili pengamatan. Untuk informasi selengkapnya tentang algoritma pembelajar linier, lihat Algoritma Pembelajar Linear di Panduan Pengembang SageMaker AI Amazon.

Dalam tutorial ini, Anda membangun model pembelajar linier yang memprediksi usia abalon. Anda menggunakan perintah CREATE MODEL pada dataset Abalone untuk menentukan hubungan antara pengukuran fisik abalone. Kemudian, Anda menggunakan model untuk menentukan usia abalon.

Contoh kasus penggunaan

Anda dapat memecahkan masalah regresi lainnya dengan linear learner dan Amazon Redshift ML, seperti memprediksi harga rumah. Anda juga dapat menggunakan Redshift ML untuk memprediksi jumlah orang yang akan menggunakan layanan penyewaan sepeda kota.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Validasi model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus menyelesaikan pengaturan Administratif untuk Amazon Redshift ML.

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

Gunakan editor kueri Amazon Redshift v2 untuk menjalankan kueri berikut. Kueri ini memuat data sampel ke Redshift dan membagi data menjadi set pelatihan dan set validasi.

1. Kueri berikut membuat `abalone_dataset` tabel.

```
CREATE TABLE abalone_dataset (
    id INT IDENTITY(1, 1),
    Sex CHAR(1),
    Length float,
    Diameter float,
    Height float,
    Whole float,
    Shucked float,
    Viscera float,
    Shell float,
    Rings integer
);
```

2. Kueri berikut menyalin data sampel dari dataset Abalone di Amazon S3 ke `abalone_dataset` tabel yang Anda buat sebelumnya di Amazon Redshift.

```
COPY abalone_dataset
FROM
    's3://redshift-ml-multiclass/abalone.csv' REGION 'us-east-1' IAM_ROLE default CSV
 IGNOREHEADER 1 NULL AS 'NULL';
```

3. Dengan membagi data secara manual, Anda akan dapat memverifikasi keakuratan model dengan mengalokasikan set prediksi tambahan. Kueri berikut membagi data menjadi dua set. `abalone_training`Tabel ini untuk pelatihan dan `abalone_validation` tabel untuk validasi.

```
CREATE TABLE abalone_training as
SELECT
    *
FROM
    abalone_dataset
WHERE
    mod(id, 10) < 8;

CREATE TABLE abalone_validation as
SELECT
```

```
    *
FROM
    abalone_dataset
WHERE
    mod(id, 10) >= 8;
```

Langkah 2: Buat model pembelajaran mesin

Pada langkah ini, Anda menggunakan pernyataan CREATE MODEL untuk membuat model pembelajaran mesin Anda dengan algoritma pembelajar linier.

Kueri berikut membuat model pembelajar linier dengan operasi CREATE MODEL menggunakan bucket S3 Anda. Ganti amzn-s3-demo-bucket dengan bucket S3 Anda sendiri.

```
CREATE MODEL model_abalone_ring_prediction
FROM
    (
        SELECT
            Sex,
            Length,
            Diameter,
            Height,
            Whole,
            Shucked,
            Viscera,
            Shell,
            Rings AS target_label
        FROM
            abalone_training
    ) TARGET target_label FUNCTION f_abalone_ring_prediction IAM_ROLE default
 MODEL_TYPE LINEAR_LEARNER PROBLEM_TYPE REGRESSION OBJECTIVE 'MSE' SETTINGS (
        S3_BUCKET 'amzn-s3-demo-bucket',
        MAX_RUNTIME 15000
    );
```

Tampilkan status pelatihan model (opsional)

Anda dapat menggunakan perintah SHOW MODEL untuk mengetahui kapan model Anda siap.

Gunakan kueri berikut untuk memantau kemajuan pelatihan model.

```
SHOW MODEL model_abalone_ring_prediction;
```

Ketika model siap, output dari operasi sebelumnya akan terlihat mirip dengan contoh berikut. Perhatikan bahwa output menyediakan `validation:mse` metrik, yang merupakan kesalahan kuadrat rata-rata. Anda akan menggunakan kesalahan persegi rata-rata untuk memvalidasi keakuratan model pada langkah berikutnya.

```
+--------------------------
+-----------------------------------------------------------------------------------------------
+
|        Model Name        |
 model_abalone_ring_prediction                                    |
+--------------------------
+-----------------------------------------------------------------------------------------------
+
| Schema Name              | public
                                        |
| Owner                    | awsuser
                                        |
| Creation Time            | Thu, 30.06.2022 18:00:10
                                        |
| Model State              | READY
                                        |
| validation:mse           |
                                   4.168633 |
| Estimated Cost           |
                                   4.291608 |
|                          |
                                        |
| TRAINING DATA:           |
                                        |
| Query                    | SELECT SEX , LENGTH , DIAMETER , HEIGHT , WHOLE ,
 SHUCKED , VISCERA , SHELL, RINGS AS TARGET_LABEL |
|                          | FROM ABALONE_TRAINING
                                        |
| Target Column            | TARGET_LABEL
                                        |
|                          |
                                        |
| PARAMETERS:              |
                                        |
```

```
| Model Type                | linear_learner
                                        |
| Problem Type              | Regression
                                        |
| Objective                 | MSE
                                        |
| AutoML Job Name           | redshiftml-20220630180010947843
                                        |
| Function Name             | f_abalone_ring_prediction
                                        |
| Function Parameters       | sex length diameter height whole shucked viscera shell
                                        |
| Function Parameter Types | bpchar float8 float8 float8 float8 float8 float8 float8
                                        |
| IAM Role                  | default-aws-iam-role
                                        |
| S3 Bucket                 | amzn-s3-demo-bucket
                                        |
| Max Runtime               |
                                  15000 |
+--------------------------
+-----------------------------------------------------------------------------------------------
+
```

Langkah 3: Validasi model

1. Kueri prediksi berikut memvalidasi keakuratan model pada `abalone_validation` dataset dengan menghitung kesalahan kuadrat rata-rata dan kesalahan kuadrat rata-rata akar.

```
SELECT
    ROUND(AVG(POWER((tgt_label - predicted), 2)), 2) mse,
    ROUND(SQRT(AVG(POWER((tgt_label - predicted), 2))), 2) rmse
FROM
    (
        SELECT
            Sex,
            Length,
            Diameter,
            Height,
            Whole,
            Shucked,
            Viscera,
            Shell,
```

```
            Rings AS tgt_label,
            f_abalone_ring_prediction(
                Sex,
                Length,
                Diameter,
                Height,
                Whole,
                Shucked,
                Viscera,
                Shell
            ) AS predicted,
            CASE
                WHEN tgt_label = predicted then 1
                ELSE 0
            END AS match,
            CASE
                WHEN tgt_label <> predicted then 1
                ELSE 0
            END AS nonmatch
        FROM
            abalone_validation
    ) t1;
```

Output dari kueri sebelumnya akan terlihat seperti contoh berikut. Nilai metrik kesalahan kuadrat rata-rata harus serupa dengan `validation:mse` metrik yang ditunjukkan oleh output operasi SHOW MODEL.

```
+-----+--------------------+
| mse |        rmse        |
+-----+--------------------+
| 5.1 | 2.2600000000000002 |
+-----+--------------------+
```

2. Gunakan kueri berikut untuk menjalankan operasi EXPLAIN_MODEL pada fungsi prediksi Anda. Operasi akan mengembalikan laporan penjelasan model. Untuk informasi selengkapnya tentang operasi EXPLAIN_MODEL, lihat fungsi EXPLA IN_MODEL di Panduan Pengembang Database Amazon Redshift.

```
SELECT
    EXPLAIN_MODEL ('model_abalone_ring_prediction');
```

Informasi berikut adalah contoh laporan keterangan model yang dihasilkan oleh operasi EXPLAIN_MODEL sebelumnya. Nilai untuk masing-masing input adalah nilai Shapley. Nilai Shapley mewakili efek setiap input pada prediksi model Anda, dengan input bernilai lebih tinggi memiliki dampak lebih besar pada prediksi. Dalam contoh ini, input bernilai lebih tinggi memiliki dampak lebih besar pada memprediksi usia abalon.

```
{
    "explanations": {
        "kernel_shap": {
            "label0": {
                "expected_value" :10.290688514709473,
                "global_shap_values": {
                    "diameter" :0.6856910187882492,
                    "height" :0.4415323937124035,
                    "length" :0.21507476107609084,
                    "sex" :0.448611774505744,
                    "shell" :1.70426496893776,
                    "shucked" :2.1181392924386994,
                    "viscera" :0.342220754059912,
                    "whole" :0.6711906974084011
                }
            }
        }
    },
    "version" :"1.0"
};
```

3. Gunakan kueri berikut untuk menghitung persentase prediksi yang benar yang dibuat model tentang abalon yang belum matang. Abalon yang belum matang memiliki 10 cincin atau kurang, dan prediksi yang benar akurat hingga dalam satu cincin dari jumlah cincin yang sebenarnya.

```
SELECT
    TRUNC(
        SUM(
            CASE
                WHEN ROUND(
                    f_abalone_ring_prediction(
                        Sex,
                        Length,
                        Diameter,
                        Height,
```

```
                        Whole,
                        Shucked,
                        Viscera,
                        Shell
                    ),
                    0
                ) BETWEEN Rings - 1
                AND Rings + 1 THEN 1
                ELSE 0
            END
        ) / CAST(COUNT(SHELL) AS FLOAT),
        4
    ) AS prediction_pct
FROM
    abalone_validation
WHERE
    Rings <= 10;
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon Redshift ML
- Operasi CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Tutorial: Membangun model klasifikasi multi-kelas dengan pembelajar linier

Dalam tutorial ini, Anda membuat model pembelajar linier dengan data dari Amazon S3, lalu menjalankan kueri prediksi dengan model menggunakan Amazon Redshift ML. Algoritma pembelajar linier SageMaker AI memecahkan masalah regresi atau klasifikasi. Untuk mempelajari selengkapnya tentang masalah regresi dan klasifikasi multi-kelas, lihat Jenis masalah untuk

paradigma pembelajaran mesin di Panduan Pengembang SageMaker AI Amazon. Dalam tutorial ini, Anda memecahkan masalah klasifikasi multi-kelas. Algoritma pembelajar linier melatih banyak model secara paralel, dan secara otomatis menentukan model yang paling dioptimalkan. Anda menggunakan operasi CREATE MODEL di Amazon Redshift, yang membuat model pembelajar linier menggunakan SageMaker AI dan mengirimkan fungsi prediksi ke Amazon Redshift. Untuk informasi selengkapnya tentang algoritma pembelajar linier, lihat Algoritma Pembelajar Linear di Panduan Pengembang SageMaker AI Amazon.

Anda dapat menggunakan perintah CREATE MODEL untuk mengekspor data pelatihan, melatih model, mengimpor model, dan menyiapkan fungsi prediksi Amazon Redshift. Gunakan operasi CREATE MODEL untuk menentukan data pelatihan baik sebagai tabel atau pernyataan SELECT.

Model pembelajar linier mengoptimalkan tujuan berkelanjutan atau tujuan diskrit. Tujuan kontinu digunakan untuk regresi, sedangkan variabel diskrit digunakan untuk klasifikasi. Beberapa metode memberikan solusi hanya untuk tujuan berkelanjutan, seperti metode regresi. Algoritma pembelajar linier memberikan peningkatan kecepatan dibandingkan teknik optimasi hyperparameter naif, seperti teknik Naive Bayes. Teknik optimasi naif mengasumsikan bahwa setiap variabel input independen. Algoritma pembelajar linier melatih banyak model secara paralel dan memilih model yang paling dioptimalkan. Algoritma serupa adalah XGBoost, yang menggabungkan perkiraan dari serangkaian model yang lebih sederhana dan lebih lemah untuk membuat prediksi. Untuk mempelajari lebih lanjut tentang XGBoost, lihat algoritma XGBoost di Panduan Pengembang AI Amazon SageMaker .

Untuk menggunakan algoritma pembelajar linier, Anda harus menyediakan kolom yang mewakili dimensi input, dan baris yang mewakili pengamatan. Untuk informasi selengkapnya tentang algoritma pembelajar linier, lihat Algoritma Pembelajar Linear di Panduan Pengembang SageMaker AI Amazon.

Dalam tutorial ini, Anda membangun model pembelajar linier yang memprediksi jenis penutup untuk area tertentu. Anda menggunakan perintah CREATE MODEL pada dataset Covertype dari Repositori Pembelajaran Mesin UCI. Kemudian, Anda menggunakan fungsi prediksi yang dibuat oleh perintah untuk menentukan jenis penutup di daerah hutan belantara. Jenis tutupan hutan biasanya merupakan jenis pohon. Input yang akan digunakan Redshift ML untuk membuat model termasuk jenis tanah, jarak ke jalan raya, dan penunjukan area hutan belantara. Untuk informasi selengkapnya tentang dataset, lihat Dataset Covertype dari Repositori Pembelajaran Mesin UCI.

Contoh kasus penggunaan

Anda dapat memecahkan masalah klasifikasi multi-kelas lainnya dengan pembelajar linier dengan Amazon Redshift ML, seperti memprediksi spesies tanaman dari gambar. Anda juga dapat memprediksi jumlah produk yang akan dibeli pelanggan.

Tugas

- Prasyarat
- Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift
- Langkah 2: Buat model pembelajaran mesin
- Langkah 3: Validasi model

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus menyelesaikan pengaturan Administratif untuk Amazon Redshift ML.

Langkah 1: Muat data dari Amazon S3 ke Amazon Redshift

Gunakan editor kueri Amazon Redshift v2 untuk menjalankan kueri berikut. Kueri ini memuat data sampel ke Redshift dan membagi data menjadi set pelatihan dan set validasi.

1. Kueri berikut membuat `covertype_data` tabel.

```
CREATE TABLE public.covertype_data (
    elevation bigint ENCODE az64,
    aspect bigint ENCODE az64,
    slope bigint ENCODE az64,
    horizontal_distance_to_hydrology bigint ENCODE az64,
    vertical_distance_to_hydrology bigint ENCODE az64,
    horizontal_distance_to_roadways bigint ENCODE az64,
    hillshade_9am bigint ENCODE az64,
    hillshade_noon bigint ENCODE az64,
    hillshade_3pm bigint ENCODE az64,
    horizontal_distance_to_fire_points bigint ENCODE az64,
    wilderness_area1 bigint ENCODE az64,
    wilderness_area2 bigint ENCODE az64,
    wilderness_area3 bigint ENCODE az64,
    wilderness_area4 bigint ENCODE az64,
    soil_type1 bigint ENCODE az64,
```

```
    soil_type2 bigint ENCODE az64,
    soil_type3 bigint ENCODE az64,
    soil_type4 bigint ENCODE az64,
    soil_type5 bigint ENCODE az64,
    soil_type6 bigint ENCODE az64,
    soil_type7 bigint ENCODE az64,
    soil_type8 bigint ENCODE az64,
    soil_type9 bigint ENCODE az64,
    soil_type10 bigint ENCODE az64,
    soil_type11 bigint ENCODE az64,
    soil_type12 bigint ENCODE az64,
    soil_type13 bigint ENCODE az64,
    soil_type14 bigint ENCODE az64,
    soil_type15 bigint ENCODE az64,
    soil_type16 bigint ENCODE az64,
    soil_type17 bigint ENCODE az64,
    soil_type18 bigint ENCODE az64,
    soil_type19 bigint ENCODE az64,
    soil_type20 bigint ENCODE az64,
    soil_type21 bigint ENCODE az64,
    soil_type22 bigint ENCODE az64,
    soil_type23 bigint ENCODE az64,
    soil_type24 bigint ENCODE az64,
    soil_type25 bigint ENCODE az64,
    soil_type26 bigint ENCODE az64,
    soil_type27 bigint ENCODE az64,
    soil_type28 bigint ENCODE az64,
    soil_type29 bigint ENCODE az64,
    soil_type30 bigint ENCODE az64,
    soil_type31 bigint ENCODE az64,
    soil_type32 bigint ENCODE az64,
    soil_type33 bigint ENCODE az64,
    soil_type34 bigint ENCODE az64,
    soil_type35 bigint ENCODE az64,
    soil_type36 bigint ENCODE az64,
    soil_type37 bigint ENCODE az64,
    soil_type38 bigint ENCODE az64,
    soil_type39 bigint ENCODE az64,
    soil_type40 bigint ENCODE az64,
    cover_type bigint ENCODE az64
) DISTSTYLE AUTO;
```

2. Kueri berikut menyalin data sampel dari dataset Covertype di Amazon S3 ke `covertype_data` tabel yang Anda buat sebelumnya di Amazon Redshift.

```
COPY public.covertype_data
FROM
    's3://redshift-ml-multiclass/covtype.data.gz' IAM_ROLE DEFAULT gzip DELIMITER ','
 REGION 'us-east-1';
```

3. Dengan membagi data secara manual, Anda akan dapat memverifikasi keakuratan model dengan mengalokasikan set pengujian tambahan. Kueri berikut membagi data menjadi tiga set. `covertype_training`Tabel ini untuk pelatihan, `covertype_validation` tabel untuk validasi, dan `covertype_test` tabel untuk menguji model Anda. Anda akan menggunakan set pelatihan untuk melatih model Anda dan set validasi untuk memvalidasi pengembangan model. Kemudian, Anda menggunakan set pengujian untuk menguji kinerja model dan melihat apakah model tersebut terlalu pas atau kurang sesuai dengan dataset.

```
CREATE TABLE public.covertype_data_prep AS
SELECT
    a.*,
    CAST (random() * 100 AS int) AS data_group_id
FROM
    public.covertype_data a;

--training dataset
 CREATE TABLE public.covertype_training as
SELECT
    *
FROM
    public.covertype_data_prep
WHERE
    data_group_id < 80;

--validation dataset
 CREATE TABLE public.covertype_validation AS
SELECT
    *
FROM
    public.covertype_data_prep
WHERE
    data_group_id BETWEEN 80
    AND 89;

--test dataset
 CREATE TABLE public.covertype_test AS
```

```
SELECT
    *
FROM
    public.covertype_data_prep
WHERE
    data_group_id > 89;
```

Langkah 2: Buat model pembelajaran mesin

Pada langkah ini, Anda menggunakan pernyataan CREATE MODEL untuk membuat model pembelajaran mesin Anda dengan algoritma pembelajar linier.

Kueri berikut membuat model pembelajar linier dengan operasi CREATE MODEL menggunakan bucket S3 Anda. Ganti amzn-s3-demo-bucket dengan bucket S3 Anda sendiri.

```
CREATE MODEL forest_cover_type_model
FROM
    (
        SELECT
            Elevation,
            Aspect,
            Slope,
            Horizontal_distance_to_hydrology,
            Vertical_distance_to_hydrology,
            Horizontal_distance_to_roadways,
            HIllshade_9am,
            Hillshade_noon,
            Hillshade_3pm,
            Horizontal_Distance_To_Fire_Points,
            Wilderness_Area1,
            Wilderness_Area2,
            Wilderness_Area3,
            Wilderness_Area4,
            soil_type1,
            Soil_Type2,
            Soil_Type3,
            Soil_Type4,
            Soil_Type5,
            Soil_Type6,
            Soil_Type7,
            Soil_Type8,
            Soil_Type9,
```

```
            Soil_Type10,
            Soil_Type11,
            Soil_Type12,
            Soil_Type13,
            Soil_Type14,
            Soil_Type15,
            Soil_Type16,
            Soil_Type17,
            Soil_Type18,
            Soil_Type19,
            Soil_Type20,
            Soil_Type21,
            Soil_Type22,
            Soil_Type23,
            Soil_Type24,
            Soil_Type25,
            Soil_Type26,
            Soil_Type27,
            Soil_Type28,
            Soil_Type29,
            Soil_Type30,
            Soil_Type31,
            Soil_Type32,
            Soil_Type33,
            Soil_Type34,
            Soil_Type36,
            Soil_Type37,
            Soil_Type38,
            Soil_Type39,
            Soil_Type40,
            Cover_type
        from
            public.covertype_training
    ) TARGET cover_type FUNCTION predict_cover_type IAM_ROLE default MODEL_TYPE
LINEAR_LEARNER PROBLEM_TYPE MULTICLASS_CLASSIFICATION OBJECTIVE 'Accuracy' SETTINGS (
        S3_BUCKET 'amzn-s3-demo-bucket',
        S3_GARBAGE_COLLECT OFF,
        MAX_RUNTIME 15000
    );
```

Tampilkan status pelatihan model (opsional)

Anda dapat menggunakan perintah SHOW MODEL untuk mengetahui kapan model Anda siap.

Gunakan kueri berikut untuk memantau kemajuan pelatihan model.

```
SHOW MODEL forest_cover_type_model;
```

Ketika model siap, output dari operasi sebelumnya akan terlihat mirip dengan contoh berikut. Perhatikan bahwa output menyediakan `validation:multiclass_accuracy` metrik, yang dapat Anda lihat di sisi kanan contoh berikut. Multi-class akurasi mengukur persentase titik data yang diklasifikasikan dengan benar oleh model. Anda akan menggunakan akurasi multi-kelas untuk memvalidasi keakuratan model di langkah berikutnya.

```
+--------------------------------
+-------------------------------------------------------------------------------------------------
+
|              Key               |


 Value


                                                        |
+--------------------------------
+-------------------------------------------------------------------------------------------------
+
| Model Name                     | forest_cover_type_model


                                                       |
| Schema Name                    | public
```

```
                                                    |
| Owner                          | awsuser

                                                    |
| Creation Time                  | Tue, 12.07.2022 20:24:32

                                                    |
| Model State                    | READY

                                                    |
| validation:multiclass_accuracy |

                                          0.724952 |
```

```
| Estimated Cost                 |

                                                5.341750 |
|                                |

                                                         |
| TRAINING DATA:                 |

                                                         |
| Query                          | SELECT ELEVATION, ASPECT, SLOPE,
 HORIZONTAL_DISTANCE_TO_HYDROLOGY, VERTICAL_DISTANCE_TO_HYDROLOGY,
 HORIZONTAL_DISTANCE_TO_ROADWAYS, HILLSHADE_9AM, HILLSHADE_NOON, HILLSHADE_3PM ,
 HORIZONTAL_DISTANCE_TO_FIRE_POINTS, WILDERNESS_AREA1, WILDERNESS_AREA2,
 WILDERNESS_AREA3, WILDERNESS_AREA4, SOIL_TYPE1, SOIL_TYPE2, SOIL_TYPE3, SOIL_TYPE4,
 SOIL_TYPE5, SOIL_TYPE6, SOIL_TYPE7, SOIL_TYPE8, SOIL_TYPE9, SOIL_TYPE10 , SOIL_TYPE11,
 SOIL_TYPE12 , SOIL_TYPE13 , SOIL_TYPE14, SOIL_TYPE15, SOIL_TYPE16, SOIL_TYPE17,
 SOIL_TYPE18, SOIL_TYPE19, SOIL_TYPE20, SOIL_TYPE21, SOIL_TYPE22, SOIL_TYPE23,
 SOIL_TYPE24, SOIL_TYPE25, SOIL_TYPE26, SOIL_TYPE27, SOIL_TYPE28, SOIL_TYPE29,
 SOIL_TYPE30, SOIL_TYPE31, SOIL_TYPE32, SOIL_TYPE33, SOIL_TYPE34, SOIL_TYPE36,
 SOIL_TYPE37, SOIL_TYPE38, SOIL_TYPE39, SOIL_TYPE40, COVER_TYPE |
|                                | FROM PUBLIC.COVERTYPE_TRAINING
```

```
                                                    |
| Target Column                  | COVER_TYPE

                                                    |
|                                |

                                                    |
| PARAMETERS:                    |

                                                    |
| Model Type                     | linear_learner
```

```
                                                    |
| Problem Type                    | MulticlassClassification

                                                    |
| Objective                       | Accuracy

                                                    |
| AutoML Job Name                 | redshiftml-20220712202432187659

                                                    |
| Function Name                   | predict_cover_type

                                                    |
```

```
| Function Parameters            | elevation aspect slope
 horizontal_distance_to_hydrology vertical_distance_to_hydrology
 horizontal_distance_to_roadways hillshade_9am hillshade_noon hillshade_3pm
 horizontal_distance_to_fire_points wilderness_area1 wilderness_area2 wilderness_area3
 wilderness_area4 soil_type1 soil_type2 soil_type3 soil_type4 soil_type5 soil_type6
 soil_type7 soil_type8 soil_type9 soil_type10 soil_type11 soil_type12 soil_type13
 soil_type14 soil_type15 soil_type16 soil_type17 soil_type18 soil_type19 soil_type20
 soil_type21 soil_type22 soil_type23 soil_type24 soil_type25 soil_type26 soil_type27
 soil_type28 soil_type29 soil_type30 soil_type31 soil_type32 soil_type33 soil_type34
 soil_type36 soil_type37 soil_type38 soil_type39 soil_type40
                                                 |
| Function Parameter Types       | int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
 int8 int8 int8 int8 int8 int8 int8 int8 int8

                                                       |
| IAM Role                       | default-aws-iam-role

                                                      |
| S3 Bucket                      | amzn-s3-demo-bucket

                                                      |
| Max Runtime                    |
```

```
                                                15000 |
+--------------------------------
+-----------------------------------------------------------------------------------------------
+
```

Langkah 3: Validasi model

1. Kueri prediksi berikut memvalidasi keakuratan model pada `covertype_validation` dataset dengan menghitung akurasi multi-kelas. Multi-class akurasi adalah persentase prediksi model yang benar.

```
SELECT
    CAST(sum(t1.match) AS decimal(7, 2)) AS predicted_matches,
    CAST(sum(t1.nonmatch) AS decimal(7, 2)) AS predicted_non_matches,
    CAST(sum(t1.match + t1.nonmatch) AS decimal(7, 2)) AS total_predictions,
    predicted_matches / total_predictions AS pct_accuracy
FROM
    (
        SELECT
            Elevation,
            Aspect,
            Slope,
            Horizontal_distance_to_hydrology,
            Vertical_distance_to_hydrology,
            Horizontal_distance_to_roadways,
            HIllshade_9am,
            Hillshade_noon,
            Hillshade_3pm,
            Horizontal_Distance_To_Fire_Points,
            Wilderness_Area1,
            Wilderness_Area2,
            Wilderness_Area3,
            Wilderness_Area4,
            soil_type1,
            Soil_Type2,
            Soil_Type3,
            Soil_Type4,
```

```
            Soil_Type5,
            Soil_Type6,
            Soil_Type7,
            Soil_Type8,
            Soil_Type9,
            Soil_Type10,
            Soil_Type11,
            Soil_Type12,
            Soil_Type13,
            Soil_Type14,
            Soil_Type15,
            Soil_Type16,
            Soil_Type17,
            Soil_Type18,
            Soil_Type19,
            Soil_Type20,
            Soil_Type21,
            Soil_Type22,
            Soil_Type23,
            Soil_Type24,
            Soil_Type25,
            Soil_Type26,
            Soil_Type27,
            Soil_Type28,
            Soil_Type29,
            Soil_Type30,
            Soil_Type31,
            Soil_Type32,
            Soil_Type33,
            Soil_Type34,
            Soil_Type36,
            Soil_Type37,
            Soil_Type38,
            Soil_Type39,
            Soil_Type40,
            Cover_type AS actual_cover_type,
            predict_cover_type(
                Elevation,
                Aspect,
                Slope,
                Horizontal_distance_to_hydrology,
                Vertical_distance_to_hydrology,
                Horizontal_distance_to_roadways,
                HIllshade_9am,
```

```
                Hillshade_noon,
                Hillshade_3pm,
                Horizontal_Distance_To_Fire_Points,
                Wilderness_Area1,
                Wilderness_Area2,
                Wilderness_Area3,
                Wilderness_Area4,
                soil_type1,
                Soil_Type2,
                Soil_Type3,
                Soil_Type4,
                Soil_Type5,
                Soil_Type6,
                Soil_Type7,
                Soil_Type8,
                Soil_Type9,
                Soil_Type10,
                Soil_Type11,
                Soil_Type12,
                Soil_Type13,
                Soil_Type14,
                Soil_Type15,
                Soil_Type16,
                Soil_Type17,
                Soil_Type18,
                Soil_Type19,
                Soil_Type20,
                Soil_Type21,
                Soil_Type22,
                Soil_Type23,
                Soil_Type24,
                Soil_Type25,
                Soil_Type26,
                Soil_Type27,
                Soil_Type28,
                Soil_Type29,
                Soil_Type30,
                Soil_Type31,
                Soil_Type32,
                Soil_Type33,
                Soil_Type34,
                Soil_Type36,
                Soil_Type37,
                Soil_Type38,
```

```
                Soil_Type39,
                Soil_Type40
            ) AS predicted_cover_type,
            CASE
                WHEN actual_cover_type = predicted_cover_type THEN 1
                ELSE 0
            END AS match,
            CASE
                WHEN actual_cover_type <> predicted_cover_type THEN 1
                ELSE 0
            END AS nonmatch
        FROM
            public.covertype_validation
    ) t1;
```

Output dari kueri sebelumnya akan terlihat seperti contoh berikut. Nilai metrik akurasi multi-kelas harus serupa dengan `validation:multiclass_accuracy` metrik yang ditunjukkan oleh output operasi SHOW MODEL.

```
+------------------+----------------------+------------------+--------------+
| predicted_matches | predicted_non_matches | total_predictions | pct_accuracy |
+------------------+----------------------+------------------+--------------+
|             41211 |                16324 |             57535 |   0.71627704 |
+------------------+----------------------+------------------+--------------+
```

2. Kueri berikut memprediksi jenis sampul yang paling umum untuk`wilderness_area2`. Dataset ini mencakup empat area hutan belantara dan tujuh jenis penutup. Daerah hutan belantara dapat memiliki beberapa jenis penutup.

```
SELECT t1. predicted_cover_type, COUNT(*)
FROM
(
SELECT
   Elevation,
   Aspect,
   Slope,
   Horizontal_distance_to_hydrology,
   Vertical_distance_to_hydrology,
   Horizontal_distance_to_roadways,
   HIllshade_9am,
   Hillshade_noon,
   Hillshade_3pm ,
```

```
    Horizontal_Distance_To_Fire_Points,
    Wilderness_Area1,
    Wilderness_Area2,
    Wilderness_Area3,
    Wilderness_Area4,
    soil_type1,
    Soil_Type2,
    Soil_Type3,
    Soil_Type4,
    Soil_Type5,
    Soil_Type6,
    Soil_Type7,
    Soil_Type8,
    Soil_Type9,
    Soil_Type10 ,
    Soil_Type11,
    Soil_Type12 ,
    Soil_Type13 ,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
    Soil_Type30,
    Soil_Type31,
    Soil_Type32,
    Soil_Type33,
    Soil_Type34,
    Soil_Type36,
    Soil_Type37,
    Soil_Type38,
    Soil_Type39,
    Soil_Type40,
```

```
    predict_cover_type( Elevation,
    Aspect,
    Slope,
    Horizontal_distance_to_hydrology,
    Vertical_distance_to_hydrology,
    Horizontal_distance_to_roadways,
    HIllshade_9am,
    Hillshade_noon,
    Hillshade_3pm ,
    Horizontal_Distance_To_Fire_Points,
    Wilderness_Area1,
    Wilderness_Area2,
    Wilderness_Area3,
    Wilderness_Area4,
    soil_type1,
    Soil_Type2,
    Soil_Type3,
    Soil_Type4,
    Soil_Type5,
    Soil_Type6,
    Soil_Type7,
    Soil_Type8,
    Soil_Type9,
    Soil_Type10,
    Soil_Type11,
    Soil_Type12,
    Soil_Type13,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
    Soil_Type30,
```

```
   Soil_Type31,
   Soil_Type32,
   Soil_Type33,
   Soil_Type34,
   Soil_Type36,
   Soil_Type37,
   Soil_Type38,
   Soil_Type39,
   Soil_Type40) AS predicted_cover_type

FROM public.covertype_test
WHERE wilderness_area2 = 1)
t1
GROUP BY 1;
```

Output dari operasi sebelumnya harus terlihat mirip dengan contoh berikut. Output ini berarti bahwa model memperkirakan bahwa sebagian besar penutup adalah tipe penutup 1, dan ada beberapa penutup tipe penutup 2 dan 7.

```
+----------------------+-------+
| predicted_cover_type | count |
+----------------------+-------+
|                    2 |   564 |
|                    7 |    97 |
|                    1 |  2309 |
+----------------------+-------+
```

3. Kueri berikut menunjukkan jenis penutup yang paling umum di satu area hutan belantara. Kueri menampilkan jumlah jenis penutup dan area hutan belantara jenis penutup.

```
SELECT t1. predicted_cover_type, COUNT(*), wilderness_area
FROM
(
SELECT
   Elevation,
   Aspect,
   Slope,
   Horizontal_distance_to_hydrology,
   Vertical_distance_to_hydrology,
   Horizontal_distance_to_roadways,
   HIllshade_9am,
   Hillshade_noon,
```

```
    Hillshade_3pm ,
    Horizontal_Distance_To_Fire_Points,
    Wilderness_Area1,
    Wilderness_Area2,
    Wilderness_Area3,
    Wilderness_Area4,
    soil_type1,
    Soil_Type2,
    Soil_Type3,
    Soil_Type4,
    Soil_Type5,
    Soil_Type6,
    Soil_Type7,
    Soil_Type8,
    Soil_Type9,
    Soil_Type10 ,
    Soil_Type11,
    Soil_Type12 ,
    Soil_Type13 ,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
    Soil_Type30,
    Soil_Type31,
    Soil_Type32,
    Soil_Type33,
    Soil_Type34,
    Soil_Type36,
    Soil_Type37,
    Soil_Type38,
    Soil_Type39,
```

```
    Soil_Type40,
    predict_cover_type( Elevation,
    Aspect,
    Slope,
    Horizontal_distance_to_hydrology,
    Vertical_distance_to_hydrology,
    Horizontal_distance_to_roadways,
    HIllshade_9am,
    Hillshade_noon,
    Hillshade_3pm ,
    Horizontal_Distance_To_Fire_Points,
    Wilderness_Area1,
    Wilderness_Area2,
    Wilderness_Area3,
    Wilderness_Area4,
    soil_type1,
    Soil_Type2,
    Soil_Type3,
    Soil_Type4,
    Soil_Type5,
    Soil_Type6,
    Soil_Type7,
    Soil_Type8,
    Soil_Type9,
    Soil_Type10,
    Soil_Type11,
    Soil_Type12,
    Soil_Type13,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
```

```
   Soil_Type30,
   Soil_Type31,
   Soil_Type32,
   Soil_Type33,
   Soil_Type34,
   Soil_Type36,
   Soil_Type37,
   Soil_Type38,
   Soil_Type39,
   Soil_Type40) AS predicted_cover_type,
   CASE WHEN Wilderness_Area1 = 1 THEN 1
        WHEN Wilderness_Area2 = 1 THEN 2
        WHEN Wilderness_Area3 = 1 THEN 3
        WHEN Wilderness_Area4 = 1 THEN 4
        ELSE 0
   END AS wilderness_area

FROM public.covertype_test)
t1
GROUP BY 1, 3
ORDER BY 2 DESC
LIMIT 1;
```

Output dari operasi sebelumnya harus terlihat mirip dengan contoh berikut.

```
+----------------------+-------+-----------------+
| predicted_cover_type | count | wilderness_area |
+----------------------+-------+-----------------+
|                    2 | 15738 |               1 |
+----------------------+-------+-----------------+
```

Topik terkait

Untuk informasi selengkapnya tentang Amazon Redshift ML, lihat dokumentasi berikut:

- Biaya untuk menggunakan Amazon Redshift ML
- Operasi CREATE MODEL
- Fungsi EXPLAIN_MODEL

Untuk informasi selengkapnya tentang pembelajaran mesin, lihat dokumentasi berikut:

- Ikhtisar pembelajaran mesin
- Pembelajaran mesin untuk pemula dan ahli
- Apa itu Keadilan dan Penjelasan Model untuk Prediksi Pembelajaran Mesin?

Integrasi Amazon Redshift ML dengan Amazon Bedrock

Bagian ini menjelaskan cara menggunakan integrasi Amazon Redshift ML dengan Amazon Bedrock. Dengan fitur ini, Anda dapat memanggil model Amazon Bedrock menggunakan SQL, dan Anda dapat menggunakan data Anda dari gudang data Amazon Redshift untuk membangun aplikasi AI generatif seperti pembuatan teks, analisis sentimen, atau terjemahan.

Topik

- Membuat atau memperbarui peran IAM untuk integrasi Amazon Redshift ML dengan Amazon Bedrock
- Membuat model eksternal untuk integrasi Amazon Redshift ML dengan Amazon Bedrock
- Menggunakan model eksternal untuk integrasi Amazon Redshift ML dengan Amazon Bedrock
- Rekayasa cepat untuk integrasi Amazon Redshift ML dengan Amazon Bedrock

Membuat atau memperbarui peran IAM untuk integrasi Amazon Redshift ML dengan Amazon Bedrock

Bagian ini menunjukkan cara membuat peran IAM untuk digunakan dengan integrasi Amazon Redshift ML dengan Amazon Bedrock.

Tambahkan kebijakan berikut ke peran IAM yang Anda gunakan dengan integrasi Amazon Redshift ML dengan Amazon Bedrock:

- `AmazonBedrockFullAccess`

Untuk mengizinkan Amazon Redshift mengambil peran untuk berinteraksi dengan layanan lain, tambahkan kebijakan kepercayaan berikut ke peran IAM:

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "redshift.amazonaws.com"
            ]
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```

Jika cluster atau namespace berada di VPC, ikuti instruksi di. Pengaturan pengelompokan dan konfigurasikan untuk administrasi Amazon Redshift ML

Jika Anda memerlukan kebijakan yang lebih ketat, Anda dapat membuat kebijakan yang hanya menyertakan izin Amazon Bedrock yang ditentukan di halaman berikut:

- Pengaturan pengelompokan dan konfigurasikan untuk administrasi Amazon Redshift ML
- Izin diperlukan untuk menggunakan pembelajaran mesin Amazon Redshift (ML)

Untuk informasi tentang membuat peran IAM, lihat Pembuatan Peran IAM di Panduan AWS Identity and Access Management Pengguna.

Membuat model eksternal untuk integrasi Amazon Redshift ML dengan Amazon Bedrock

Bagian ini menunjukkan cara membuat model eksternal untuk digunakan sebagai antarmuka untuk Amazon Bedrock dalam gudang data Amazon Redshift Anda.

Untuk memanggil model Amazon Bedrock dari Amazon Redshift, Anda harus menjalankan perintah terlebih dahulu. `CREATE EXTERNAL MODEL` Perintah ini membuat objek model eksternal dalam database, dan fungsi pengguna terkait yang Anda gunakan untuk menghasilkan konten teks dengan Amazon Bedrock.

Contoh kode berikut menunjukkan `CREATE EXTERNAL MODEL` perintah dasar:

```
CREATE EXTERNAL MODEL llm_claude
```

```
FUNCTION llm_claude_func
IAM_ROLE '<IAM role arn>'
MODEL_TYPE BEDROCK
SETTINGS (
   MODEL_ID 'anthropic.claude-v2:1',
   PROMPT 'Summarize the following text:');
```

Per `CREATE EXTERNAL MODEL` intah ini memiliki antarmuka yang terpadu dan konsisten dengan Amazon Bedrock untuk semua Model Foundation (FM) yang mendukung pesan. Ini adalah opsi default saat menggunakan `CREATE EXTERNAL MODEL` perintah atau ketika secara eksplisit menentukan jenis permintaan`UNIFIED`. Untuk informasi selengkapnya, lihat dokumentasi API Converse di dokumentasi Amazon Bedrock API.

Jika FM tidak mendukung pesan, maka Anda harus mengatur `request_type` pengaturan ke`RAW`, seperti yang ditunjukkan contoh berikut. Saat Anda menyet `RAW` el `request_type` ke, Anda harus membuat permintaan yang dikirim ke Amazon Bedrock saat menggunakan fungsi inferensi berdasarkan FM yang dipilih. Pastikan Anda mengaktifkan akses ke model Titan Text G1 — Express di Amazon Bedrock sebelum menjalankan contoh berikut.

```
CREATE EXTERNAL MODEL titan_raw
FUNCTION func_titan_raw
IAM_ROLE '<IAM role arn>'
MODEL_TYPE BEDROCK
SETTINGS (
   MODEL_ID 'amazon.titan-text-express-v1',
   REQUEST_TYPE RAW,
   RESPONSE_TYPE SUPER);
```

Jika Anda memerlukan informasi lebih lanjut tentang permintaan input seperti token total, Anda dapat meminta `RESPONSE_TYPE` untuk menjadi `super` saat Anda membuat model.

```
CREATE EXTERNAL MODEL patient_recommendations_v2
FUNCTION patient_recommendations_func_v2
IAM_ROLE '<IAM role arn>'
MODEL_TYPE BEDROCK
SETTINGS (
   MODEL_ID 'anthropic.claude-v2',
   PROMPT 'Generate personalized diet plan for following patient:',
   RESPONSE_TYPE SUPER);
```

`PROMPT`Parameter untuk `CREATE EXTERNAL MODEL` perintah adalah prompt statis. Jika Anda memerlukan prompt dinamis untuk aplikasi Anda, Anda harus menentukannya saat menggunakan fungsi inferensi. Untuk detail selengkapnya, lihat Rekayasa cepat untuk integrasi Amazon Redshift ML dengan Amazon Bedrock.

Untuk informasi selengkapnya tentang `CREATE EXTERNAL MODEL` pernyataan dan parameter serta pengaturannya, lihatBUAT MODEL EKSTERNAL.

Menggunakan model eksternal untuk integrasi Amazon Redshift ML dengan Amazon Bedrock

Bagian ini menunjukkan cara memanggil model eksternal untuk menghasilkan teks sebagai respons terhadap prompt yang disediakan. Untuk memanggil model eksternal, gunakan fungsi inferensi yang Anda buat dengan. `CREATE EXTERNAL MODEL`

Topik

- Inferensi dengan model tipe permintaan UNIFIED
- Inferensi dengan model tipe permintaan RAW
- Fungsi inferensi sebagai fungsi khusus pemimpin
- Catatan penggunaan fungsi inferensi

Inferensi dengan model tipe permintaan UNIFIED

Fungsi inferensi untuk model dengan tipe permintaan `UNIFIED` memiliki tiga parameter berikut yang diteruskan ke fungsi secara berurutan:

- Teks masukan (wajib): Parameter ini menentukan teks masukan yang diteruskan Amazon Redshift ke Amazon Bedrock.
- Konfigurasi inferensi dan bidang permintaan model tambahan (opsional): Amazon Redshift meneruskan parameter ini ke parameter yang sesuai untuk API model Converse.

Contoh kode berikut menunjukkan cara menggunakan fungsi inferensi `UNIFIED` tipe:

```
SELECT llm_claude_func(input_text, object('temperature', 0.7, 'maxtokens', 500))
   FROM some_data;
```

Inferensi dengan **`model tipe`** permintaan RAW

Fungsi inferensi untuk model dengan tipe permintaan hanya `RAW` memiliki satu parameter tipe `SUPER` data. Sintaks parameter ini tergantung pada model Amazon Bedrock yang digunakan.

Contoh kode berikut menunjukkan cara menggunakan fungsi inferensi `RAW` tipe:

```
SELECT llm_titan_func(
    object(
        "inputText", "Summarize the following text: " | input_text,
        "textGenerationConfig", object("temperature", 0.5, "maxTokenCount", 500)
    )
)
FROM some_data;
```

Fungsi inferensi sebagai fungsi khusus pemimpin

Fungsi inferensi untuk model Amazon Bedrock dapat dijalankan sebagai fungsi utama node ketika kueri yang menggunakannya tidak mereferensikan tabel apa pun. Ini dapat membantu jika Anda ingin dengan cepat mengajukan pertanyaan kepada LLM.

Contoh kode berikut menunjukkan cara menggunakan fungsi inferensi khusus pemimpin:

```
SELECT general_titan_llm_func('Summarize the benefits of LLM on data analytics in 100
 words');
```

Catatan penggunaan fungsi inferensi

Perhatikan hal berikut saat menggunakan fungsi inferensi dengan integrasi Amazon Redshift ML dengan Amazon Bedrock:

- Nama-nama parameter untuk semua model Amazon Bedrock peka huruf besar/kecil. Jika parameter Anda tidak cocok dengan yang diperlukan oleh model, Amazon Bedrock mungkin diam-diam mengabaikannya.
- Throughput kueri inferensi dibatasi oleh kuota runtime dari berbagai model yang ditawarkan oleh Amazon Bedrock di Wilayah yang berbeda. Untuk informasi selengkapnya, lihat Kuota untuk Amazon Bedrock di Panduan Pengguna Amazon Bedrock.
- Jika Anda membutuhkan throughput yang terjamin dan konsisten, pertimbangkan untuk mendapatkan throughput yang disediakan untuk model yang Anda butuhkan dari Amazon Bedrock.

Untuk informasi selengkapnya, lihat Mening katkan kapasitas pemanggilan model dengan Throughput Tersediakan di Amazon Bedrock di Panduan Pengguna Amazon Bedrock.

- Kueri inferensi dengan sejumlah besar data mungkin mendapatkan pengecualian pelambatan. Ini karena kuota runtime terbatas untuk Amazon Bedrock. Amazon Redshift mencoba kembali permintaan beberapa kali, tetapi kueri masih dapat dibatasi karena throughput untuk model yang tidak disediakan mungkin bervariasi.
- Jika Anda menemukan pengecualian pelambatan yang berasal dari Amazon Bedrock seperti `Too many requests, please wait before trying again.` bahkan dengan sejumlah kecil data, periksa kuota di bawah Kuota Layanan di akun Amazon Bedrock Anda. Periksa apakah kuota tingkat akun yang diterapkan setidaknya sama dengan nilai kuota AWS default untuk InvokeModel permintaan untuk model yang Anda gunakan.

Rekayasa cepat untuk integrasi Amazon Redshift ML dengan Amazon Bedrock

Bagian ini menunjukkan cara menggunakan prompt statis dengan model eksternal.

Untuk menggunakan petunjuk awalan statis dan akhiran untuk model eksternal Anda, berikan mereka menggunakan `SUFFIX` parameter `PROMPT` dan `CREATE EXTERNAL MODEL` pernyataan. Prompt ini ditambahkan ke setiap kueri menggunakan model eksternal.

Contoh berikut menunjukkan cara menambahkan petunjuk awalan dan akhiran ke model eksternal:

```
CREATE EXTERNAL MODEL llm_claude
FUNCTION llm_claude_func
IAM_ROLE '<IAM role arn>'
MODEL_TYPE BEDROCK
SETTINGS (
   MODEL_ID 'anthropic.claude-v2:1',
   PROMPT 'Summarize the following text:',
   SUFFIX 'Respond in an analytic tone');
```

Untuk menggunakan prompt dinamis, Anda dapat menyediakannya saat menggunakan fungsi inferensi dengan menggabungkannya dalam input fungsi. Contoh berikut menunjukkan cara menggunakan prompt dinamis dengan fungsi inferensi:

```
SELECT llm_claude_func('Summarize the following review:' | input_text | 'The review
 should have formal tone.')
```

```
FROM some_data
```

Penyetelan kinerja kueri

Amazon Redshift menggunakan kueri berdasarkan bahasa kueri terstruktur (SQL) untuk berinteraksi dengan data dan objek dalam sistem. Bahasa manipulasi data (DML) adalah subset SQL yang Anda gunakan untuk melihat, menambah, mengubah, dan menghapus data. Bahasa definisi data (DDL) adalah subset SQL yang Anda gunakan untuk menambah, mengubah, dan menghapus objek database seperti tabel dan tampilan.

Setelah sistem Anda diatur, Anda biasanya bekerja dengan DML paling banyak, terutama SELECT perintah untuk mengambil dan melihat data. Untuk menulis kueri pengambilan data yang efektif di Amazon Redshift, kenali SELECT dan terapkan tips yang diuraikan dalam Praktik terbaik Amazon Redshift untuk merancang tabel untuk memaksimalkan efisiensi kueri.

Untuk memahami bagaimana Amazon Redshift memproses kueri, gunakan Analisis dan peningkatan kueri bagian Pemrosesan kueri dan. Kemudian Anda dapat menerapkan informasi ini dalam kombinasi dengan alat diagnostik untuk mengidentifikasi dan menghapus masalah dalam kinerja kueri.

Untuk mengidentifikasi dan mengatasi beberapa masalah paling umum dan paling serius yang mungkin Anda temui dengan kueri Amazon Redshift, gunakan Pemecahan masalah kueri bagian ini.

Topik

- Pemrosesan kueri
- Analisis dan peningkatan kueri
- Pemecahan masalah kueri

Pemrosesan kueri

Amazon Redshift merutekan kueri SQL yang dikirimkan melalui parser dan pengoptimal untuk mengembangkan rencana kueri. Mesin eksekusi kemudian menerjemahkan rencana kueri ke dalam kode dan mengirimkan kode itu ke node komputasi untuk dieksekusi.

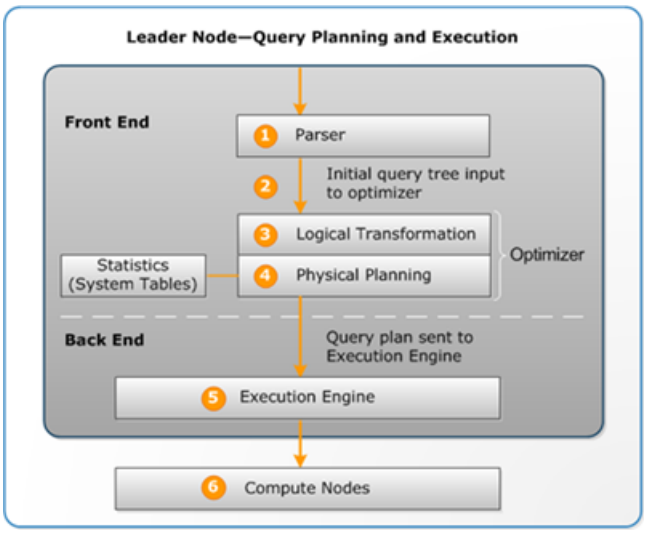
Topik

- Perencanaan kueri dan alur kerja pelaksanaan
- Membuat dan menafsirkan rencana kueri
- Meninjau langkah-langkah rencana kueri

- Faktor-faktor yang mempengaruhi kinerja kueri

Perencanaan kueri dan alur kerja pelaksanaan

Ilustrasi berikut memberikan tampilan tingkat tinggi dari alur kerja perencanaan kueri dan eksekusi.



Alur kerja perencanaan kueri dan eksekusi mengikuti langkah-langkah berikut:

1. Node pemimpin menerima kueri dan mengurai SQL.
2. Parser menghasilkan pohon kueri awal yang merupakan representasi logis dari kueri asli. Amazon Redshift kemudian memasukkan pohon kueri ini ke pengoptimal kueri.
3. Pengoptimal mengevaluasi dan jika perlu menulis ulang kueri untuk memaksimalkan efisiensinya. Proses ini terkadang menghasilkan pembuatan beberapa kueri terkait untuk menggantikan satu kueri.
4. Pengoptimal menghasilkan rencana kueri (atau beberapa, jika langkah sebelumnya menghasilkan beberapa kueri) untuk eksekusi dengan kinerja terbaik. Rencana kueri menentukan opsi eksekusi seperti tipe gabungan, urutan gabungan, opsi agregasi, dan persyaratan distribusi data.

   Anda dapat menggunakan EXPLAIN perintah untuk melihat rencana kueri. Rencana kueri adalah alat mendasar untuk menganalisis dan menyetel kueri kompleks. Untuk informasi selengkapnya, lihat Membuat dan menafsirkan rencana kueri.
5. Mesin eksekusi menerjemahkan rencana kueri ke dalam langkah-langkah, segmen, dan aliran:

Langkah

Setiap langkah adalah operasi individual yang diperlukan selama eksekusi query. Langkah-langkah dapat digabungkan untuk memungkinkan node komputasi melakukan query, join, atau operasi database lainnya.

Segment

Kombinasi beberapa langkah yang dapat dilakukan oleh satu proses, juga unit kompilasi terkecil yang dapat dieksekusi oleh irisan node komputasi. I ris an adalah unit pemrosesan paralel di Amazon Redshift. Segmen dalam aliran berjalan secara paralel.

Streaming

Kumpulan segmen yang akan dibagikan di atas irisan simpul komputasi yang tersedia.

Mesin eksekusi menghasilkan kode yang dikompilasi berdasarkan langkah, segmen, dan aliran. Kode yang dikompilasi berjalan lebih cepat daripada kode yang ditafsirkan dan menggunakan kapasitas komputasi yang lebih sedikit. Kode yang dikompilasi ini kemudian disiarkan ke node komputasi.

> **Note**
>
> Saat membandingkan kueri Anda, Anda harus selalu membandingkan waktu untuk eksekusi kedua kueri, karena waktu eksekusi pertama mencakup overhead kompilasi kode. Untuk informasi selengkapnya, lihat Faktor-faktor yang mempengaruhi kinerja kueri.

6. Irisan simpul komputasi menjalankan segmen kueri secara paralel. Sebagai bagian dari proses ini, Amazon Redshift memanfaatkan komunikasi jaringan, memori, dan manajemen disk yang dioptimalkan untuk meneruskan hasil perantara dari satu langkah rencana kueri ke langkah berikutnya. Ini juga membantu mempercepat eksekusi query.

Langkah 5 dan 6 terjadi sekali untuk setiap aliran. Mesin membuat segmen yang dapat dieksekusi untuk satu aliran dan mengirimkannya ke node komputasi. Ketika segmen aliran itu selesai, mesin menghasilkan segmen untuk aliran berikutnya. Dengan cara ini, mesin dapat menganalisis apa yang terjadi di aliran sebelumnya (misalnya, apakah operasi berbasis disk) untuk mempengaruhi generasi segmen di aliran berikutnya.

Ketika node komputasi selesai, mereka mengembalikan hasil kueri ke node pemimpin untuk pemrosesan akhir. Node pemimpin menggabungkan data ke dalam satu set hasil dan menangani penyortiran atau agregasi yang diperlukan. Node pemimpin kemudian mengembalikan hasil ke klien.

> **Note**
>
> Node komputasi mungkin mengembalikan beberapa data ke node pemimpin selama eksekusi kueri jika perlu. Misalnya, jika Anda memiliki subquery dengan klausa LIMIT, batas diterapkan pada node pemimpin sebelum data didistribusikan kembali di seluruh cluster untuk diproses lebih lanjut.

Membuat dan menafsirkan rencana kueri

Anda dapat menggunakan rencana kueri untuk mendapatkan informasi tentang operasi individual yang diperlukan untuk menjalankan kueri. Sebelum Anda bekerja dengan rencana kueri, sebaiknya Anda terlebih dahulu memahami bagaimana Amazon Redshift menangani kueri pemrosesan dan membuat rencana kueri. Untuk informasi selengkapnya, lihat Perencanaan kueri dan alur kerja pelaksanaan.

Untuk membuat rencana kueri, jalankan EXPLAIN perintah diikuti oleh teks kueri yang sebenarnya. Paket kueri memberi Anda informasi berikut:

- Operasi apa yang dilakukan mesin eksekusi, membaca hasilnya dari bawah ke atas.
- Jenis langkah apa yang dilakukan setiap operasi.
- Tabel dan kolom mana yang digunakan dalam setiap operasi.
- Berapa banyak data yang diproses dalam setiap operasi, dalam hal jumlah baris dan lebar data dalam byte.
- Biaya relatif operasi. Biaya adalah ukuran yang membandingkan waktu eksekusi relatif dari langkah-langkah dalam rencana. Biaya tidak memberikan informasi yang tepat tentang waktu eksekusi aktual atau konsumsi memori, juga tidak memberikan perbandingan yang berarti antara rencana eksekusi. Ini memberi Anda indikasi operasi mana dalam kueri yang menghabiskan sumber daya paling banyak.

Perintah EXPLAIN tidak benar-benar menjalankan kueri. Ini hanya menunjukkan rencana yang dijalankan Amazon Redshift jika kueri dijalankan dalam kondisi operasi saat ini. Jika Anda mengubah

skema atau data untuk tabel dan menjalan MENGANALISA kannya lagi untuk memperbarui metadata statistik, rencana kueri mungkin berbeda.

Output rencana kueri oleh EXPLAIN adalah tampilan eksekusi kueri tingkat tinggi yang disederhanakan. Itu tidak menggambarkan detail pemrosesan kueri paralel. Untuk melihat informasi terperinci, jalankan kueri itu sendiri, lalu dapatkan informasi ringkasan kueri dari tampilan SVL_QUERY_SUMMARY atau SVL_QUERY_REPORT. Untuk informasi selengkapnya tentang menggunakan tampilan ini, lihatMenganalisis ringkasan kueri.

Contoh berikut menunjukkan output EXPLAIN untuk kueri GROUP BY sederhana pada tabel EVENT:

```
explain select eventname, count(*) from event group by eventname;

                            QUERY PLAN
------------------------------------------------------------------------
XN HashAggregate  (cost=131.97..133.41 rows=576 width=17)
  ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=17)
```

EXPLAIN mengembalikan metrik berikut untuk setiap operasi:

Biaya

Nilai relatif yang berguna untuk membandingkan operasi dalam rencana. Biaya terdiri dari dua nilai desimal yang dipisahkan oleh dua periode, misalnya`cost=131.97..133.41`. Nilai pertama, dalam hal ini 131,97, memberikan biaya relatif untuk mengembalikan baris pertama untuk operasi ini. Nilai kedua, dalam hal ini 133,41, memberikan biaya relatif untuk menyelesaikan operasi. Biaya dalam rencana kueri bersifat kumulatif saat Anda membaca rencana, jadi HashAggregate biaya dalam contoh ini (131.97.. 133.41) termasuk biaya Seq Scan di bawahnya (0.00.. 87.98).

Baris

Perkiraan jumlah baris yang akan dikembalikan. Dalam contoh ini, pemindaian diharapkan mengembalikan 8798 baris. HashAggregate Operator sendiri diharapkan mengembalikan 576 baris (setelah nama peristiwa duplikat dibuang dari kumpulan hasil).

> **Note**
>
> Perkiraan baris didasarkan pada statistik yang tersedia yang dihasilkan oleh perintah ANALYZE. Jika ANALYZE belum dijalankan baru-baru ini, estimasi kurang dapat diandalkan.

Lebar

Perkiraan lebar baris rata-rata, dalam byte. Dalam contoh ini, baris rata-rata diharapkan lebar 17 byte.

JELASKAN operator

Bagian ini menjelaskan secara singkat operator yang paling sering Anda lihat dalam output EXPLAIN. Untuk daftar lengkap operator, lihat EXPLAIN di bagian Perintah SQL.

Operator pemindaian berurutan

Operator pemindaian sekuensial (Seq Scan) menunjukkan pemindaian tabel. Seq Scan memindai setiap kolom dalam tabel secara berurutan dari awal hingga akhir dan mengevaluasi batasan kueri (dalam klausa WHERE) untuk setiap baris.

Bergabunglah dengan operator

Amazon Redshift memilih operator gabungan berdasarkan desain fisik tabel yang digabungkan, lokasi data yang diperlukan untuk bergabung, dan persyaratan khusus kueri itu sendiri.

- Lingkaran bersarang

  Gabungan yang paling tidak optimal, loop bersarang digunakan terutama untuk sambungan silang (produk Cartesian) dan beberapa gabungan ketidaksetaraan.
- Hash Join dan Hash

  Biasanya lebih cepat daripada gabungan loop bersarang, gabungan hash dan hash digunakan untuk join dalam dan join luar kiri dan kanan. Operator ini digunakan saat menggabungkan tabel di mana kolom gabungan bukan kunci distribusi dan kunci pengurutan. Operator hash membuat tabel hash untuk tabel bagian dalam di join; operator hash join membaca tabel luar, men-hash kolom penggabungan, dan menemukan kecocokan di tabel hash bagian dalam.
- Gabungkan Gabung

  Biasanya gabungan tercepat, gabungan gabungan digunakan untuk gabungan dalam dan gabungan luar. Gabungan gabungan tidak digunakan untuk gabungan penuh. Operator ini digunakan saat menggabungkan tabel di mana kolom gabungan adalah kunci distribusi dan kunci peng urutan, dan ketika kurang dari 20 persen tabel penggabungan tidak disortir. Ini membaca dua tabel yang diurutkan secara berurutan dan menemukan baris yang cocok. Untuk melihat persentase baris yang tidak disortir, kueri tabel SVV_TABLE_INFO sistem.

- Gabung Spasial

  Biasanya penggabungan cepat berdasarkan kedekatan data spasial, digunakan untuk `GEOMETRY` dan `GEOGRAPHY` tipe data.

Operator agregat

Rencana kueri menggunakan operator berikut dalam kueri yang melibatkan fungsi agregat dan operasi GROUP BY.

- Agregat

  Operator untuk fungsi agregat skalar seperti AVG dan SUM.

- HashAggregate

  Operator untuk fungsi agregat yang dikelompokkan yang tidak disortir.

- GroupAggregate

  Operator untuk fungsi agregat yang dikelompokkan yang diurutkan.

Sortir operator

Paket kueri menggunakan operator berikut ketika kueri harus mengurutkan atau menggabungkan kumpulan hasil.

- Urutkan

  Mengevaluasi klausa ORDER BY dan operasi pengurutan lainnya, seperti pengurutan yang diperlukan oleh kueri dan gabungan UNION, kueri SELECT DISTINCT, dan fungsi jendela.

- Menggabungkan

  Menghasilkan hasil akhir yang diurutkan menurut hasil diurutkan menengah yang berasal dari operasi paralel.

Operator UNION, INTERSECT, dan EXCEPT

Rencana kueri menggunakan operator berikut untuk kueri yang melibatkan operasi set dengan UNION, INTERSECT, dan EXCEPT.

- Subkueri

  Digunakan untuk menjalankan query UNION.

- Hash Berpotongan Berbeda

  Digunakan untuk menjalankan query INTERSECT.

- SetOp Kecuali

  Digunakan untuk menjalankan query EXCEPT (atau MINUS).

Operator lainnya

Operator berikut juga sering muncul dalam output EXPLAIN untuk kueri rutin.

- Unik

  Menghapus duplikat untuk kueri SELECT DISTINCT dan query UNION.

- Batasi

  Memproses klausa LIMIT.

- Jendela

  Menjalankan fungsi jendela.

- Hasil

  Menjalankan fungsi skalar yang tidak melibatkan akses tabel apa pun.

- Subrencana

  Digunakan untuk subquery tertentu.

- Jaringan

  Mengirim hasil perantara ke node pemimpin untuk diproses lebih lanjut.

- Terwujud

  Menyimpan baris untuk input ke gabungan loop bersarang dan beberapa gabungan gabungan.

Bergabung dalam EXPLAIN

Pengoptimal kueri menggunakan jenis gabungan yang berbeda untuk mengambil data tabel, tergantung pada struktur kueri dan tabel yang mendasarinya. Output EXPLAIN mereferensikan jenis gabungan, tabel yang digunakan, dan cara data tabel didistribusikan di seluruh cluster untuk menggambarkan bagaimana query diproses.

Contoh tipe gabungan

Contoh berikut menunjukkan jenis gabungan yang berbeda yang dapat digunakan pengoptimal kueri. Jenis gabungan yang digunakan dalam rencana kueri tergantung pada desain fisik tabel yang terlibat.

Contoh: Hash menggabungkan dua tabel

Kueri berikut bergabung dengan EVENT dan CATEGORY pada kolom CATID. CATID adalah kunci distribusi dan pengurutan untuk CATEGORY tetapi tidak untuk EVENT. Gabungan hash dilakukan dengan EVENT sebagai tabel luar dan CATEGORY sebagai tabel dalam. Karena CATEGORY adalah tabel yang lebih kecil, perencana menyiarkan salinannya ke node komputasi selama pemrosesan kueri dengan menggunakan DS_BCAST_INNER. Biaya gabungan dalam contoh ini menyumbang sebagian besar biaya kumulatif paket.

```
explain select * from category, event where category.catid=event.catid;

                              QUERY PLAN
----------------------------------------------------------------------
 XN Hash Join DS_BCAST_INNER  (cost=0.14..6600286.07 rows=8798 width=84)
   Hash Cond: ("outer".catid = "inner".catid)
   ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=35)
   ->  XN Hash  (cost=0.11..0.11 rows=11 width=49)
         ->  XN Seq Scan on category  (cost=0.00..0.11 rows=11 width=49)
```

> **Note**
>
> Indentasi selaras untuk operator dalam output EXPLAIN terkadang menunjukkan bahwa operasi tersebut tidak bergantung satu sama lain dan dapat dimulai secara paralel. Dalam contoh sebelumnya, meskipun pemindaian pada tabel EVENT dan operasi hash disejajarkan, pemindaian EVENT harus menunggu sampai operasi hash selesai sepenuhnya.

Contoh: Gabungkan gabungkan dua tabel

Kueri berikut juga menggunakan SELECT *, tetapi bergabung dengan SALES dan LISTING pada kolom LISTID, di mana LISTID telah ditetapkan sebagai kunci distribusi dan pengurutan untuk kedua tabel. Gabungan gabungan dipilih, dan tidak ada redistribusi data yang diperlukan untuk bergabung (DS_DIST_NONE).

```
explain select * from sales, listing where sales.listid = listing.listid;
QUERY PLAN
------------------------------------------------------------------------------
XN Merge Join DS_DIST_NONE  (cost=0.00..6285.93 rows=172456 width=97)
  Merge Cond: ("outer".listid = "inner".listid)
  ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=44)
  ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=53)
```

Contoh berikut menunjukkan berbagai jenis gabungan dalam kueri yang sama. Seperti pada contoh sebelumnya, SALES dan LISTING digabungkan, tetapi tabel ketiga, EVENT, harus digabungkan dengan hash dengan hasil gabungan gabungan. Sekali lagi, hash join menimbulkan biaya siaran.

```
explain select * from sales, listing, event
where sales.listid = listing.listid and sales.eventid = event.eventid;
                                  QUERY PLAN
------------------------------------------------------------------------------
XN Hash Join DS_BCAST_INNER  (cost=109.98..3871130276.17 rows=172456 width=132)
  Hash Cond: ("outer".eventid = "inner".eventid)
  ->  XN Merge Join DS_DIST_NONE  (cost=0.00..6285.93 rows=172456 width=97)
        Merge Cond: ("outer".listid = "inner".listid)
        ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=44)
        ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=53)
  ->  XN Hash  (cost=87.98..87.98 rows=8798 width=35)
        ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=35)
```

Contoh: Gabung, agregasi, dan urutkan

Kueri berikut menjalankan gabungan hash dari tabel SALES dan EVENT, diikuti dengan operasi agregasi dan pengurutan untuk memperhitungkan fungsi SUM yang dikelompokkan dan klausa ORDER BY. Operator pengurutan awal berjalan secara paralel pada node komputasi. Kemudian operator Jaringan mengirimkan hasil ke node pemimpin, di mana operator Merge menghasilkan hasil akhir yang diurutkan.

```
explain select eventname, sum(pricepaid) from sales, event
```

```
where sales.eventid=event.eventid group by eventname
order by 2 desc;
                                          QUERY PLAN
------------------------------------------------------------------------------
 XN Merge  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
  Merge Key: sum(sales.pricepaid)
  ->  XN Network  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
        Send to leader
        ->  XN Sort  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
              Sort Key: sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=2815366577.07..2815366578.51 rows=576
 width=27)
                    ->  XN Hash Join DS_BCAST_INNER  (cost=109.98..2815365714.80
 rows=172456 width=27)
                          Hash Cond: ("outer".eventid = "inner".eventid)
                          ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456
 width=14)
                          ->  XN Hash  (cost=87.98..87.98 rows=8798 width=21)
                                ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798
 width=21)
```

Redistribusi data

Output EXPLAIN untuk join juga menentukan metode bagaimana data dipindahkan di sekitar cluster untuk memfasilitasi penggabungan. Pergerakan data ini dapat berupa siaran atau redistribusi. Dalam siaran, nilai data dari satu sisi gabungan disalin dari setiap node komputasi ke setiap node komputasi lainnya, sehingga setiap node komputasi berakhir dengan salinan lengkap data. Dalam redistribusi, nilai data yang berpartisipasi dikirim dari irisan mereka saat ini ke irisan baru (mungkin pada node yang berbeda). Data biasanya didistribusikan kembali untuk mencocokkan kunci distribusi tabel lain yang berpartisipasi dalam gabungan jika kunci distribusi itu adalah salah satu kolom penggabungan. Jika tidak ada tabel yang memiliki kunci distribusi pada salah satu kolom penggabungan, kedua tabel didistribusikan atau tabel bagian dalam disiarkan ke setiap node.

Output EXPLAIN juga mereferensikan tabel dalam dan luar. Tabel bagian dalam dipindai terlebih dahulu, dan muncul lebih dekat bagian bawah rencana kueri. Meja bagian dalam adalah tabel yang diperiksa untuk pertandingan. Biasanya disimpan dalam memori, biasanya tabel sumber untuk hashing, dan jika memungkinkan, adalah tabel yang lebih kecil dari keduanya yang digabungkan. Tabel luar adalah sumber baris untuk dicocokkan dengan tabel bagian dalam. Biasanya dibaca dari disk. Pengoptimal kueri memilih tabel dalam dan luar berdasarkan statistik database dari menjalankan perintah ANALYZE terbaru. Urutan tabel dalam klausa FROM dari kueri tidak menentukan tabel mana yang dalam dan mana yang luar.

Gunakan atribut berikut dalam rencana kueri untuk mengidentifikasi bagaimana data dipindahkan untuk memfasilitasi kueri:

- DS_BCAST_INNER

  Salinan seluruh tabel bagian dalam disiarkan ke semua node komputasi.
- DS_DIST_ALL_NONE

  Tidak diperlukan redistribusi, karena tabel bagian dalam telah didistribusikan ke setiap node menggunakan DISTSTYLE ALL.
- DS_DIST_NONE

  Tidak ada tabel yang didistribusikan kembali. Gabungan yang dikolokasikan dimungkinkan karena irisan yang sesuai digabungkan tanpa memindahkan data antar node.
- DS_DIST_INNER

  Tabel bagian dalam didistribusikan kembali.
- DS_DIST_LUAR

  Tabel luar didistribusikan kembali.
- DS_DIST_ALL_INNER

  Seluruh tabel bagian dalam didistribusikan kembali ke satu irisan karena tabel luar menggunakan DISTSTYLE ALL.
- DS_DIST_KEDUANYA

  Kedua tabel didistribusikan kembali.

Meninjau langkah-langkah rencana kueri

Anda dapat melihat langkah-langkah dalam rencana kueri dengan menjalankan perintah EXPLAIN. Contoh berikut menunjukkan query SQL dan menjelaskan output. Membaca rencana kueri dari bawah ke atas, Anda dapat melihat setiap operasi logis yang digunakan untuk melakukan kueri. Untuk informasi selengkapnya, lihat Membuat dan menafsirkan rencana kueri.

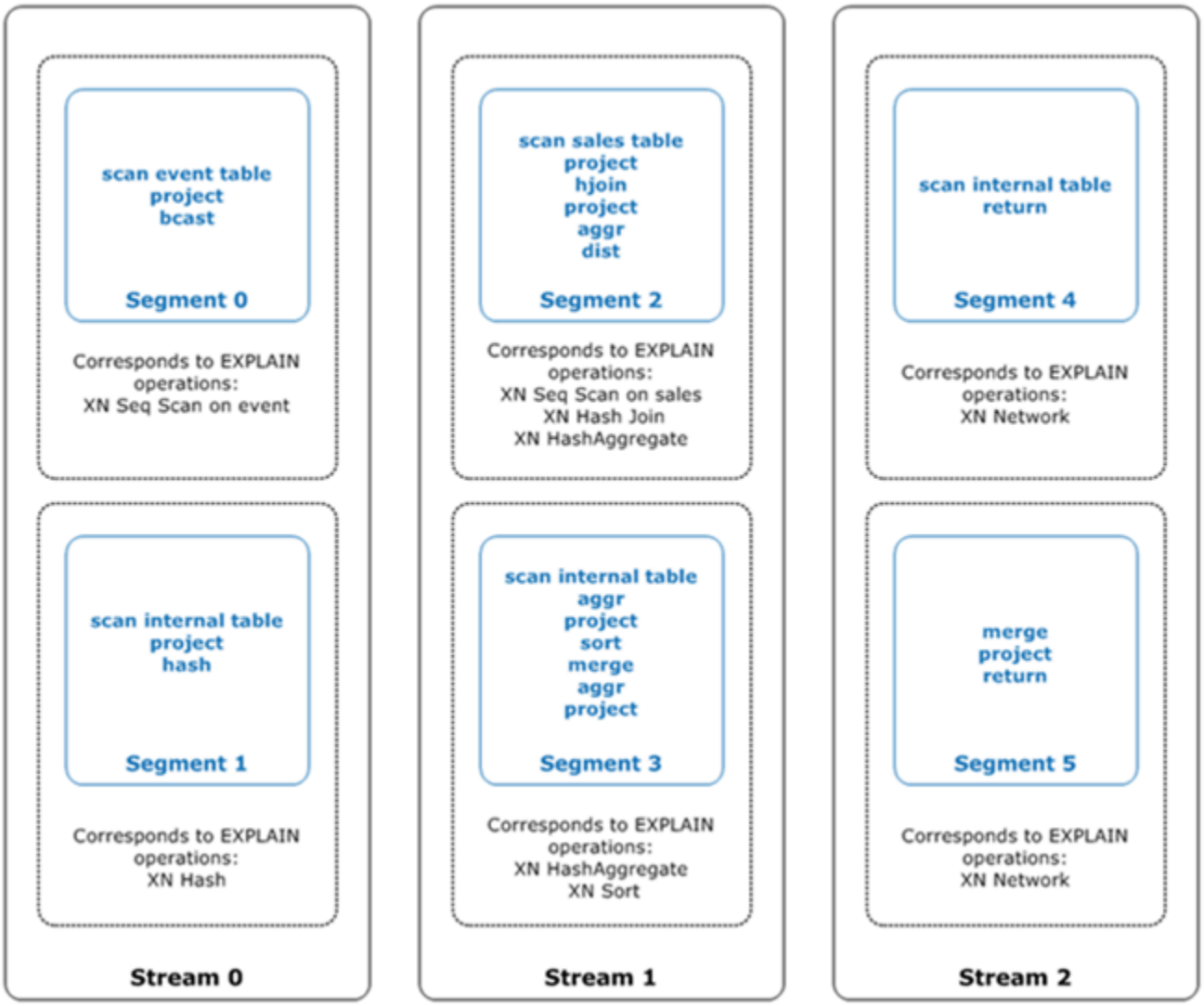
```
explain
select eventname, sum(pricepaid) from sales, event
where sales.eventid = event.eventid
```

```
group by eventname
order by 2 desc;
```

```
XN Merge  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
  Merge Key: sum(sales.pricepaid)
  ->  XN Network  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
        Send to leader
        ->  XN Sort  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
              Sort Key: sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=2815366577.07..2815366578.51 rows=576
 width=27)
                    ->  XN Hash Join DS_BCAST_INNER  (cost=109.98..2815365714.80
 rows=172456 width=27)
                          Hash Cond: ("outer".eventid = "inner".eventid)
                          ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456
 width=14)
                          ->  XN Hash  (cost=87.98..87.98 rows=8798 width=21)
                                ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798
 width=21)
```

Sebagai bagian dari pembuatan rencana kueri, pengoptimal kueri memecah rencana menjadi aliran, segmen, dan langkah-langkah. Pengoptimal kueri memecah rencana untuk mempersiapkan pendistribusian data dan beban kerja kueri ke node komputasi. Untuk informasi selengkapnya tentang aliran, segmen, dan langkah, lihatPerencanaan kueri dan alur kerja pelaksanaan.

Ilustrasi berikut menunjukkan kueri sebelumnya dan rencana kueri terkait. Ini menampilkan bagaimana operasi kueri yang terlibat dipetakan ke langkah-langkah yang digunakan Amazon Redshift untuk menghasilkan kode yang dikompilasi untuk irisan simpul komputasi. Setiap operasi rencana kueri memetakan ke beberapa langkah dalam segmen, dan terkadang ke beberapa segmen dalam aliran.



Dalam ilustrasi ini, pengoptimal kueri menjalankan rencana kueri sebagai berikut:

1. Di`Stream 0`, kueri berjalan `Segment 0` dengan operasi pemindaian berurutan untuk memindai `events` tabel. Kueri berlanjut `Segment 1` dengan operasi hash untuk membuat tabel hash untuk tabel bagian dalam di join.
2. Di`Stream 1`, kueri berjalan `Segment 2` dengan operasi pemindaian berurutan untuk memindai `sales` tabel. Ini berlanjut `Segment 2` dengan gabungan hash untuk bergabung dengan tabel di mana kolom gabungan bukan kunci distribusi dan kunci pengurutan. Sekali lagi dil `Segment 2` anjutkan dengan agregat hash untuk mengumpulkan hasil. Kemudian query berjalan `Segment 3` dengan operasi agregat hash untuk melakukan fungsi agregat yang dikelompokkan yang tidak disortir, dan operasi pengurutan untuk mengevaluasi klausa ORDER BY dan operasi pengurutan lainnya.

3. Di`Stream 2`, kueri menjalankan operasi jaringan di `Segment 4` dan `Segment 5` mengirim hasil perantara ke node pemimpin untuk diproses lebih lanjut.

Segmen terakhir dari kueri mengembalikan data. Jika set pengembalian dikumpulkan atau diurutkan, node komputasi masing-masing mengirimkan bagian hasil perantara mereka ke node pemimpin. Node pemimpin kemudian menggabungkan data sehingga hasil akhir dapat dikirim kembali ke klien yang meminta.

Untuk informasi selengkapnya tentang operator EXPLAIN, lihatEXPLAIN.

Faktor-faktor yang mempengaruhi kinerja kueri

Sejumlah faktor dapat mempengaruhi kinerja kueri. Aspek berikut dari operasi data, cluster, dan database Anda semua berperan dalam seberapa cepat proses kueri Anda.

- Jumlah node, prosesor, atau irisan — Node komputasi dipartisi menjadi irisan. Lebih banyak node berarti lebih banyak prosesor dan lebih banyak irisan, yang memungkinkan kueri Anda diproses lebih cepat dengan menjalankan bagian kueri secara bersamaan di seluruh irisan. Namun, lebih banyak node juga berarti biaya yang lebih besar, jadi Anda perlu menemukan keseimbangan biaya dan kinerja yang sesuai untuk sistem Anda. Untuk informasi selengkapnya tentang arsitektur cluster Amazon Redshift, lihatArsitektur sistem gudang data.
- Jenis node — Cluster Amazon Redshift dapat menggunakan salah satu dari beberapa jenis node. Setiap jenis node menawarkan ukuran dan batas yang berbeda untuk membantu Anda menskalakan cluster dengan tepat. Ukuran node menentukan kapasitas penyimpanan, memori, CPU, dan harga setiap node dalam cluster. Untuk informasi selengkapnya tentang jenis node, lihat Ring kasan cluster Amazon Redshift di Panduan Manajemen Amazon Redshift.
- Distribusi data — Amazon Redshift menyimpan data tabel pada node komputasi sesuai dengan gaya distribusi tabel. Saat Anda menjalankan kueri, pengoptimal kueri mendistribusikan kembali data ke node komputasi sesuai kebutuhan untuk melakukan gabungan dan agregasi apa pun. Memilih gaya distribusi yang tepat untuk tabel membantu meminimalkan dampak langkah redistribusi dengan menemukan data di tempat yang diperlukan sebelum gabungan dilakukan. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.
- Urutan pengurutan data — Amazon Redshift menyimpan data tabel pada disk dalam urutan yang diurutkan sesuai dengan kunci pengurutan tabel. Pengoptimal kueri dan prosesor kueri menggunakan informasi tentang di mana data berada untuk mengurangi jumlah blok yang perlu dipindai dan dengan demikian meningkatkan kecepatan kueri. Untuk informasi selengkapnya, lihat Urutkan kunci.

- Ukuran dataset — Volume data yang lebih tinggi dalam cluster dapat memperlambat kinerja kueri untuk kueri, karena lebih banyak baris perlu dipindai dan didistribusikan kembali. Anda dapat mengurangi efek ini dengan menyedot debu dan pengarsipan data secara teratur, dan dengan menggunakan predikat untuk membatasi kumpulan data kueri.
- Operasi bersamaan — Menjalankan beberapa operasi sekaligus dapat memengaruhi kinerja kueri. Setiap operasi mengambil satu atau lebih slot dalam antrian kueri yang tersedia dan menggunakan memori yang terkait dengan slot tersebut. Jika operasi lain sedang berjalan, slot antrian kueri yang cukup mungkin tidak tersedia. Dalam hal ini, kueri harus menunggu slot terbuka sebelum dapat mulai diproses. Untuk informasi selengkapnya tentang membuat dan mengonfigurasi antrian kueri, lihatManajemen beban kerja.
- Struktur kueri — Bagaimana kueri Anda ditulis memengaruhi kinerjanya. Sebisa mungkin, tulis kueri untuk memproses dan mengembalikan data sesedikit mungkin sesuai kebutuhan Anda. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang kueri.
- Kompilasi kode — Amazon Redshift menghasilkan dan mengkompilasi kode yang dioptimalkan untuk setiap rencana eksekusi kueri. Kode yang dikompilasi berjalan lebih cepat karena menghilangkan overhead menggunakan penerjemah. Untuk meminimalkan latensi untuk kueri baru sambil mempertahankan manfaat kinerja kode yang dikompilasi, Amazon Redshift menggunakan teknik yang disebut komposisi. Komposisi menghasilkan pengaturan ringan dari logika yang sudah ada sebelumnya untuk memproses kueri baru segera, sementara secara bersamaan mengkompilasi kode khusus kueri yang sangat dioptimalkan di latar belakang. Ini menghapus kompilasi dari jalur kritis eksekusi kueri, sehingga kueri baru dimulai lebih cepat dan memberikan kinerja yang konsisten dengan proses berikutnya.

  Amazon Redshift juga menggunakan layanan kompilasi tanpa server untuk menskalakan kompilasi kueri di luar sumber daya komputasi cluster Amazon Redshift. Segmen kode yang dikompilasi di-cache baik secara lokal di cluster dan dalam cache jarak jauh yang hampir tidak terbatas yang tetap ada setelah reboot cluster. Eksekusi berikutnya dari kueri yang sama berjalan lebih cepat karena mereka dapat melewati fase kompilasi. Dengan menggunakan layanan kompilasi yang dapat diskalakan, Amazon Redshift mengkompilasi kode secara paralel untuk memberikan kinerja cepat secara konsisten.

Analisis dan peningkatan kueri

Mengambil informasi dari gudang data Amazon Redshift melibatkan menjalankan kueri kompleks pada sejumlah besar data, yang dapat memakan waktu lama untuk diproses. Untuk memastikan

proses query secepat mungkin, ada sejumlah alat yang dapat Anda gunakan untuk mengidentifikasi potensi masalah kinerja.

Topik

- Alur kerja analisis kueri
- Meninjau peringatan kueri
- Menganalisis rencana kueri
- Menganalisis ringkasan kueri
- Peningkatan kinerja kueri
- Kueri diagnostik untuk penyetelan kueri

Alur kerja analisis kueri

Jika kueri membutuhkan waktu lebih lama dari yang diharapkan, gunakan langkah-langkah berikut untuk mengidentifikasi dan memperbaiki masalah yang mungkin berdampak negatif pada kinerja kueri. Jika Anda tidak yakin kueri apa di sistem Anda yang mungkin mendapat manfaat dari penyetelan kinerja, mulailah dengan menjalankan kueri diagnostik diMengidentifikasi kueri yang merupakan kandidat teratas untuk penyetelan.

1. Pastikan tabel Anda dirancang sesuai dengan praktik terbaik. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang tabel.
2. Lihat apakah Anda dapat menghapus atau mengarsipkan data yang tidak diperlukan di tabel Anda. Misalnya, kueri Anda selalu menargetkan data senilai 6 bulan terakhir tetapi Anda memiliki nilai 18 bulan terakhir di tabel Anda. Dalam hal ini, Anda dapat menghapus atau mengarsipkan data lama untuk mengurangi jumlah catatan yang harus dipindai dan didistribusikan.
3. Jalankan VAKUM perintah pada tabel dalam kueri untuk merebut kembali ruang dan mengurutkan ulang baris. Menjalankan VACUUM membantu jika wilayah yang tidak disortir besar dan kueri menggunakan kunci pengurutan dalam gabungan atau predikat.
4. J MENGANALISA alankan perintah pada tabel dalam kueri untuk memastikan bahwa statistik diperbarui. Menjalankan ANALYZE membantu jika salah satu tabel dalam kueri baru-baru ini banyak berubah ukurannya. Jika menjalankan perintah ANALYZE penuh akan memakan waktu terlalu lama, jalankan ANALYZE pada satu kolom untuk mengurangi waktu pemrosesan. Pendekatan ini masih memperbarui statistik ukuran tabel; ukuran tabel merupakan faktor penting dalam perencanaan kueri.

5. Pastikan kueri Anda telah dijalankan sekali untuk setiap jenis klien (berdasarkan jenis protokol koneksi yang digunakan klien) sehingga kueri dikompilasi dan di-cache. Pendekatan ini mempercepat proses kueri berikutnya. Untuk informasi selengkapnya, lihat Faktor-faktor yang mempengaruhi kinerja kueri.
6. Periksa STL_ALERT_EVENT_LOG tabel untuk mengidentifikasi dan memperbaiki kemungkinan masalah dengan kueri Anda. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.
7. Jalan EXPLAIN kan perintah untuk mendapatkan rencana kueri dan menggunakannya untuk mengoptimalkan kueri. Untuk informasi selengkapnya, lihat Menganalisis rencana kueri.
8. Gunakan SVL_QUERY_REPORT tampilan SVL_QUERY_SUMMARY dan untuk mendapatkan informasi ringkasan dan menggunakannya untuk mengoptimalkan kueri. Untuk informasi selengkapnya, lihat Menganalisis ringkasan kueri.

Terkadang kueri yang harus dijalankan dengan cepat dipaksa untuk menunggu sampai kueri lain yang berjalan lebih lama selesai. Dalam hal ini, Anda mungkin tidak perlu meningkatkan kueri itu sendiri, tetapi Anda dapat meningkatkan kinerja sistem secara keseluruhan dengan membuat dan menggunakan antrian kueri untuk berbagai jenis kueri. Untuk mendapatkan gambaran tentang waktu tunggu antrian untuk kueri Anda, lihatMeninjau waktu tunggu antrian untuk kueri. Untuk informasi selengkapnya tentang mengonfigurasi antrian kueri, lihatManajemen beban kerja.

Meninjau peringatan kueri

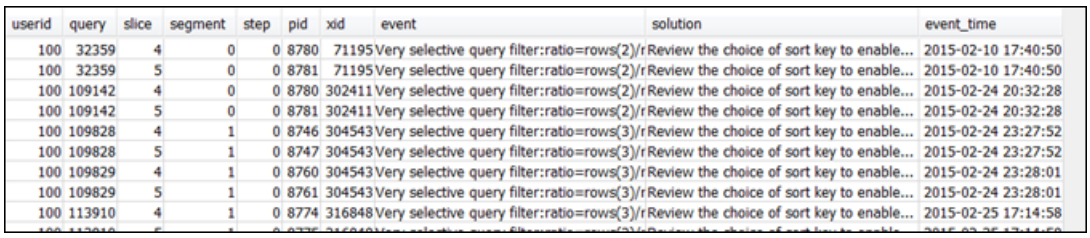
Untuk menggunakan tabel STL_ALERT_EVENT_LOG sistem untuk mengidentifikasi dan memperbaiki potensi masalah kinerja dengan kueri Anda, ikuti langkah-langkah berikut:

1. Jalankan yang berikut ini untuk menentukan ID kueri Anda:

```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

   Periksa teks kueri terpotong di `substring` bidang untuk menentukan `query` nilai mana yang akan dipilih. Jika Anda telah menjalankan kueri lebih dari sekali, gunakan `query` nilai dari baris dengan `elapsed` nilai yang lebih rendah. Itu adalah baris untuk versi yang dikompilasi. Jika Anda telah menjalankan banyak kueri, Anda dapat menaikkan nilai yang digunakan oleh klausa LIMIT yang digunakan untuk memastikan bahwa kueri Anda disertakan.
2. Pilih baris dari STL_ALERT_EVENT_LOG untuk kueri Anda:

```
Select * from stl_alert_event_log where query = MyQueryID;
```



userid	query	slice	segment	step	pid	xid	event	solution	event_time
100	32359	4	0	0	8780	71195	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-10 17:40:50
100	32359	5	0	0	8781	71195	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-10 17:40:50
100	109142	4	0	0	8780	302411	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-24 20:32:28
100	109142	5	0	0	8781	302411	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-24 20:32:28
100	109828	4	1	0	8746	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:27:52
100	109828	5	1	0	8747	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:27:52
100	109829	4	1	0	8760	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:28:01
100	109829	5	1	0	8761	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:28:01
100	113910	4	1	0	8774	316848	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-25 17:14:58

3. Evaluasi hasil untuk kueri Anda. Gunakan tabel berikut untuk menemukan solusi potensial untuk masalah apa pun yang telah Anda identifikasi.

> **Note**
>
> Tidak semua kueri memiliki baris di STL_ALERT_EVENT_LOG, hanya yang memiliki masalah yang teridentifikasi.

Isu	Nilai acara	Nilai solusi	Solusi yang disarankan
Statistik untuk tabel dalam kueri hilang atau kedaluwarsa.	Statistik perencana kueri hilang	Jalankan perintah ANALYZE	Lihat Statistik tabel hilang atau kedaluwarsa.
Ada gabungan loop bersarang (gabungan paling tidak optimal) dalam rencana kueri.	Bergabung dengan Loop Bersarang dalam rencana kueri	Tinjau predikat gabungan untuk menghindari produk Cartesian	Lihat Loop Bersarang.
Pemindaian melewatkan sejumlah besar baris yang ditandai sebagai dihapus tetapi tidak disedot, atau baris yang telah disisipkan tetapi tidak dikomite.	Memindai sejumlah besar baris yang dihapus	Jalankan perintah VACUUM untuk merebut kembali ruang yang dihapus	Lihat Baris hantu atau baris yang tidak diikat.

Isu	Nilai acara	Nilai solusi	Solusi yang disarankan
Lebih dari 1.000.000 baris didistribusikan kembali untuk gabungan hash atau agregasi.	Mendistribusikan sejumlah besar baris di seluruh jaringan: RowCount baris didistribusikan untuk memproses agregasi	Tinjau pilihan kunci distribusi untuk mengkolokasikan gabungan atau agregasi	Lihat Distribusi data yang kurang optimal.
Lebih dari 1.000.000 baris disiarkan untuk hash join.	Menyiarkan sejumlah besar baris di seluruh jaringan	Tinjau pilihan kunci distribusi untuk mengkolokasikan gabungan dan pertimbangkan untuk menggunakan tabel terdistribusi	Lihat Distribusi data yang kurang optimal.
Gaya redistribusi DS_DIST_ALL_INNER ditunjukkan dalam rencana kueri, yang memaksa eksekusi serial karena seluruh tabel bagian dalam didistribusikan kembali ke satu node.	DS_DIST_ALL_INNER untuk Hash Bergabung dalam rencana kueri	Tinjau pilihan strategi distribusi untuk mendistribusikan tabel bagian dalam, bukan luar	Lihat Distribusi data yang kurang optimal.

Menganalisis rencana kueri

Jalan EXPLAIN kan perintah untuk mendapatkan rencana kueri.

Sebelum menganalisis rencana kueri, Anda harus terbiasa dengan cara membacanya. Jika Anda tidak terbiasa membaca rencana kueri, kami sarankan Anda membaca Membuat dan menafsirkan rencana kueri sebelum melanjutkan.

Untuk menganalisis data yang disediakan oleh rencana kueri, ikuti langkah-langkah berikut:

1. Identifikasi langkah-langkah dengan biaya tertinggi. Berkonsentrasilah untuk mengoptimalkannya saat melanjutkan melalui langkah-langkah yang tersisa.
2. Lihatlah jenis gabungannya:
   - Loop Bersar ang: Gabungan seperti itu biasanya terjadi karena kondisi gabungan dihilangkan. Untuk solusi yang disarankan, lihatLoop Bersarang.
   - Hash dan Hash Join: Gabungan hash digunakan saat menggabungkan tabel di mana kolom gabungan bukan kunci distribusi dan juga bukan kunci pengurutan. Untuk solusi yang disarankan, lihatGabung hash.
   - Gabungkan Gabung: Tidak diperlukan perubahan.
3. Perhatikan tabel mana yang digunakan untuk sambungan dalam, dan mana untuk gabungan luar. Mesin kueri umumnya memilih tabel yang lebih kecil untuk gabungan dalam, dan tabel yang lebih besar untuk gabungan luar. Jika pilihan seperti itu tidak terjadi, statistik Anda kemungkinan sudah ketinggalan zaman. Untuk solusi yang disarankan, lihatStatistik tabel hilang atau kedaluwarsa.
4. Lihat apakah ada operasi pengurutan berbiaya tinggi. Jika ada, lihat Baris yang tidak disortir atau salah disortir solusi yang disarankan.
5. Cari operator siaran berikut di mana ada operasi berbiaya tinggi:
   - DS_BCAST_INNER: Menunjukkan bahwa tabel disiarkan ke semua node komputasi. Ini bagus untuk meja kecil, tetapi tidak ideal untuk meja yang lebih besar.
   - DS_DIST_ALL_INNER: Menunjukkan bahwa semua beban kerja ada pada satu irisan.
   - DS_DIST_BOTH: Menunjukkan redistribusi berat.

   Untuk solusi yang direkomendasikan untuk situasi ini, lihatDistribusi data yang kurang optimal.

Menganalisis ringkasan kueri

Untuk mendapatkan langkah-langkah eksekusi dan statistik secara lebih rinci daripada dalam rencana kueri EXPLAIN yang dihasilkan, gunakan SVL_QUERY_SUMMARY tampilan dan SVL_QUERY_REPORT sistem.

SVL_QUERY_SUMMARY menyediakan statistik kueri berdasarkan aliran. Anda dapat menggunakan informasi yang diberikannya untuk mengidentifikasi masalah dengan langkah-langkah mahal, langkah yang berjalan lama, dan langkah-langkah yang menulis ke disk.

Dengan tampilan sistem SVL_QUERY_REPORT, Anda dapat melihat informasi yang mirip dengan SVL_QUERY_SUMMARY, hanya dengan irisan simpul komputasi daripada dengan aliran. Anda dapat menggunakan informasi tingkat iris untuk mendeteksi distribusi data yang tidak merata di seluruh cluster (juga dikenal sebagai kemiringan distribusi data), yang memaksa beberapa node melakukan lebih banyak pekerjaan daripada yang lain dan merusak kinerja kueri.

Topik

- Menggunakan tampilan SVL_QUERY_SUMMARY
- Menggunakan tampilan SVL_QUERY_REPORT
- Memetakan rencana kueri ke ringkasan kueri

Menggunakan tampilan SVL_QUERY_SUMMARY

Untuk menganalisis informasi ringkasan kueri berdasarkan aliran menggunakanSVL_QUERY_SUMMARY, lakukan hal berikut:

1. Jalankan kueri berikut untuk menentukan ID kueri Anda:

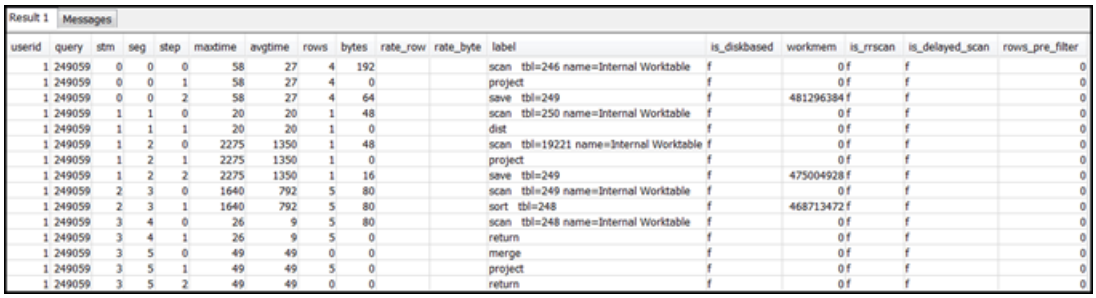
   ```
   select query, elapsed, substring
   from svl_qlog
   order by query
   desc limit 5;
   ```

   Periksa teks kueri terpotong di `substring` bidang untuk menentukan `query` nilai mana yang mewakili kueri Anda. Jika Anda telah menjalankan kueri lebih dari sekali, gunakan `query` nilai dari baris dengan `elapsed` nilai yang lebih rendah. Itu adalah baris untuk versi yang dikompilasi. Jika Anda telah menjalankan banyak kueri, Anda dapat menaikkan nilai yang digunakan oleh klausa LIMIT yang digunakan untuk memastikan bahwa kueri Anda disertakan.

2. Pilih baris dari SVL_QUERY_SUMMARY untuk kueri Anda. Urutkan hasil berdasarkan aliran, segmen, dan langkah:

   ```
   select * from svl_query_summary where query = MyQueryID order by stm, seg, step;
   ```

Berikut ini adalah contoh hasil.



Result 1 Messages

userid	query	stm	seg	step	maxtime	avgtime	rows	bytes	rate_row	rate_byte	label	is_diskbased	workmem	is_rrscan	is_delayed_scan	rows_pre_filter
1	249059	0	0	0	58	27	4	192			scan tbl=246 name=Internal Worktable	f	0	f	f	0
1	249059	0	0	1	58	27	4	0			project	f	0	f	f	0
1	249059	0	0	2	58	27	4	64			save tbl=249	f	481296384	f	f	0
1	249059	1	1	0	20	20	1	48			scan tbl=250 name=Internal Worktable	f	0	f	f	0
1	249059	1	1	1	20	20	1	0			dist	f	0	f	f	0
1	249059	1	2	0	2275	1350	1	48			scan tbl=19221 name=Internal Worktable	f	0	f	f	0
1	249059	1	2	1	2275	1350	1	0			project	f	0	f	f	0
1	249059	1	2	2	2275	1350	1	16			save tbl=249	f	475004928	f	f	0
1	249059	2	3	0	1640	792	5	80			scan tbl=249 name=Internal Worktable	f	0	f	f	0
1	249059	2	3	1	1640	792	5	80			sort tbl=248	f	468713472	f	f	0
1	249059	3	4	0	26	9	5	80			scan tbl=248 name=Internal Worktable	f	0	f	f	0
1	249059	3	4	1	26	9	5	0			return	f	0	f	f	0
1	249059	3	5	0	49	49	0	0			merge	f	0	f	f	0
1	249059	3	5	1	49	49	5	0			project	f	0	f	f	0
1	249059	3	5	2	49	49	0	0			return	f	0	f	f	0

3. Petakan langkah-langkah ke operasi dalam rencana kueri menggunakan informasi diMemetakan rencana kueri ke ringkasan kueri. Mereka harus memiliki nilai yang kira-kira sama untuk baris dan byte (baris * lebar dari rencana kueri). Jika tidak, lihat solusi Statistik tabel hilang atau kedaluwarsa yang disarankan.
4. Lihat apakah `is_diskbased` bidang memiliki nilai `t` (true) untuk langkah apa pun. Hash, agregat, dan sort adalah operator yang cenderung menulis data ke disk jika sistem tidak memiliki cukup memori yang dialokasikan untuk pemrosesan kueri.

   Jika `is_diskbased` benar, lihat Memori yang tidak mencukupi yang dialokasikan untuk kueri solusi yang disarankan.
5. Tinjau nilai `label` bidang dan lihat apakah ada AGG-DIST-AGG urutan di mana pun dalam langkah-langkah. Kehadirannya menunjukkan agregasi dua langkah, yang mahal. Untuk memperbaikinya, ubah klausa GROUP BY untuk menggunakan kunci distribusi (kunci pertama, jika ada beberapa).
6. Tinjau `maxtime` nilai untuk setiap segmen (sama di semua langkah di segmen). Identifikasi segmen dengan `maxtime` nilai tertinggi dan tinjau langkah-langkah di segmen ini untuk operator berikut.

> **Note**
>
> `maxtime`Nilai tinggi tidak selalu menunjukkan masalah dengan segmen. Meskipun nilainya tinggi, segmen ini mungkin tidak membutuhkan waktu lama untuk diproses. Semua segmen dalam aliran mulai diatur waktunya secara serempak. Namun, beberapa segmen hilir mungkin tidak dapat berjalan sampai mereka mendapatkan data dari segmen hulu. Efek ini mungkin membuat mereka tampaknya membutuhkan waktu lama karena nil `maxtime` ainya mencakup waktu tunggu dan waktu pemrosesan mereka.

- BCAST atau DIST: Dalam kasus ini, `maxtime` nilai tinggi mungkin merupakan hasil dari mendistribusikan kembali sejumlah besar baris. Untuk solusi yang disarankan, lihatDistribusi data yang kurang optimal.
- HJOIN (hash join): Jika langkah yang dimaksud memiliki nilai yang sangat tinggi di `rows` bidang dibandingkan dengan `rows` nilai pada langkah RETURN terakhir dalam kueri, lihat Gabung hash solusi yang disarankan.
- SCAN/SORT: Cari urutan langkah SCAN, SORT, SCAN, MERGE tepat sebelum langkah bergabung. Pola ini menunjukkan bahwa data yang tidak disortir sedang dipindai, diurutkan, dan kemudian digabungkan dengan area tabel yang diurutkan.

  Lihat apakah nilai baris untuk langkah SCAN memiliki nilai yang sangat tinggi dibandingkan dengan nilai baris pada langkah RETURN terakhir dalam kueri. Pola ini menunjukkan bahwa mesin eksekusi memindai baris yang kemudian dibuang, yang tidak efisien. Untuk solusi yang disarankan, lihatPredikat yang tidak cukup membatasi.

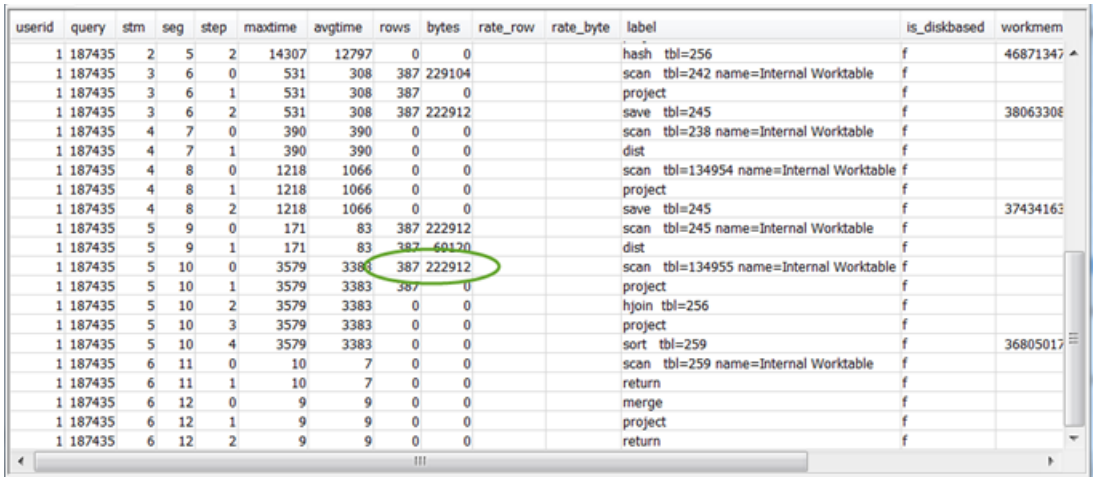
  Jika `maxtime` nilai untuk langkah SCAN tinggi, lihat Klausa WHERE yang tidak optimal solusi yang disarankan.

  Jika `rows` nilai untuk langkah SORT bukan nol, lihat Baris yang tidak disortir atau salah disortir solusi yang disarankan.

7. Tinjau `bytes` nilai `rows` dan untuk 5-10 langkah yang mendahului langkah RETURN terakhir untuk mendapatkan gambaran tentang jumlah data yang dikembalikan ke klien. Proses ini bisa menjadi sedikit seni.

   Misalnya, dalam ringkasan kueri contoh berikut, langkah PROJECT ketiga memberikan `rows` nilai, tetapi bukan `bytes` nilai. Dengan melihat langkah-langkah sebelumnya untuk satu dengan `rows` nilai yang sama, Anda menemukan langkah SCAN yang menyediakan informasi baris dan byte.

   Berikut ini adalah hasil sampel.



userid	query	stm	seg	step	maxtime	avgtime	rows	bytes	rate_row	rate_byte	label	is_diskbased	workmem
1	187435	2	5	2	14307	12797	0	0			hash tbl=256	f	46871347
1	187435	3	6	0	531	308	387	229104			scan tbl=242 name=Internal Worktable	f	
1	187435	3	6	1	531	308	387	0			project	f	
1	187435	3	6	2	531	308	387	222912			save tbl=245	f	38063308
1	187435	4	7	0	390	390	0	0			scan tbl=238 name=Internal Worktable	f	
1	187435	4	7	1	390	390	0	0			dist	f	
1	187435	4	8	0	1218	1066	0	0			scan tbl=134954 name=Internal Worktable	f	
1	187435	4	8	1	1218	1066	0	0			project	f	
1	187435	4	8	2	1218	1066	0	0			save tbl=245	f	37434163
1	187435	5	9	0	171	83	387	222912			scan tbl=245 name=Internal Worktable	f	
1	187435	5	9	1	171	83	387	60120			dist	f	
1	187435	5	10	0	3579	3383	387	222912			scan tbl=134955 name=Internal Worktable	f	
1	187435	5	10	1	3579	3383	387	0			project	f	
1	187435	5	10	2	3579	3383	0	0			hjoin tbl=256	f	
1	187435	5	10	3	3579	3383	0	0			project	f	
1	187435	5	10	4	3579	3383	0	0			sort tbl=259	f	36805017
1	187435	6	11	0	10	7	0	0			scan tbl=259 name=Internal Worktable	f	
1	187435	6	11	1	10	7	0	0			return	f	
1	187435	6	12	0	9	9	0	0			merge	f	
1	187435	6	12	1	9	9	0	0			project	f	
1	187435	6	12	2	9	9	0	0			return	f	

Jika Anda mengembalikan volume data yang luar biasa besar, lihat Set hasil yang sangat besar solusi yang disarankan.

8. Lihat apakah nil `bytes` ainya relatif tinggi terhadap `rows` nilai untuk langkah apa pun, dibandingkan dengan langkah lainnya. Pola ini dapat menunjukkan bahwa Anda memilih banyak kolom. Untuk solusi yang disarankan, lihatDaftar SELECT besar.

Menggunakan tampilan SVL_QUERY_REPORT

Untuk menganalisis informasi ringkasan kueri berdasarkan irisan menggunakanSVL_QUERY_REPORT, lakukan hal berikut:

1. Jalankan yang berikut ini untuk menentukan ID kueri Anda:

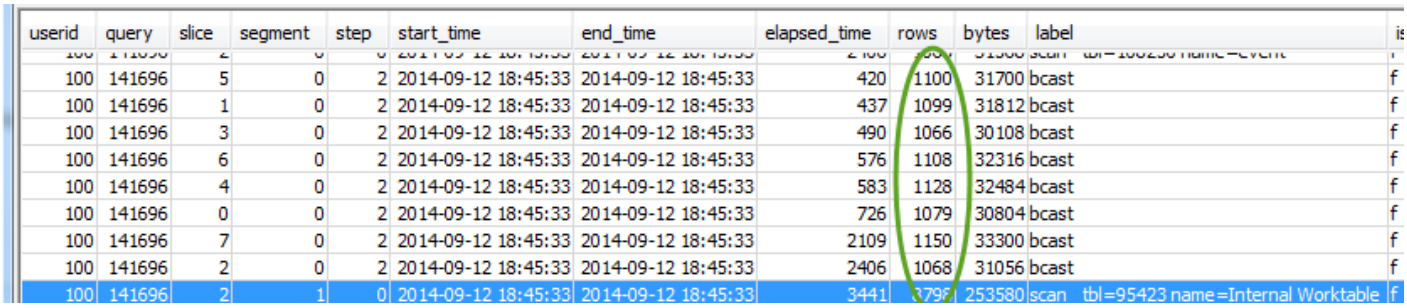
```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

Periksa teks kueri terpotong di `substring` bidang untuk menentukan `query` nilai mana yang mewakili kueri Anda. Jika Anda telah menjalankan kueri lebih dari sekali, gunakan `query` nilai dari baris dengan `elapsed` nilai yang lebih rendah. Itu adalah baris untuk versi yang dikompilasi. Jika Anda telah menjalankan banyak kueri, Anda dapat menaikkan nilai yang digunakan oleh klausa LIMIT yang digunakan untuk memastikan bahwa kueri Anda disertakan.

2. Pilih baris dari SVL_QUERY_REPORT untuk kueri Anda. Urutkan hasil berdasarkan segmen, langkah, elapsed_time, dan baris:

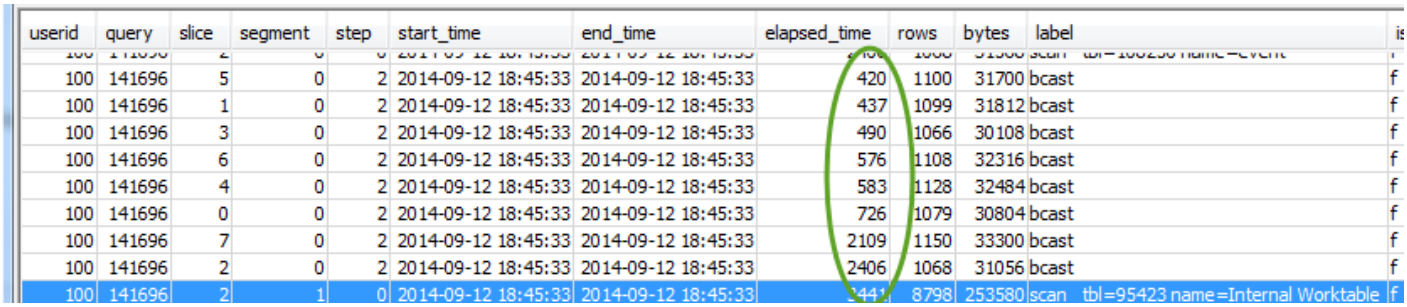
```
select * from svl_query_report where query = MyQueryID order by segment, step,
 elapsed_time, rows;
```

3. Untuk setiap langkah, periksa untuk melihat bahwa semua irisan memproses kira-kira jumlah baris yang sama:



userid	query	slice	segment	step	start_time	end_time	elapsed_time	rows	bytes	label	is
100	141696	5	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	420	1100	31700	bcast	f
100	141696	1	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	437	1099	31812	bcast	f
100	141696	3	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	490	1066	30108	bcast	f
100	141696	6	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	576	1108	32316	bcast	f
100	141696	4	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	583	1128	32484	bcast	f
100	141696	0	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	726	1079	30804	bcast	f
100	141696	7	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	bcast	f
100	141696	2	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2406	1068	31056	bcast	f
100	141696	2	1	0	2014-09-12 18:45:33	2014-09-12 18:45:33	3441	8798	253580	scan tbl=95423 name=Internal Worktable	f

Periksa juga untuk melihat bahwa semua irisan membutuhkan waktu yang kira-kira sama:



userid	query	slice	segment	step	start_time	end_time	elapsed_time	rows	bytes	label	is
100	141696	5	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	420	1100	31700	bcast	f
100	141696	1	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	437	1099	31812	bcast	f
100	141696	3	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	490	1066	30108	bcast	f
100	141696	6	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	576	1108	32316	bcast	f
100	141696	4	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	583	1128	32484	bcast	f
100	141696	0	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	726	1079	30804	bcast	f
100	141696	7	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	bcast	f
100	141696	2	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2406	1068	31056	bcast	f
100	141696	2	1	0	2014-09-12 18:45:33	2014-09-12 18:45:33	3441	8798	253580	scan tbl=95423 name=Internal Worktable	f

Perbedaan besar dalam nilai-nilai ini dapat menunjukkan kemiringan distribusi data karena gaya distribusi yang kurang optimal untuk kueri khusus ini. Untuk solusi yang disarankan, lihatDistribusi data yang kurang optimal.

Memetakan rencana kueri ke ringkasan kueri

Saat menganalisis ringkasan kueri, Anda bisa mendapatkan detail lebih lanjut dengan memetakan operasi dari rencana kueri ke langkah-langkah (diidentifikasi oleh nilai bidang label) dalam ringkasan kueri. Tabel berikut memetakan operasi rencana kueri ke langkah-langkah ringkasan kueri.

Operasi rencana kueri	Nilai bidang label	Deskripsi
Agregat HashAggregate	AGGR	Mengevaluasi fungsi agregat dan kondisi GROUP BY.

Operasi rencana kueri	Nilai bidang label	Deskripsi
GroupAggregate		
DS_BCAST_INNER	BCAST (siaran)	Menyiarkan seluruh tabel atau beberapa set baris (seperti kumpulan baris yang difilter dari tabel) ke semua node.
Tidak muncul dalam rencana kueri	DELETE	Menghapus baris dari tabel.
DS_DIST_NONE DS_DIST_ALL_NONE DS_DIST_INNER DS_DIST_ALL_INNER DS_DIST_SEMUA_KEDUANYA	DIST (mendistribusikan)	Mendistribusikan baris ke node untuk tujuan penggabungan paralel atau pemrosesan paralel lainnya.
HASH	HASH	Membangun tabel hash untuk digunakan dalam gabungan hash.
Gabung Hash	HJOIN (gabungan hash)	Melakukan gabungan hash dari dua tabel atau set hasil perantara.
Tidak muncul dalam rencana kueri	INSERT	Menyisipkan baris ke dalam tabel.
Kuota	LIMIT	Menerapkan klausa LIMIT ke set hasil.
Gabungkan	MERGE	Menggabungkan baris yang berasal dari operasi pengurutan paralel atau gabungan.

Operasi rencana kueri	Nilai bidang label	Deskripsi
Gabungkan Gabung	MJOIN (gabungkan bergabung)	Melakukan gabungan gabungan dari dua tabel atau kumpulan hasil perantara.
Lingkaran bersarang	NLOOP (loop bersarang)	Melakukan penggabungan loop bersarang dari dua tabel atau kumpulan hasil perantara .
Tidak muncul dalam rencana kueri	MENGURAI	Mengurai string menjadi nilai biner untuk memuat.
Proyek	PROYEK	Mengevaluasi ekspresi.
Jaringan	KEMBALI	Mengembalikan baris ke pemimpin atau klien.
Tidak muncul dalam rencana kueri	MENYIMPAN	Mewujudkan baris untuk digunakan pada langkah pemrosesan berikutnya.
Seq Scan	MEMINDAI	Memindai tabel atau set hasil perantara.
Urutkan	SORT	Mengurutkan baris atau kumpulan hasil perantara seperti yang dipersyaratkan oleh operasi berikutnya lainnya (seperti gabungan atau agregasi) atau untuk memenuhi klausa ORDER BY.
Unik	UNIK	Menerapkan klausa SELECT DISTINCT atau menghapus duplikat seperti yang dipersyar atkan oleh operasi lain.

Operasi rencana kueri	Nilai bidang label	Deskripsi
Jendela	JENDELA	Menghitung fungsi jendela agregat dan peringkat.

Peningkatan kinerja kueri

Berikut ini adalah beberapa masalah umum yang memengaruhi kinerja kueri Amazon Redshift, dengan petunjuk tentang cara mendiagnosis dan menyelesaikannya.

Topik

- Statistik tabel hilang atau kedaluwarsa
- Loop Bersarang
- Gabung hash
- Baris hantu atau baris yang tidak diikat
- Baris yang tidak disortir atau salah disortir
- Distribusi data yang kurang optimal
- Memori yang tidak mencukupi yang dialokasikan untuk kueri
- Klausa WHERE yang tidak optimal
- Predikat yang tidak cukup membatasi
- Set hasil yang sangat besar
- Daftar SELECT besar

Statistik tabel hilang atau kedaluwarsa

Jika statistik tabel hilang atau kedaluwarsa, Anda mungkin melihat yang berikut:

- Pesan peringatan dalam hasil perintah EXPLAIN.
- Peristiwa peringatan statistik yang hilang di STL_ALERT_EVENT_LOG. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.

Untuk memperbaiki masalah ini, jalankanMENGANALISA.

Loop Bersarang

Jika loop bersarang ada, Anda mungkin melihat peristiwa peringatan loop bersarang di STL_ALERT_EVENT_LOG. Anda juga dapat mengidentifikasi jenis acara ini dengan menjalankan kueri diMengidentifikasi kueri dengan loop bersarang. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.

Untuk memperbaikinya, tinjau kueri Anda untuk cross-join dan hapus jika memungkinkan. Cross-joins adalah gabungan tanpa kondisi gabungan yang menghasilkan produk Cartesian dari dua tabel. Mereka biasanya dijalankan sebagai gabungan loop bersarang, yang merupakan jenis gabungan yang paling lambat.

Gabung hash

Jika hash join hadir, Anda mungkin melihat yang berikut ini:

- Operasi gabungan hash dan hash dalam rencana kueri. Untuk informasi selengkapnya, lihat Menganalisis rencana kueri.
- Langkah HJOIN di segmen dengan nilai maxtime tertinggi di SVL_QUERY_SUMMARY. Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_SUMMARY.

Untuk memperbaiki masalah ini, Anda dapat mengambil beberapa pendekatan:

- Tulis ulang kueri untuk menggunakan gabungan gabungan jika memungkinkan. Anda dapat melakukan ini dengan menentukan kolom gabungan yang merupakan kunci distribusi dan kunci pengurutan.
- Jika langkah HJOIN di SVL_QUERY_SUMMARY memiliki nilai yang sangat tinggi di bidang baris dibandingkan dengan nilai baris pada langkah RETURN terakhir dalam kueri, periksa apakah Anda dapat menulis ulang kueri untuk bergabung pada kolom unik. Ketika kueri tidak bergabung pada kolom unik, seperti kunci primer, itu meningkatkan jumlah baris yang terlibat dalam gabungan.

Baris hantu atau baris yang tidak diikat

Jika ada baris hantu atau baris yang tidak diikat, Anda mungkin melihat peristiwa peringatan di STL_ALERT_EVENT_LOG yang menunjukkan baris hantu yang berlebihan. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.

Untuk memperbaiki masalah ini, Anda dapat mengambil beberapa pendekatan:

- Periksa tab Beban di konsol Amazon Redshift Anda untuk operasi pemuatan aktif pada salah satu tabel kueri. Jika Anda melihat operasi pemuatan aktif, tunggu hingga selesai sebelum mengambil tindakan.
- Jika tidak ada operasi pemuatan aktif, jalankan VAKUM pada tabel kueri untuk menghapus baris yang dihapus.

Baris yang tidak disortir atau salah disortir

Jika ada baris yang tidak disortir atau salah disortir, Anda mungkin melihat peristiwa peringatan filter yang sangat selektif di STL_ALERT_EVENT_LOG. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.

Anda juga dapat memeriksa untuk melihat apakah ada tabel dalam kueri Anda yang memiliki area besar yang tidak disortir dengan menjalankan kueri diMengidentifikasi tabel dengan data miring atau baris yang tidak disortir.

Untuk memperbaiki masalah ini, Anda dapat mengambil beberapa pendekatan:

- Jalan VAKUM kan pada tabel kueri untuk mengurutkan ulang baris.
- Tinjau kunci pengurutan pada tabel kueri untuk melihat apakah ada perbaikan yang dapat dilakukan. Ingatlah untuk menimbang kinerja kueri ini terhadap kinerja kueri penting lainnya dan sistem secara keseluruhan sebelum membuat perubahan apa pun. Untuk informasi selengkapnya, lihat Urutkan kunci.

Distribusi data yang kurang optimal

Jika distribusi data kurang optimal, Anda mungkin melihat yang berikut:

- Eksekusi serial, siaran besar, atau peristiwa peringatan distribusi besar muncul di STL_ALERT_EVENT_LOG. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.
- Irisan tidak memproses kira-kira jumlah baris yang sama untuk langkah tertentu. Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_REPORT.
- Irisan tidak memakan waktu yang kira-kira sama untuk langkah tertentu. Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_REPORT.

Jika tidak ada yang sebelumnya benar, Anda juga dapat melihat apakah salah satu tabel dalam kueri Anda memiliki kemiringan data dengan menjalankan kueri diMengidentifikasi tabel dengan data miring atau baris yang tidak disortir.

Untuk memperbaiki masalah ini, tinjau gaya distribusi untuk tabel dalam kueri dan lihat apakah ada perbaikan yang dapat dilakukan. Ingatlah untuk menimbang kinerja kueri ini terhadap kinerja kueri penting lainnya dan sistem secara keseluruhan sebelum membuat perubahan apa pun. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

Memori yang tidak mencukupi yang dialokasikan untuk kueri

Jika memori tidak mencukupi dialokasikan untuk kueri Anda, Anda mungkin melihat langkah di SVL_QUERY_SUMMARY yang memiliki nilai true. `is_diskbased` Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_SUMMARY.

Untuk memperbaiki masalah ini, alokasikan lebih banyak memori ke kueri dengan meningkatkan sementara jumlah slot kueri yang digunakannya. Manajemen Beban Kerja (WLM) mencadangkan slot dalam antrian kueri yang setara dengan tingkat konkurensi yang ditetapkan untuk antrian. Misalnya, antrian dengan tingkat konkurensi 5 memiliki 5 slot. Memori yang ditetapkan ke antrian dialokasikan secara merata untuk setiap slot. Menetapkan beberapa slot ke satu kueri memberikan akses kueri itu ke memori untuk semua slot tersebut. Untuk informasi selengkapnya tentang cara meningkatkan sementara slot untuk kueri, lihatwlm_query_slot_count.

Klausa WHERE yang tidak optimal

Jika klausa WHERE menyebabkan pemindaian tabel berlebihan, Anda mungkin melihat langkah SCAN di segmen dengan `maxtime` nilai tertinggi di SVL_QUERY_SUMMARY. Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_SUMMARY.

Untuk memperbaiki masalah ini, tambahkan klausa WHERE ke kueri berdasarkan kolom jenis utama dari tabel terbesar. Pendekatan ini membantu meminimalkan waktu pemindaian. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang tabel.

Predikat yang tidak cukup membatasi

Jika kueri Anda memiliki predikat terbatas yang tidak cukup, Anda mungkin melihat langkah SCAN di segmen dengan `maxtime` nilai tertinggi di SVL_QUERY_SUMMARY yang memiliki nilai sangat tinggi dibandingkan dengan `rows` nilai pada langkah RETURN terakhir `rows` dalam kueri. Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_SUMMARY.

Untuk memperbaiki masalah ini, coba tambahkan predikat ke kueri atau membuat predikat yang ada lebih membatasi untuk mempersempit output.

Set hasil yang sangat besar

Jika kueri Anda mengembalikan kumpulan hasil yang sangat besar, pertimbangkan untuk menulis ulang kueri MEMBONGKAR untuk digunakan untuk menulis hasil ke Amazon S3. Pendekatan ini meningkatkan kinerja langkah RETURN dengan mengambil keuntungan dari pemrosesan paralel. Untuk informasi selengkapnya tentang memeriksa kumpulan hasil yang sangat besar, lihatMenggunakan tampilan SVL_QUERY_SUMMARY.

Daftar SELECT besar

Jika kueri Anda memiliki daftar SELECT yang luar biasa besar, Anda mungkin melihat `bytes` nilai yang relatif tinggi terhadap `rows` nilai untuk setiap langkah (dibandingkan dengan langkah lain) di SVL_QUERY_SUMMARY. `bytes`Nilai tinggi ini dapat menjadi indikator bahwa Anda memilih banyak kolom. Untuk informasi selengkapnya, lihat Menggunakan tampilan SVL_QUERY_SUMMARY.

Untuk memperbaiki masalah ini, tinjau kolom yang Anda pilih dan lihat apakah ada yang dapat dihapus.

Kueri diagnostik untuk penyetelan kueri

Gunakan kueri berikut untuk mengidentifikasi masalah dengan kueri atau tabel dasar yang dapat memengaruhi kinerja kueri. Sebaiknya gunakan kueri ini dengan proses penyetelan kueri yang dibahas diAnalisis dan peningkatan kueri.

> **Note**
>
> Kueri ini untuk cluster yang disediakan Amazon Redshift. Kueri ini tidak untuk digunakan dengan kelompok kerja Redshift Serverless.

Topik

- Mengidentifikasi kueri yang merupakan kandidat teratas untuk penyetelan
- Mengidentifikasi tabel dengan data miring atau baris yang tidak disortir
- Mengidentifikasi kueri dengan loop bersarang
- Meninjau waktu tunggu antrian untuk kueri
- Meninjau peringatan kueri berdasarkan tabel

- Mengidentifikasi tabel dengan statistik yang hilang

Mengidentifikasi kueri yang merupakan kandidat teratas untuk penyetelan

Kueri berikut mengidentifikasi 50 pernyataan paling memakan waktu yang telah dijalankan dalam 7 hari terakhir. Anda dapat menggunakan hasil untuk mengidentifikasi kueri yang membutuhkan waktu yang sangat lama. Anda juga dapat mengidentifikasi kueri yang sering dijalankan (yang muncul lebih dari sekali dalam kumpulan hasil). Kueri ini sering kali merupakan kandidat yang baik untuk penyetelan untuk meningkatkan kinerja sistem.

Kueri ini juga menyediakan hitungan peristiwa peringatan yang terkait dengan setiap kueri yang diidentifikasi. Peringatan ini memberikan detail yang dapat Anda gunakan untuk meningkatkan kinerja kueri. Untuk informasi selengkapnya, lihat Meninjau peringatan kueri.

```
select trim(database) as db, count(query) as n_qry,
max(substring (qrytext,1,80)) as qrytext,
min(run_minutes) as "min" ,
max(run_minutes) as "max",
avg(run_minutes) as "avg", sum(run_minutes) as total,
max(query) as max_query_id,
max(starttime)::date as last_run,
sum(alerts) as alerts, aborted
from (select userid, label, stl_query.query,
trim(database) as database,
trim(querytxt) as qrytext,
md5(trim(querytxt)) as qry_md5,
starttime, endtime,
(datediff(seconds, starttime,endtime)::numeric(12,2))/60 as run_minutes,
alrt.num_events as alerts, aborted
from stl_query
left outer join
(select query, 1 as num_events from stl_alert_event_log group by query ) as alrt
on alrt.query = stl_query.query
where userid <> 1 and starttime >=  dateadd(day, -7, current_date))
group by database, label, qry_md5, aborted
order by total desc limit 50;
```

Mengidentifikasi tabel dengan data miring atau baris yang tidak disortir

Kueri berikut mengidentifikasi tabel yang memiliki distribusi data yang tidak merata (kemiringan data) atau persentase baris yang tidak disortir yang tinggi.

`skew`Nilai rendah menunjukkan bahwa data tabel didistribusikan dengan benar. Jika tabel memiliki `skew` nilai 4,00 atau lebih tinggi, pertimbangkan untuk memodifikasi gaya distribusi datanya. Untuk informasi selengkapnya, lihat Distribusi data yang kurang optimal.

Jika tabel memiliki `pct_unsorted` nilai lebih dari 20 persen, pertimbangkan untuk menjalankan VAKUM perintah. Untuk informasi selengkapnya, lihat Baris yang tidak disortir atau salah disortir.

Juga tinj `mbytes` au `pct_of_total` nilai dan untuk setiap tabel. Kolom ini mengidentifikasi ukuran tabel dan berapa persentase ruang disk mentah yang dikonsumsi tabel. Ruang disk mentah mencakup ruang yang disediakan oleh Amazon Redshift untuk penggunaan internal, sehingga lebih besar dari kapasitas disk nominal, yang merupakan jumlah ruang disk yang tersedia untuk pengguna. Gunakan informasi ini untuk memverifikasi bahwa Anda memiliki ruang disk kosong setidaknya 2,5 kali ukuran tabel terbesar Anda. Memiliki ruang ini tersedia memungkinkan sistem untuk menulis hasil perantara ke disk saat memproses kueri kompleks.

```
select trim(pgn.nspname) as schema,
trim(a.name) as table, id as tableid,
decode(pgc.reldiststyle,0, 'even',1,det.distkey ,8,'all') as distkey,
 dist_ratio.ratio::decimal(10,4) as skew,
det.head_sort as "sortkey",
det.n_sortkeys as "#sks", b.mbytes,
decode(b.mbytes,0,0,((b.mbytes/part.total::decimal)*100)::decimal(5,2)) as
 pct_of_total,
decode(det.max_enc,0,'n','y') as enc, a.rows,
decode( det.n_sortkeys, 0, null, a.unsorted_rows ) as unsorted_rows ,
decode( det.n_sortkeys, 0, null, decode( a.rows,0,0, (a.unsorted_rows::decimal(32)/
a.rows)*100) )::decimal(5,2) as pct_unsorted
from (select db_id, id, name, sum(rows) as rows,
sum(rows)-sum(sorted_rows) as unsorted_rows
from stv_tbl_perm a
group by db_id, id, name) as a
join pg_class as pgc on pgc.oid = a.id
join pg_namespace as pgn on pgn.oid = pgc.relnamespace
left outer join (select tbl, count(*) as mbytes
from stv_blocklist group by tbl) b on a.id=b.tbl
inner join (select attrelid,
min(case attisdistkey when 't' then attname else null end) as "distkey",
min(case attsortkeyord when 1 then attname  else null end ) as head_sort ,
max(attsortkeyord) as n_sortkeys,
max(attencodingtype) as max_enc
from pg_attribute group by 1) as det
on det.attrelid = a.id
```

```
inner join ( select tbl, max(mbytes)::decimal(32)/min(mbytes) as ratio
from (select tbl, trim(name) as name, slice, count(*) as mbytes
from svv_diskusage group by tbl, name, slice )
group by tbl, name ) as dist_ratio on a.id = dist_ratio.tbl
join ( select sum(capacity) as  total
from stv_partitions where part_begin=0 ) as part on 1=1
where mbytes is not null
order by  mbytes desc;
```

Mengidentifikasi kueri dengan loop bersarang

Kueri berikut mengidentifikasi kueri yang telah mencatat peristiwa peringatan untuk loop bersarang. Untuk informasi tentang cara memperbaiki kondisi loop bersarang, lihatLoop Bersarang.

```
select query, trim(querytxt) as SQL, starttime
from stl_query
where query in (
select distinct query
from stl_alert_event_log
where event like 'Nested Loop Join in the query plan%')
order by starttime desc;
```

Meninjau waktu tunggu antrian untuk kueri

Kueri berikut menunjukkan berapa lama kueri terbaru menunggu slot terbuka dalam antrian kueri sebelum dijalankan. Jika Anda melihat tren waktu tunggu yang tinggi, Anda mungkin ingin mengubah konfigurasi antrian kueri untuk throughput yang lebih baik. Untuk informasi selengkapnya, lihat Menerapkan manual WLM.

```
select trim(database) as DB , w.query,
substring(q.querytxt, 1, 100) as querytxt,  w.queue_start_time,
w.service_class as class, w.slot_count as slots,
w.total_queue_time/1000000 as queue_seconds,
w.total_exec_time/1000000 exec_seconds, (w.total_queue_time+w.total_Exec_time)/1000000
 as total_seconds
from stl_wlm_query w
left join stl_query q on q.query = w.query and q.userid = w.userid
where w.queue_start_Time >= dateadd(day, -7, current_Date)
and w.total_queue_Time > 0  and w.userid >1
and q.starttime >= dateadd(day, -7, current_Date)
order by w.total_queue_time desc, w.queue_start_time desc limit 35;
```

Meninjau peringatan kueri berdasarkan tabel

Kueri berikut mengidentifikasi tabel yang telah mencatat peristiwa peringatan untuk mereka, dan juga mengidentifikasi jenis peringatan apa yang paling sering dimunculkan.

Jika `minutes` nilai untuk baris dengan tabel yang diidentifikasi tinggi, periksa tabel tersebut untuk melihat apakah perlu pemeliharaan rutin, seperti memiliki MENGANALISA atau VAKUM menjalankannya.

Jika nil `count` ainya tinggi untuk baris tetapi nil `table` ainya nol, jalankan kueri terhadap STL_ALERT_EVENT_LOG untuk `event` nilai terkait untuk menyelidiki mengapa peringatan itu sering muncul.

```
select trim(s.perm_table_name) as table,
(sum(abs(datediff(seconds, s.starttime, s.endtime)))/60)::numeric(24,0) as minutes,
 trim(split_part(l.event,':',1)) as event,  trim(l.solution) as solution,
max(l.query) as sample_query, count(*)
from stl_alert_event_log as l
left join stl_scan as s on s.query = l.query and s.slice = l.slice
and s.segment = l.segment and s.step = l.step
where l.event_time >=  dateadd(day, -7, current_Date)
group by 1,3,4
order by 2 desc,6 desc;
```

Mengidentifikasi tabel dengan statistik yang hilang

Kueri berikut menyediakan jumlah kueri yang Anda jalankan terhadap tabel yang tidak memiliki statistik. Jika kueri ini mengembalikan baris apa pun, lihat `plannode` nilai untuk menentukan tabel yang terpengaruh, lalu jalankan MENGANALISA di atasnya.

```
select substring(trim(plannode),1,100) as plannode, count(*)
from stl_explain
where plannode like '%missing statistics%'
group by plannode
order by 2 desc;
```

Pemecahan masalah kueri

Bagian ini menyediakan referensi cepat untuk mengidentifikasi dan mengatasi beberapa masalah paling umum dan paling serius yang mungkin Anda temui dengan kueri Amazon Redshift.

Topik

- Koneksi gagal
- Kueri hang
- Kueri memakan waktu terlalu lama
- Beban gagal
- Beban memakan waktu terlalu lama
- Memuat data salah
- Mengatur parameter ukuran pengambilan JDBC

Saran ini memberi Anda titik awal untuk pemecahan masalah. Anda juga dapat merujuk ke sumber daya berikut untuk informasi lebih rinci.

Untuk informasi tentang perubahan perilaku dalam fungsionalitas Amazon Redshift yang dapat memengaruhi aplikasi Anda, lihat Perubahan Peril aku.

- Mengakses cluster dan database Amazon Redshift
- Pengoptimalan tabel otomatis
- Memuat data di Amazon Redshift
- Tutorial: Memuat data dari Amazon S3

Koneksi gagal

Koneksi kueri Anda dapat gagal karena alasan berikut. Kami menyarankan pendekatan pemecahan masalah berikut.

Klien tidak dapat terhubung ke server

Jika Anda menggunakan SSL atau sertifikat server, pertama-tama hapus kompleksitas ini saat Anda memecahkan masalah koneksi. Kemudian tambahkan SSL atau sertifikat server kembali ketika Anda telah menemukan solusi. Untuk informasi selengkapnya, buka Konfigurasi Opsi Keamanan untuk Koneksi di Panduan Manajemen Amazon Redshift.

Koneksi ditolak

Umumnya, ketika Anda menerima pesan kesalahan yang menunjukkan bahwa ada kegagalan untuk membuat koneksi, itu berarti ada masalah dengan izin untuk mengakses cluster. Untuk informasi selengkapnya, buka Koneksi ditolak atau gagal di Panduan Manajemen Amazon Redshift.

Kueri hang

Kueri Anda dapat macet, atau berhenti merespons, karena alasan berikut. Kami menyarankan pendekatan pemecahan masalah berikut.

Koneksi ke database terputus

Kurangi ukuran unit transmisi maksimum (MTU). Ukuran MTU menentukan ukuran maksimum, dalam byte, dari paket yang dapat ditransfer dalam satu bingkai Ethernet melalui koneksi jaringan Anda. Untuk informasi selengkapnya, buka Koneksi ke database dihentikan di Panduan Manajemen Amazon Redshift.

Koneksi ke database timeout

Koneksi klien Anda ke database tampaknya hang atau habis waktu saat menjalankan kueri panjang, seperti perintah COPY. Dalam hal ini, Anda mungkin mengamati bahwa konsol Amazon Redshift menampilkan bahwa kueri telah selesai, tetapi alat klien itu sendiri tampaknya masih menjalankan kueri. Hasil kueri mungkin hilang atau tidak lengkap tergantung kapan koneksi berhenti. Efek ini terjadi ketika koneksi idle diakhiri oleh komponen jaringan perantara. Untuk informasi selengkapnya, buka Masalah Batas Waktu Firewall di Panduan Manajemen Amazon Redshift.

Client-side kesalahan kehabisan memori terjadi dengan ODBC

Jika aplikasi klien Anda menggunakan koneksi ODBC dan kueri Anda membuat kumpulan hasil yang terlalu besar untuk muat dalam memori, Anda dapat melakukan streaming set hasil ke aplikasi klien Anda dengan menggunakan kursor. Untuk informasi selengkapnya, lihat MENYATAKAN dan Pertimbangan kinerja saat menggunakan kursor.

Client-side kesalahan kehabisan memori terjadi dengan JDBC

Saat Anda mencoba mengambil kumpulan hasil besar melalui koneksi JDBC, Anda mungkin mengalami kesalahan kehabisan memori sisi klien. Untuk informasi selengkapnya, lihat Mengatur parameter ukuran pengambilan JDBC.

Ada potensi kebuntuan

Jika ada potensi kebuntuan, coba yang berikut ini:

- Lihat tabel STV_LOCKS dan STL_TR_KONFLIK sistem untuk menemukan konflik yang melibatkan pembaruan ke lebih dari satu tabel.
- Gunakan PG_CANCEL_BACKEND fungsi untuk membatalkan satu atau lebih kueri yang saling bertentangan.
- Gunakan PG_TERMINATE_BACKEND fungsi untuk mengakhiri sesi, yang memaksa transaksi yang sedang berjalan dalam sesi yang dihentikan untuk melepaskan semua kunci dan mengembalikan transaksi.
- Jadwalkan operasi penulisan bersamaan dengan hati-hati. Untuk informasi selengkapnya, lihat Mengelola operasi penulisan bersamaan.

Kueri memakan waktu terlalu lama

Permintaan Anda bisa memakan waktu terlalu lama karena alasan berikut. Kami menyarankan pendekatan pemecahan masalah berikut.

Tabel tidak dioptimalkan

Atur kunci pengurutan, gaya distribusi, dan pengkodean kompresi tabel untuk memanfaatkan sepenuhnya pemrosesan paralel. Untuk informasi selengkapnya, lihat Pengoptimalan tabel otomatis

Query menulis ke disk

Kueri Anda mungkin menulis ke disk untuk setidaknya sebagian dari eksekusi kueri. Untuk informasi selengkapnya, lihat Peningkatan kinerja kueri.

Kueri harus menunggu kueri lain selesai

Anda mungkin dapat meningkatkan kinerja sistem secara keseluruhan dengan membuat antrian kueri dan menetapkan berbagai jenis kueri ke antrian yang sesuai. Untuk informasi selengkapnya, lihat Manajemen beban kerja.

Kueri tidak dioptimalkan

Analisis rencana penjelasan untuk menemukan peluang untuk menulis ulang kueri atau mengoptimalkan database. Untuk informasi selengkapnya, lihat Membuat dan menafsirkan rencana kueri.

Query membutuhkan lebih banyak memori untuk dijalankan

Jika kueri tertentu membutuhkan lebih banyak memori, Anda dapat meningkatkan memori yang tersedia dengan meningkatkanwlm_query_slot_count.

Database membutuhkan perintah VACUUM untuk dijalankan

Jalankan perintah VACUUM setiap kali Anda menambahkan, menghapus, atau memodifikasi sejumlah besar baris, kecuali Anda memuat data Anda dalam urutan kunci. Perintah VACUUM mengatur ulang data Anda untuk mempertahankan urutan pengurutan dan memulihkan kinerja. Untuk informasi selengkapnya, lihat Meja penyedot debu.

Sumber daya tambahan untuk memecahkan masalah kueri yang berjalan lama

Berikut ini adalah topik tampilan sistem dan bagian dokumentasi lainnya yang berguna untuk penyetelan kueri:

- T STV_INFLIGHT ampilan sistem menunjukkan kueri mana yang berjalan di cluster. Akan sangat membantu untuk menggunakannya bersama STV_TERBARU untuk menentukan kueri mana yang sedang berjalan atau baru saja selesai.
- SYS_QUERY_HISTORYberguna untuk pemecahan masalah. Ini menunjukkan kueri DDL dan DML dengan properti yang relevan seperti statusnya saat ini, seperti `running` atau`failed`, waktu yang dibutuhkan masing-masing untuk dijalankan, dan apakah kueri berjalan pada cluster penskalaan bersamaan.
- STL_QUERYTEXTmenangkap teks kueri untuk perintah SQL. Selain itu,SVV_QUERY_DALAM PENERBANGAN, yang menggabungkan STL_QUERYTEXT ke STV_INFLIGHT, menampilkan lebih banyak metadata kueri.
- Konflik kunci transaksi dapat menjadi kemungkinan sumber masalah kinerja kueri. Untuk informasi tentang transaksi yang saat ini menyimpan kunci pada tabel, lihatSVV_TRANSAKSI.
- Mengidentifikasi kueri yang merupakan kandidat teratas untuk penyetelan menyediakan kueri pemecahan masalah yang membantu Anda menentukan kueri yang baru-baru ini dijalankan yang paling memakan waktu. Ini dapat membantu Anda memfokuskan upaya Anda pada pertanyaan yang perlu ditingkatkan.
- Jika Anda ingin menjelajahi manajemen kueri lebih lanjut dan memahami cara mengelola antrian kueri, Manajemen beban kerja tunjukkan cara melakukannya. Manajemen beban kerja adalah fitur canggih dan kami merekomendasikan manajemen beban kerja otomatis dalam banyak kasus.

Beban gagal

Pemuatan data Anda dapat gagal karena alasan berikut. Kami menyarankan pendekatan pemecahan masalah berikut.

Sumber data berada di tempat yang berbeda AWS Region

Secara default, bucket Amazon S3 atau tabel Amazon DynamoDB yang ditentukan dalam perintah COPY harus berada di Wilayah yang sama AWS dengan cluster. Jika data dan cluster Anda berada di Wilayah yang berbeda, Anda menerima kesalahan yang mirip dengan berikut ini:

```
The bucket you are attempting to access must be addressed using the specified endpoint.
```

Jika memungkinkan, pastikan cluster dan sumber data Anda berada di Wilayah yang sama. Anda dapat menentukan Wilayah yang berbeda dengan menggunakan REGION opsi dengan perintah COPY.

> **Note**
>
> Jika cluster dan sumber data Anda berada di Wil AWS ayah yang berbeda, Anda akan dikenakan biaya transfer data. Anda juga memiliki latensi yang lebih tinggi.

Perintah COPY gagal

Kueri STL_LOAD_ERRORS untuk menemukan kesalahan yang terjadi selama pemuatan tertentu. Untuk informasi selengkapnya, lihat STL_LOAD_ERROR.

Beban memakan waktu terlalu lama

Operasi beban Anda bisa memakan waktu terlalu lama karena alasan berikut. Kami menyarankan pendekatan pemecahan masalah berikut.

COPY memuat data dari satu file

Pisahkan data muat Anda menjadi beberapa file. Saat Anda memuat semua data dari satu file besar, Amazon Redshift dipaksa untuk melakukan pemuatan serial, yang jauh lebih lambat. Jumlah file harus kelipatan dari jumlah irisan di cluster Anda, dan file harus berukuran hampir sama, antara 1 MB dan 1 GB setelah kompresi. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang kueri.

Operasi beban menggunakan beberapa perintah COPY

Jika Anda menggunakan beberapa perintah COPY bersamaan untuk memuat satu tabel dari beberapa file, Amazon Redshift dipaksa untuk melakukan pemuatan serial, yang jauh lebih lambat. Dalam hal ini, gunakan perintah COPY tunggal.

Memuat data salah

Operasi COPY Anda dapat memuat data yang salah dengan cara berikut. Kami menyarankan pendekatan pemecahan masalah berikut.

File yang salah dimuat

Menggunakan awalan objek untuk menentukan file data dapat menyebabkan file yang tidak diinginkan dibaca. Sebagai gantinya, gunakan file manifes untuk menentukan dengan tepat file mana yang akan dimuat. Untuk informasi selengkapnya, lihat copy_from_s3_manifest_file opsi untuk perintah SALIN dan Example: COPY from Amazon S3 using a manifest dalam contoh SALIN.

Mengatur parameter ukuran pengambilan JDBC

Secara default, driver Redshift JDBC menggunakan buffer cincin untuk mengelola memori secara efisien dan mencegah kesalahan kehabisan memori. Parameter ukuran fetch hanya berlaku ketika buffer cincin dinonaktifkan secara eksplisit. Untuk informasi lebih lanjut, tinjau tautannya. Dalam konfigurasi ini, Anda harus mengatur ukuran pengambilan untuk mengontrol berapa banyak baris yang diambil di setiap batch.

Kapan Menggunakan Ukuran Fetch?

Gunakan parameter ukuran fetch saat:

- Anda memerlukan kontrol halus atas batching berbasis baris
- Bekerja dengan aplikasi lama yang memerlukan perilaku ukuran pengambilan tradisional

Mengatur Ukuran Fetch

Ketika buffer dering dinonaktifkan, driver JDBC mengumpulkan semua hasil untuk kueri pada satu waktu secara default. Kueri yang mengembalikan set hasil besar dapat menghabiskan memori yang berlebihan. Untuk mengambil set hasil dalam batch, bukan sekaligus, atur parameter ukuran pengambilan JDBC di aplikasi Anda.

> **Note**
>
> Ukuran pengambilan tidak didukung untuk ODBC.

Untuk kinerja terbaik, atur ukuran pengambilan ke nilai tertinggi yang tidak menyebabkan kesalahan kehabisan memori. Nilai ukuran fetch yang lebih rendah menghasilkan lebih banyak perjalanan server, yang memperpanjang waktu eksekusi. Server menyimpan sumber daya, termasuk slot kueri WLM dan memori terkait, sampai klien mengambil seluruh kumpulan hasil atau kueri dibatalkan. Saat Anda menyetel ukuran pengambilan dengan tepat, sumber daya tersebut dilepaskan lebih cepat, membuatnya tersedia untuk kueri lain.

> **Note**
>
> Jika Anda perlu mengekstrak kumpulan data besar, sebaiknya gunakan pernyataan UNLOAD untuk mentransfer data ke Amazon S3. Saat Anda menggunakan UNLOAD, node komputasi bekerja secara paralel untuk mempercepat transfer data.

Untuk informasi selengkapnya tentang pengaturan parameter ukuran pengambilan JDBC, buka Mendap atkan hasil berdasarkan kursor dalam dokumentasi PostgreSQL.

Manajemen beban kerja

Anda dapat mengonfigurasi Amazon Redshift WLM untuk dijalankan dengan WLM otomatis atau WLM manual.

Dengan Amazon Redshift, Anda dapat mengelola dan memprioritaskan kueri bersamaan dan beban kerja pengguna untuk mengoptimalkan kinerja dan pemanfaatan sumber daya. Dengan manajemen beban kerja (WLM), Anda dapat menentukan antrian, grup pengguna, dan konstruksi lainnya untuk mengontrol sumber daya yang dialokasikan ke berbagai jenis kueri atau pengguna.

Bagian berikut menguraikan fitur manajemen beban kerja tertentu di Amazon Redshift, dan memandu Anda melalui konfigurasi dan pemantauannya.

WLM otomatis

Untuk memaksimalkan throughput sistem dan menggunakan sumber daya secara efektif, Anda dapat mengaktifkan Amazon Redshift untuk mengelola cara sumber daya dibagi untuk menjalankan kueri bersamaan dengan WLM otomatis. WLM otomatis mengelola sumber daya yang diperlukan untuk menjalankan kueri. Amazon Redshift menentukan berapa banyak kueri yang dijalankan secara bersamaan dan berapa banyak memori yang dialokasikan untuk setiap kueri yang dikirim. Gunakan Auto WLM jika Anda ingin Amazon Redshift mengelola bagaimana sumber daya dibagi untuk menjalankan kueri bersamaan. Untuk informasi selengkapnya, lihat Menerapkan WLM otomatis.

Bekerja dengan penskalaan konkurensi dan WLM otomatis, Anda dapat mendukung pengguna bersamaan dan kueri bersamaan yang hampir tidak terbatas, dengan kinerja kueri yang cepat secara konsisten. Untuk informasi selengkapnya, lihat Penskalaan konkurensi.

> **Note**
>
> Dalam kebanyakan kasus kami menyarankan Anda menggunakan WLM otomatis. Jika Anda menggunakan WLM manual dan Anda ingin bermigrasi dari ke WLM otomatis, lihat. Migrasi dari WLM manual ke WLM otomatis

Dengan Auto WLM, dimungkinkan untuk menentukan prioritas kueri untuk beban kerja dalam antrian. Untuk informasi selengkapnya tentang prioritas kueri, lihatPrioritas kueri.

WLM manual

Anda mungkin memiliki beberapa sesi atau pengguna yang menjalankan kueri secara bersamaan. Beberapa kueri mungkin mengkonsumsi sumber daya cluster untuk waktu yang lama dan memengaruhi kinerja yang lain. Manual WLM dapat membantu mengelola ini untuk kasus penggunaan khusus. Gunakan Manual WLM ketika Anda ingin lebih banyak kontrol atas konkurensi.

Anda dapat mengelola kinerja sistem dengan memodifikasi konfigurasi WLM untuk membuat antrian terpisah untuk kueri yang berjalan lama dan kueri yang berjalan singkat. Saat runtime, Anda dapat merutekan kueri ke antrian ini sesuai dengan grup pengguna atau grup kueri.

Anda dapat mengatur aturan untuk merutekan kueri ke antrian tertentu berdasarkan pengguna yang menjalankan kueri atau label yang Anda tentukan. Anda juga dapat mengonfigurasi jumlah memori yang dialokasikan untuk setiap antrian, sehingga kueri besar berjalan dalam antrian dengan lebih banyak memori daripada antrian lainnya. Anda juga dapat mengonfigurasi aturan pemantauan kueri (QMR) untuk membatasi kueri yang berjalan lama. Untuk informasi selengkapnya, lihat Menerapkan manual WLM.

> **Note**
>
> Sebaiknya konfigurasi antrian kueri WLM manual Anda dengan total 15 slot kueri atau lebih sedikit. Untuk informasi selengkapnya, lihat Tingkat konkurensi.

Perhatikan bahwa sehubungan dengan konfigurasi WLM manual, slot maksimum yang dapat Anda alokasikan ke antrian adalah 50. Namun, ini tidak berarti bahwa dalam konfigurasi WLM otomatis, cluster Amazon Redshift selalu menjalankan 50 kueri secara bersamaan. Hal ini dapat berubah, berdasarkan kebutuhan memori atau jenis alokasi sumber daya lainnya pada cluster.

Topik

- Mengalihkan mode WLM
- Memodifikasi konfigurasi WLM
- Menerapkan WLM otomatis
- Menerapkan manual WLM
- Penskalaan konkurensi
- Akselerasi kueri singkat
- Aturan penetapan antrian WLM
- Menetapkan kueri ke antrian
- Properti konfigurasi dinamis dan statis WLM

- Aturan pemantauan kueri WLM
- Tabel dan tampilan sistem WLM

Mengalihkan mode WLM

Anda dapat mengaktifkan WLM otomatis atau manual menggunakan konsol Amazon Redshift:

1. Pilih Ber alih mode WLM.
2. Untuk mengaturnya ke WLM otomatis, pilih Auto W LM. Dengan pilihan ini, hingga delapan antrian digunakan untuk mengelola kueri, dan Memori dan Konkur ensi pada bidang utama keduanya diatur ke Otomatis. Selain itu, prioritas default kueri diatur ke Normal.
3. Untuk mengaktifkan konfigurasi manual menggunakan konsol Amazon Redshift, beralih ke Manual W LM. Dengan pilihan ini, Anda menentukan antrian yang digunakan untuk mengelola kueri, dan Memori dan Konkur ensi pada nilai bidang utama. Dengan konfigurasi manual, Anda dapat mengonfigurasi hingga delapan antrian kueri dan mengatur jumlah kueri yang dapat dijalankan di masing-masing antrian tersebut secara bersamaan.

Memodifikasi konfigurasi WLM

Cara termudah untuk memodifikasi konfigurasi WLM adalah dengan menggunakan konsol Amazon Redshift. Anda juga dapat menggunakan AWS CLI atau Amazon Redshift API.

Ketika Anda mengalihkan cluster Anda antara WLM otomatis dan manual, cluster Anda dimasukkan ke dalam `pending reboot` status. Perubahan tidak berlaku sampai reboot cluster berikutnya.

Untuk informasi terperinci tentang memodifikasi konfigurasi WLM, lihat Mengon figurasi Manajemen Beb an Kerja di Panduan Manajemen Amazon Redshift.

Migrasi dari WLM manual ke WLM otomatis

Untuk memaksimalkan throughput sistem dan menggunakan sumber daya dengan paling efektif, sebaiknya siapkan WLM otomatis untuk antrian Anda. Pertimbangkan untuk mengambil pendekatan berikut untuk mengatur transisi yang mulus dari WLM manual ke WLM otomatis.

Untuk bermigrasi dari WLM manual ke WLM otomatis dan menggunakan prioritas kueri, sebaiknya buat grup parameter baru, lalu lampirkan grup parameter tersebut ke cluster Anda. Untuk informasi selengkapnya, lihat Grup Parameter Amazon Redshift di Panduan Manajemen Amazon Redshift.

> **Important**
>
> Untuk mengubah grup parameter atau beralih dari manual ke WLM otomatis memerlukan reboot cluster. Untuk informasi selengkapnya, lihat Properti konfigurasi dinamis dan statis WLM.

Mari kita ambil contoh di mana ada tiga antrian WLM manual. Masing-masing satu untuk beban kerja ETL, beban kerja analitik, dan beban kerja ilmu data. Beban kerja ETL berjalan setiap 6 jam, beban kerja analitik berjalan sepanjang hari, dan beban kerja ilmu data dapat melonjak kapan saja. Dengan WLM manual, Anda menentukan memori dan konkurensi yang didapat setiap antrian beban kerja berdasarkan pemahaman Anda tentang pentingnya setiap beban kerja bagi bisnis. Menentukan memori dan konkurensi tidak hanya sulit untuk diketahui, tetapi juga mengakibatkan sumber daya cluster dipartisi secara statis dan dengan demikian terbuang ketika hanya sebagian dari beban kerja yang berjalan.

Anda dapat menggunakan WLM otomatis dengan prioritas kueri untuk menunjukkan prioritas relatif beban kerja, menghindari masalah sebelumnya. Untuk contoh ini, ikuti langkah-langkah ini:

- Buat grup parameter baru dan beralih ke mode Auto WLM.
- Tambahkan antrian untuk masing-masing dari tiga beban kerja: beban kerja ETL, beban kerja analitik, dan beban kerja ilmu data. Gunakan grup pengguna yang sama untuk setiap beban kerja yang digunakan dengan mode WLM Manual.
- Tetapkan prioritas untuk beban kerja ETL ke`High`, beban kerja analitik ke`Normal`, dan ilmu data ke`Low`. Prioritas ini mencerminkan prioritas bisnis Anda untuk beban kerja atau grup pengguna yang berbeda.
- Secara opsional, aktifkan penskalaan konkurensi untuk antrian analitik atau ilmu data sehingga kueri dalam antrian ini mendapatkan kinerja yang konsisten bahkan ketika beban kerja ETL berjalan setiap 6 jam.

Dengan prioritas kueri, ketika hanya beban kerja analitik yang berjalan di cluster, itu akan mendapatkan seluruh sistem untuk dirinya sendiri. Ini menghasilkan throughput tinggi dengan pemanfaatan sistem yang lebih baik. Namun, ketika beban kerja ETL dimulai, itu menjadi benar karena memiliki prioritas yang lebih tinggi. Kueri yang berjalan sebagai bagian dari beban kerja ETL mendapatkan prioritas selama penerimaan, selain alokasi sumber daya preferensial setelah diterima. Akibatnya, beban kerja ETL bekerja secara dapat diprediksi terlepas dari apa lagi yang mungkin

berjalan di sistem. Kinerja yang dapat diprediksi untuk beban kerja prioritas tinggi datang dengan mengorbankan beban kerja lain dengan prioritas lebih rendah yang berjalan lebih lama karena kueri mereka menunggu di belakang kueri yang lebih penting untuk diselesaikan. Atau, karena mereka mendapatkan sebagian kecil sumber daya ketika mereka berjalan bersamaan dengan kueri prioritas yang lebih tinggi. Algoritma penjadwalan yang digunakan oleh Amazon Redshift memfasilitasi bahwa kueri prioritas rendah tidak menderita kelaparan, melainkan terus membuat kemajuan meskipun pada kecepatan yang lebih lambat.

Note

- Bidang batas waktu tidak tersedia di WLM otomatis. Sebagai gantinya, gunakan aturan QMR,`query_execution_time`. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.
- Tindakan QMR, HOP, tidak berlaku untuk WLM otomatis. Sebagai gantinya, gunakan `change priority` tindakan. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.
- Cluster menggunakan antrian WLM otomatis dan WLM manual secara berbeda, yang dapat menyebabkan kebingungan dengan konfigurasi Anda. Misalnya, Anda dapat mengonfigurasi properti prioritas dalam antrian WLM otomatis tetapi tidak dalam antrian WLM manual. Dengan demikian, hindari mencampur antrian WLM otomatis dan antrian WLM manual dalam grup parameter. Sebagai gantinya, buat grup parameter baru saat bermigrasi ke WLM otomatis.

Menerapkan WLM otomatis

Dengan manajemen beban kerja otomatis (WLM), Amazon Redshift mengelola konkurensi kueri dan alokasi memori. Anda dapat membuat hingga delapan antrian dengan pengidentifikasi kelas layanan 100-107. Setiap antrian memiliki prioritas. Untuk informasi selengkapnya, lihat Prioritas kueri.

WLM otomatis menentukan jumlah sumber daya yang dibutuhkan kueri dan menyesuaikan konkurensi berdasarkan beban kerja. Ketika kueri yang membutuhkan sejumlah besar sumber daya ada dalam sistem (misalnya, hash bergabung di antara tabel besar), konkurensinya lebih rendah. Ketika kueri yang lebih ringan (seperti sisipan, penghapusan, pemindaian, atau agregasi sederhana) dikirimkan, konkurensi lebih tinggi.

WLM otomatis terpisah dari akselerasi kueri pendek (SQA) dan mengevaluasi kueri secara berbeda. WLM dan SQA otomatis bekerja sama untuk memungkinkan kueri yang berjalan pendek dan ringan diselesaikan bahkan saat kueri intensif sumber daya aktif dalam waktu lama. Untuk informasi lebih lanjut tentang SQA, lihatAkselerasi kueri singkat.

Amazon Redshift mengaktifkan WLM otomatis melalui grup parameter:

- Jika cluster Anda menggunakan grup parameter default, Amazon Redshift mengaktifkan WLM otomatis untuknya.
- Jika cluster Anda menggunakan grup parameter khusus, Anda dapat mengonfigurasi cluster untuk mengaktifkan WLM otomatis. Sebaiknya buat grup parameter terpisah untuk konfigurasi WLM otomatis Anda.

Untuk mengkonfigurasi WLM, edit `wlm_json_configuration` parameter dalam grup parameter yang dapat dikaitkan dengan satu atau lebih cluster. Untuk informasi selengkapnya, lihat Memodifikasi konfigurasi WLM.

Anda menentukan antrian kueri dalam konfigurasi WLM. Anda dapat menambahkan antrian kueri tambahan ke konfigurasi WLM default, hingga total delapan antrian pengguna. Anda dapat mengonfigurasi berikut ini untuk setiap antrian kueri:

- Prioritas
- Mode penskalaan konkurensi
- Grup pengguna
- Grup kueri
- Aturan pemantauan kueri

Prioritas

Anda dapat menentukan kepentingan relatif kueri dalam beban kerja dengan menetapkan nilai prioritas. Prioritas ditentukan untuk antrian dan diwarisi oleh semua kueri yang terkait dengan antrian. Untuk informasi selengkapnya, lihat Prioritas kueri.

Mode penskalaan konkurensi

Saat penskalaan konkurensi diaktifkan, Amazon Redshift secara otomatis menambahkan kapasitas cluster tambahan saat Anda membutuhkannya untuk memproses peningkatan kueri baca dan tulis

bersamaan. Pengguna Anda melihat data terbaru, apakah kueri berjalan di cluster utama atau pada cluster penskalaan konkurensi.

Anda mengelola kueri mana yang dikirim ke cluster penskalaan konkurensi dengan mengonfigurasi antrian WLM. Saat Anda mengaktifkan penskalaan konkurensi untuk antrian, kueri yang memenuhi syarat dikirim ke cluster penskalaan konkurensi alih-alih menunggu dalam antrian. Untuk informasi selengkapnya, lihat Penskalaan konkurensi.

Grup pengguna

Anda dapat menetapkan satu set grup pengguna ke antrian dengan menentukan setiap nama grup pengguna atau dengan menggunakan wildcard. Ketika anggota grup pengguna terdaftar menjalankan kueri, kueri tersebut berjalan dalam antrian yang sesuai. Tidak ada batasan yang ditetapkan pada jumlah grup pengguna yang dapat ditetapkan ke antrian. Untuk informasi selengkapnya, lihat Menetapkan kueri ke antrian berdasarkan grup pengguna.

Peran pengguna

Anda dapat menetapkan satu set peran pengguna ke antrian dengan menentukan setiap nama peran pengguna atau dengan menggunakan wildcard. Ketika anggota peran pengguna terdaftar menjalankan kueri, kueri tersebut berjalan dalam antrian yang sesuai. Tidak ada batasan yang ditetapkan pada jumlah peran pengguna yang dapat ditetapkan ke antrian. Untuk informasi selengkapnya, lihat Menetapkan kueri ke antrian berdasarkan peran pengguna.

Grup kueri

Anda dapat menetapkan kumpulan grup kueri ke antrian dengan menentukan setiap nama grup kueri atau dengan menggunakan wildcard. Grup query hanyalah sebuah label. Saat runtime, Anda dapat menetapkan label grup kueri ke serangkaian kueri. Setiap kueri yang ditetapkan ke grup kueri terdaftar dijalankan dalam antrian yang sesuai. Tidak ada batasan yang ditetapkan untuk jumlah grup kueri yang dapat ditetapkan ke antrian. Untuk informasi selengkapnya, lihat Menetapkan kueri ke grup kueri.

Wildcard

Jika wildcard diaktifkan dalam konfigurasi antrian WLM, Anda dapat menetapkan grup pengguna dan grup kueri ke antrian baik secara individual atau dengan menggunakan wildcard gaya shell Unix. Pencocokan pola tidak peka huruf besar/kecil.

Misalnya, karakter wildcard '*' cocok dengan sejumlah karakter. Jadi, jika Anda menambahkan `dba_*` ke daftar grup pengguna untuk antrian, kueri yang dijalankan pengguna yang termasuk dalam grup dengan nama yang dimulai dengan `dba_` ditetapkan ke antrian tersebut. Contohnya adalah `dba_admin` atau `DBA_primary`. Yang '?' karakter wildcard cocok dengan karakter tunggal mana pun. Jadi, jika antrian menyertakan grup penggunadba?1, maka grup pengguna diberi `dba11` nama dan `dba21` cocok, tetapi `dba12` tidak cocok.

Secara default, wildcard tidak diaktifkan.

Aturan pemantauan kueri

Aturan pemantauan kueri menentukan batas kinerja berbasis metrik untuk antrian WLM dan menentukan tindakan apa yang harus diambil ketika kueri melampaui batas tersebut. Misalnya, untuk antrian yang didedikasikan untuk kueri berjalan singkat, Anda dapat membuat aturan yang membatalkan kueri yang berjalan selama lebih dari 60 detik. Untuk melacak kueri yang dirancang dengan buruk, Anda mungkin memiliki aturan lain yang mencatat kueri yang berisi loop bersarang. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

Memeriksa WLM otomatis

Untuk memeriksa apakah WLM otomatis diaktifkan, jalankan kueri berikut. Jika kueri mengembalikan setidaknya satu baris, maka WLM otomatis diaktifkan.

```
select * from stv_wlm_service_class_config
where service_class >= 100;
```

Query berikut menunjukkan jumlah query yang melewati setiap antrian kueri (kelas layanan). Ini juga menunjukkan waktu eksekusi rata-rata, jumlah kueri dengan waktu tunggu pada persentil ke-90, dan waktu tunggu rata-rata. Kueri WLM otomatis menggunakan kelas layanan 100 hingga 107.

```
select final_state, service_class, count(*), avg(total_exec_time),
percentile_cont(0.9) within group (order by total_queue_time), avg(total_queue_time)
from stl_wlm_query where userid >= 100 group by 1,2 order by 2,1;
```

Untuk menemukan kueri mana yang dijalankan oleh WLM otomatis, dan berhasil diselesaikan, jalankan kueri berikut.

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
```

```
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class >= 100 and a.final_state = 'Completed'
order by b.query desc limit 5;
```

Prioritas kueri

Dengan Amazon Redshift, Anda dapat mengelola prioritas kueri dan alokasi sumber daya di seluruh kueri dan beban kerja bersamaan menggunakan Manajemen Beban Kerja (WM). Bagian berikut merinci cara mengonfigurasi antrian kueri WM, menentukan properti antrian seperti alokasi memori dan penskalaan konkurensi, dan menerapkan aturan prioritas yang disesuaikan dengan persyaratan beban kerja Anda.

Tidak semua kueri sama pentingnya, dan seringkali kinerja satu beban kerja atau kumpulan pengguna mungkin lebih penting. Jika Anda telah mengaktifkan WLM otomatis, Anda dapat menentukan kepentingan relatif kueri dalam beban kerja dengan menetapkan nilai prioritas. Prioritas ditentukan untuk antrian dan diwarisi oleh semua kueri yang terkait dengan antrian. Anda mengaitkan kueri ke antrian dengan memetakan grup pengguna dan grup kueri ke antrian. Anda dapat mengatur prioritas berikut (tercantum dari prioritas tertinggi ke terendah):

1. `HIGHEST`
2. `HIGH`
3. `NORMAL`
4. `LOW`
5. `LOWEST`

Administrator menggunakan prioritas ini untuk menunjukkan pentingnya relatif beban kerja mereka ketika ada kueri dengan prioritas berbeda yang bersaing untuk sumber daya yang sama. Amazon Redshift menggunakan prioritas saat membiarkan kueri masuk ke sistem, dan untuk menentukan jumlah sumber daya yang dialokasikan untuk kueri. Secara default, kueri berjalan dengan prioritasnya ditetapkan ke`NORMAL`.

Prioritas tambahan`CRITICAL`,, yang merupakan prioritas lebih tinggi daripada`HIGHEST`, tersedia untuk pengguna super. Untuk mengatur prioritas ini, Anda dapat menggunakan fungsi`UBAH_KUERI_PRIORITAS`,`UBAH_PRIORITAS_SESI`. dan`UBAH_PRIORITAS_PENGGUNA`. Untuk memberikan izin pengguna database untuk menggunakan fungsi-fungsi ini, Anda dapat membuat prosedur tersimpan dan memberikan izin kepada pengguna. Sebagai contoh, lihat `UBAH_PRIORITAS_SESI`.

> **Note**
>
> Hanya satu `CRITICAL` query yang dapat dijalankan pada satu waktu.
> Rollback selalu berjalan sebagai prioritas KRITIS.

Mari kita ambil contoh di mana prioritas beban kerja ekstrak, transformasi, muat (ETL) lebih tinggi daripada prioritas beban kerja analitik. Beban kerja ETL berjalan setiap enam jam, dan beban kerja analitik berjalan sepanjang hari. Ketika hanya beban kerja analitik yang berjalan di cluster, itu mendapatkan seluruh sistem untuk dirinya sendiri, menghasilkan throughput tinggi dengan pemanfaatan sistem yang optimal. Namun, ketika beban kerja ETL dimulai, itu menjadi benar karena memiliki prioritas yang lebih tinggi. Kueri yang berjalan sebagai bagian dari beban kerja ETL mendapatkan hak selama penerimaan dan juga alokasi sumber daya preferensial setelah diterima. Akibatnya, beban kerja ETL bekerja secara dapat diprediksi terlepas dari apa lagi yang mungkin berjalan di sistem. Dengan demikian, ini memberikan kinerja yang dapat diprediksi dan kemampuan bagi administrator untuk memberikan perjanjian tingkat layanan (SLA) untuk pengguna bisnis mereka.

Dalam cluster tertentu, kinerja yang dapat diprediksi untuk beban kerja prioritas tinggi datang dengan mengorbankan beban kerja lain dengan prioritas lebih rendah. Beban kerja prioritas rendah mungkin berjalan lebih lama karena kueri mereka menunggu di belakang kueri yang lebih penting untuk diselesaikan. Atau mereka mungkin berjalan lebih lama karena mereka mendapatkan fraksi sumber daya yang lebih kecil ketika mereka berjalan bersamaan dengan kueri prioritas yang lebih tinggi. Kueri prioritas rendah tidak menderita kelaparan, melainkan terus membuat kemajuan dengan kecepatan yang lebih lambat.

Dalam contoh sebelumnya, administrator dapat mengaktifkan penskalaan konkurensi untuk beban kerja analitik. Melakukan hal ini memungkinkan beban kerja tersebut mempertahankan throughput, meskipun beban kerja ETL berjalan dengan prioritas tinggi.

Mengkonfigurasi prioritas antrian

Jika Anda telah mengaktifkan WLM otomatis, setiap antrian memiliki nilai prioritas. Kueri dialihkan ke antrian berdasarkan grup pengguna dan grup kueri. Mulailah dengan prioritas antrian yang disetel ke`NORMAL`. Tetapkan prioritas lebih tinggi atau lebih rendah berdasarkan beban kerja yang terkait dengan grup pengguna antrian dan grup kueri.

Anda dapat mengubah prioritas antrian di konsol Amazon Redshift. Di konsol Amazon Redshift, halaman Manajemen Beban Kerja menampilkan antrian dan memungkinkan pengeditan properti

antrian seperti Prioritas. Untuk mengatur prioritas menggunakan operasi CLI atau API, gunakan `wlm_json_configuration` parameter. Untuk informasi selengkapnya, lihat Meng onfigurasi Manajemen B eban Kerja di Panduan Manajemen Amazon Redshift.

`wlm_json_configuration`Contoh berikut mendefinisikan tiga grup pengguna (`ingest`,`reporting`, dan`analytics`). Kueri yang dikirimkan dari pengguna dari salah satu grup ini dijalankan dengan prioritas `highest`,`normal`, dan`low`, masing-masing.

```
[
    {
        "user_group": [
            "ingest"
        ],
        "priority": "highest",
        "queue_type": "auto"
    },
    {
        "user_group": [
            "reporting"
        ],
        "priority": "normal",
        "queue_type": "auto"
    },
    {
        "user_group": [
            "analytics"
        ],
        "priority": "low",
        "queue_type": "auto",
        "auto_wlm": true
    }
]
```

Mengubah prioritas kueri dengan aturan pemantauan kueri

Dengan aturan pemantauan kueri (QMR), Anda dapat mengubah prioritas kueri berdasarkan perilakunya saat sedang berjalan. Anda melakukan ini dengan menentukan atribut prioritas dalam predikat QMR selain tindakan. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

Misalnya, Anda dapat menentukan aturan untuk membatalkan kueri apa pun yang diklasifikasikan sebagai `high` prioritas yang berjalan selama lebih dari 10 menit.

```
"rules" :[
  {
    "rule_name":"rule_abort",
    "predicate":[
      {
        "metric_name":"query_cpu_time",
        "operator":">",
        "value":600
      },
      {
        "metric_name":"query_priority",
        "operator":"=",
        "value":"high"
      }
    ],
    "action":"abort"
  }
]
```

Contoh lain adalah menentukan aturan untuk mengubah prioritas kueri `lowest` untuk setiap kueri dengan prioritas saat ini `normal` yang menumpahkan lebih dari 1 TB ke disk.

```
"rules":[
  {
    "rule_name":"rule_change_priority",
    "predicate":[
      {
        "metric_name":"query_temp_blocks_to_disk",
        "operator":">",
        "value":1000000
      },
      {
        "metric_name":"query_priority",
        "operator":"=",
        "value":"normal"
      }
    ],
    "action":"change_query_priority",
    "value":"lowest"
  }
]
```

Prioritas kueri pemantauan

Untuk menampilkan prioritas untuk menunggu dan menjalankan kueri, lihat `query_priority` kolom di tabel sistem stv_wlm_query_state.

```
query    | service_cl | wlm_start_time             | state            | queue_time |
 query_priority
---------+------------+----------------------------+------------------+------------
+----------------
2673299  | 102        | 2019-06-24 17:35:38.866356 | QueuedWaiting    | 265116     |
 Highest
2673236  | 101        | 2019-06-24 17:35:33.313854 | Running          | 0          |
 Highest
2673265  | 102        | 2019-06-24 17:35:33.523332 | Running          | 0          |
 High
2673284  | 102        | 2019-06-24 17:35:38.477366 | Running          | 0          |
 Highest
2673288  | 102        | 2019-06-24 17:35:38.621819 | Running          | 0          |
 Highest
2673310  | 103        | 2019-06-24 17:35:39.068513 | QueuedWaiting    | 62970      |
 High
2673303  | 102        | 2019-06-24 17:35:38.968921 | QueuedWaiting    | 162560     |
 Normal
2673306  | 104        | 2019-06-24 17:35:39.002733 | QueuedWaiting    | 128691     |
 Lowest
```

Untuk mencantumkan prioritas kueri untuk kueri yang diselesaikan, lihat `query_priority` kolom di tabel sistem stl_wlm_query.

```
select query, service_class as svclass, service_class_start_time as starttime,
 query_priority
from stl_wlm_query order by 3 desc limit 10;
```

```
  query  | svclass |         starttime          |    query_priority
---------+---------+----------------------------+----------------------
 2723254 |     100 | 2019-06-24 18:14:50.780094 | Normal
 2723251 |     102 | 2019-06-24 18:14:50.749961 | Highest
 2723246 |     102 | 2019-06-24 18:14:50.725275 | Highest
 2723244 |     103 | 2019-06-24 18:14:50.719241 | High
 2723243 |     101 | 2019-06-24 18:14:50.699325 | Low
```

```
2723242 |     102 | 2019-06-24 18:14:50.692573 | Highest
2723239 |     101 | 2019-06-24 18:14:50.668535 | Low
2723237 |     102 | 2019-06-24 18:14:50.661918 | Highest
2723236 |     102 | 2019-06-24 18:14:50.643636 | Highest
```

Untuk mengoptimalkan throughput beban kerja Anda, Amazon Redshift dapat mengubah prioritas kueri yang dikirimkan pengguna. Amazon Redshift menggunakan algoritma pembelajaran mesin tingkat lanjut untuk menentukan kapan pengoptimalan ini menguntungkan beban kerja Anda dan menerapkannya secara otomatis ketika semua kondisi berikut terpenuhi.

- WLM otomatis diaktifkan.
- Hanya satu antrian WLM yang didefinisikan.
- Anda belum menentukan aturan pemantauan kueri (QMR) yang menetapkan prioritas kueri. Aturan tersebut termasuk metrik QMR `query_priority` atau tindakan QMR. `change_query_priority` Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

Menerapkan manual WLM

Dengan WLM manual, Anda dapat mengelola kinerja sistem dan pengalaman pengguna Anda dengan memodifikasi konfigurasi WLM untuk membuat antrian terpisah untuk kueri yang berjalan lama dan kueri yang berjalan pendek.

Saat pengguna menjalankan kueri di Amazon Redshift, kueri dialihkan ke antrian kueri. Setiap antrian kueri berisi sejumlah slot kueri. Setiap antrian dialokasikan sebagian dari memori cluster yang tersedia. Memori antrian dibagi di antara slot query antrian. Anda dapat mengaktifkan Amazon Redshift untuk mengelola konkurensi kueri dengan WLM otomatis. Untuk informasi selengkapnya, lihat Menerapkan WLM otomatis.

Atau Anda dapat mengkonfigurasi properti WLM untuk setiap antrian kueri. Anda melakukannya untuk menentukan cara memori dialokasikan di antara slot dan bagaimana kueri dapat dialihkan ke antrian tertentu saat runtime. Anda juga dapat mengonfigurasi properti WLM untuk membatalkan kueri yang berjalan lama.

Secara default, Amazon Redshift mengonfigurasi antrian kueri berikut:

- Satu antrian superuser

  Antrian superuser disediakan untuk pengguna super saja dan tidak dapat dikonfigurasi. Gunakan antrian ini hanya ketika Anda perlu menjalankan kueri yang memengaruhi sistem atau untuk tujuan

pemecahan masalah. Misalnya, gunakan antrian ini saat Anda perlu membatalkan kueri pengguna yang sudah berjalan lama atau menambahkan pengguna ke database. Jangan menggunakannya untuk melakukan query rutin. Antrian tidak muncul di konsol, tetapi muncul di tabel sistem di database sebagai antrian kelima. Untuk menjalankan kueri dalam antrian superuser, pengguna harus masuk sebagai superuser, dan harus menjalankan kueri menggunakan grup kueri yang telah ditentukan sebelumnya`superuser`.

- Satu antrian pengguna default

  Antrian default awalnya dikonfigurasi untuk menjalankan lima kueri secara bersamaan. Saat menggunakan WLM manual, Anda dapat mengubah properti konkurensi, batas waktu, dan alokasi memori untuk antrian default, tetapi Anda tidak dapat menentukan grup pengguna atau grup kueri. Antrian default harus menjadi antrian terakhir dalam konfigurasi WLM. Setiap kueri yang tidak dialihkan ke antrian lain dijalankan dalam antrian default.

Antrian kueri didefinisikan dalam konfigurasi WLM. Konfigurasi WLM adalah parameter yang dapat diedit (`wlm_json_configuration`) dalam grup parameter, yang dapat dikaitkan dengan satu atau lebih cluster. Untuk informasi selengkapnya, lihat Meng onfigurasi Manajemen B eban Kerja di Panduan Manajemen Amazon Redshift.

Anda dapat menambahkan antrian kueri tambahan ke konfigurasi WLM default, hingga total delapan antrian pengguna. Anda dapat mengonfigurasi berikut ini untuk setiap antrian kueri:

- Mode penskalaan konkurensi
- Tingkat konkurensi
- Grup pengguna
- Grup kueri
- Persentase memori WLM untuk digunakan
- Batas waktu WLM
- Melompat antrian kueri WLM
- Aturan pemantauan kueri

Mode penskalaan konkurensi

Saat penskalaan konkurensi diaktifkan, Amazon Redshift secara otomatis menambahkan kapasitas cluster tambahan saat Anda membutuhkannya untuk memproses peningkatan kueri baca dan tulis

bersamaan. Pengguna melihat data terbaru, apakah kueri berjalan di cluster utama atau pada cluster penskalaan konkurensi.

Anda mengelola kueri mana yang dikirim ke cluster penskalaan konkurensi dengan mengonfigurasi antrian WLM. Saat Anda mengaktifkan penskalaan konkurensi untuk antrian, kueri yang memenuhi syarat dikirim ke cluster penskalaan konkurensi alih-alih menunggu dalam antrian. Untuk informasi selengkapnya, lihat Penskalaan konkurensi.

Tingkat konkurensi

Kueri dalam antrian dijalankan secara bersamaan hingga mencapai jumlah slot kueri WLM, atau tingkat konkurensi, yang ditentukan untuk antrian tersebut. Kueri selanjutnya kemudian menunggu dalam antrian.

> **Note**
>
> Tingkat konkurensi WLM berbeda dari jumlah koneksi pengguna bersamaan yang dapat dibuat ke cluster. Untuk informasi selengkapnya, lihat Menghubungkan ke Cluster di Panduan Manajemen Amazon Redshift.

Dalam konfigurasi WLM otomatis, yang direkomendasikan, tingkat konkurensi diatur ke Otomatis. Amazon Redshift mengalokasikan memori secara dinamis ke kueri, yang kemudian menentukan berapa banyak yang akan dijalankan secara bersamaan. Ini didasarkan pada sumber daya yang diperlukan untuk kueri yang berjalan dan mengantri. Auto WLM tidak dapat dikonfigurasi. Untuk informasi selengkapnya, lihat Menerapkan WLM otomatis.

Dalam konfigurasi WLM manual, Amazon Redshift secara statis mengalokasikan jumlah memori tetap untuk setiap antrian. Memori antrian dibagi secara merata di antara slot query. Sebagai ilustrasi, jika antrian dialokasikan 20% dari memori cluster dan memiliki 10 slot, setiap kueri dialokasikan 2% dari memori cluster. Alokasi memori tetap tetap terlepas dari jumlah kueri yang berjalan secara bersamaan. Karena alokasi memori tetap ini, kueri yang berjalan seluruhnya di memori ketika jumlah slot adalah 5 mungkin menulis hasil perantara ke disk jika jumlah slot ditingkatkan menjadi 20. Dalam hal ini setiap bagian query dari memori antrian dikurangi dari 1/5th menjadi 1/20th. Disk tambahan I/O dapat menurunkan kinerja.

Jumlah slot maksimum di semua antrian yang ditentukan pengguna adalah 50. Ini membatasi total slot untuk semua antrian, termasuk antrian default. Satu-satunya antrian yang tidak tunduk pada batas adalah antrian superuser yang dicadangkan.

Secara default, antrian WLM manual memiliki tingkat konkurensi 5. Beban kerja Anda mungkin mendapat manfaat dari tingkat konkurensi yang lebih tinggi dalam kasus-kasus tertentu, seperti berikut ini:

- Jika banyak kueri kecil terpaksa menunggu kueri yang berjalan lama, buat antrian terpisah dengan jumlah slot yang lebih tinggi dan tetapkan kueri yang lebih kecil ke antrian itu. Antrian dengan tingkat konkurensi yang lebih tinggi memiliki lebih sedikit memori yang dialokasikan untuk setiap slot kueri, tetapi kueri yang lebih kecil membutuhkan lebih sedikit memori.

  > **Note**
  >
  > Jika Anda mengaktifkan akselerasi kueri pendek (SQA), WLM secara otomatis memprioritaskan kueri pendek daripada kueri yang berjalan lebih lama, sehingga Anda tidak memerlukan antrian terpisah untuk kueri pendek untuk sebagian besar alur kerja. Untuk informasi selengkapnya, lihat Akselerasi kueri singkat.

- Jika Anda memiliki beberapa kueri yang masing-masing mengakses data pada satu irisan, siapkan antrian WLM terpisah untuk menjalankan kueri tersebut secara bersamaan. Amazon Redshift menetapkan kueri bersamaan untuk irisan terpisah, yang memungkinkan beberapa kueri berjalan secara paralel pada beberapa irisan. Misalnya, jika kueri adalah agregat sederhana dengan predikat pada kunci distribusi, data untuk kueri terletak pada satu irisan.

Contoh WLM manual

Contoh ini adalah skenario WLM manual sederhana untuk menunjukkan bagaimana slot dan memori dapat dialokasikan. Anda menerapkan WLM manual dengan tiga antrian, yaitu sebagai berikut:

- antrian penyer apan data — Ini diatur untuk menelan data. Ini dialokasikan 20% dari memori cluster dan memiliki 5 slot. Selanjutnya, 5 kueri dapat berjalan secara bersamaan dalam antrian dan masing-masing dialokasikan 4% dari memori.
- antrian ilmuwan data — Ini dirancang untuk kueri intensif memori. Ini dialokasikan 40% dari memori cluster dan memiliki 5 slot. Selanjutnya, 5 kueri dapat berjalan secara bersamaan dan masing-masing dialokasikan 8% dari memori.
- antrian default — Ini dirancang untuk sebagian besar pengguna di organisasi. Ini termasuk grup penjualan dan akuntansi yang biasanya memiliki kueri berjalan pendek atau menengah yang tidak rumit. Ini dialokasikan 40% dari memori cluster dan memiliki 40 slot. 40 kueri dapat berjalan secara bersamaan dalam antrian ini, dengan setiap kueri dialokasikan 1% dari memori. Ini adalah jumlah

maksimum slot yang dapat dialokasikan untuk antrian ini karena di antara semua antrian batasnya adalah 50.

Jika Anda menjalankan WLM otomatis dan beban kerja Anda memerlukan lebih dari 15 kueri untuk dijalankan secara paralel, sebaiknya aktifkan penskalaan konkurensi. Ini karena meningkatkan jumlah slot kueri di atas 15 dapat menciptakan pertentangan untuk sumber daya sistem dan membatasi throughput keseluruhan dari satu cluster. Dengan penskalaan konkurensi, Anda dapat menjalankan ratusan kueri secara paralel, hingga jumlah cluster penskalaan konkurensi yang dikonfigurasi. Jumlah cluster penskalaan konkurensi dikendalikan olehklaster penskalaan maksimum. Untuk informasi selengkapnya tentang penskalaan konkurensi, lihatPenskalaan konkurensi.

Untuk informasi selengkapnya, lihat Peningkatan kinerja kueri.

Grup pengguna

Anda dapat menetapkan satu set grup pengguna ke antrian dengan menentukan setiap nama grup pengguna atau dengan menggunakan wildcard. Ketika anggota grup pengguna terdaftar menjalankan kueri, kueri tersebut berjalan dalam antrian yang sesuai. Tidak ada batasan yang ditetapkan pada jumlah grup pengguna yang dapat ditetapkan ke antrian. Untuk informasi selengkapnya, lihat Menetapkan kueri ke antrian berdasarkan grup pengguna.

Peran pengguna

Anda dapat menetapkan satu set peran pengguna ke antrian dengan menentukan setiap nama peran pengguna atau dengan menggunakan wildcard. Ketika anggota peran pengguna terdaftar menjalankan kueri, kueri tersebut berjalan dalam antrian yang sesuai. Tidak ada batasan yang ditetapkan pada jumlah peran pengguna yang dapat ditetapkan ke antrian. Untuk informasi selengkapnya, lihat Menetapkan kueri ke antrian berdasarkan peran pengguna.

Grup kueri

Anda dapat menetapkan kumpulan grup kueri ke antrian dengan menentukan setiap nama grup kueri atau dengan menggunakan wildcard. Grup query hanyalah sebuah label. Saat runtime, Anda dapat menetapkan label grup kueri ke serangkaian kueri. Setiap kueri yang ditetapkan ke grup kueri terdaftar dijalankan dalam antrian yang sesuai. Tidak ada batasan yang ditetapkan untuk jumlah grup kueri yang dapat ditetapkan ke antrian. Untuk informasi selengkapnya, lihat Menetapkan kueri ke grup kueri.

Wildcard

Jika wildcard diaktifkan dalam konfigurasi antrian WLM, Anda dapat menetapkan grup pengguna dan grup kueri ke antrian baik secara individual atau dengan menggunakan wildcard gaya shell Unix. Pencocokan pola tidak peka huruf besar/kecil.

Misalnya, karakter wildcard '*' cocok dengan sejumlah karakter. Jadi, jika Anda menambahkan `dba_*` ke daftar grup pengguna untuk antrian, kueri yang dijalankan pengguna yang termasuk dalam grup dengan nama yang dimulai dengan `dba_` ditetapkan ke antrian tersebut. Contohnya adalah `dba_admin` atau`DBA_primary`,. Yang '?' karakter wildcard cocok dengan karakter tunggal mana pun. Jadi, jika antrian menyertakan grup penggunadba?1, maka grup pengguna diberi `dba11` nama dan `dba21` cocok, tetapi `dba12` tidak cocok.

Karakter pengganti dimatikan secara default.

Persentase memori WLM untuk digunakan

Dalam konfigurasi WLM otomatis, persen memori diatur ke**`auto`**. Untuk informasi selengkapnya, lihat Menerapkan WLM otomatis.

Dalam konfigurasi WLM manual, untuk menentukan jumlah memori yang tersedia yang dialokasikan untuk kueri, Anda dapat mengatur `WLM Memory Percent to Use` parameter. Secara default, setiap antrian yang ditentukan pengguna dialokasikan porsi yang sama dari memori yang tersedia untuk kueri yang ditentukan pengguna. Misalnya, jika Anda memiliki empat antrian yang ditentukan pengguna, setiap antrian dialokasikan 25 persen dari memori yang tersedia. Antrian superuser memiliki memori yang dialokasikan sendiri dan tidak dapat dimodifikasi. Untuk mengubah alokasi, Anda menetapkan persentase integer memori untuk setiap antrian, hingga total 100 persen. Memori yang tidak terisi dikelola oleh Amazon Redshift dan dapat diberikan sementara ke antrian jika antrian meminta memori tambahan untuk diproses.

Misalnya, jika Anda mengonfigurasi empat antrian, Anda dapat mengalokasikan memori sebagai berikut: 20 persen, 30 persen, 15 persen, 15 persen. Sisanya 20 persen tidak dialokasikan dan dikelola oleh layanan.

Batas waktu WLM

WLM timeout (`max_execution_time`) tidak digunakan lagi. Sebagai gantinya, buat aturan pemantauan kueri (QMR) menggunakan `query_execution_time` untuk membatasi waktu eksekusi yang telah berlalu untuk kueri. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

Untuk membatasi jumlah waktu kueri dalam antrian WLM tertentu diizinkan untuk digunakan, Anda dapat mengatur nilai batas waktu WLM untuk setiap antrian. Parameter batas waktu menentukan jumlah waktu, dalam milidetik, dimana Amazon Redshift menunggu kueri dijalankan sebelum membatalkan atau melompati kueri. Waktu tunggu didasarkan pada waktu eksekusi kueri dan tidak termasuk waktu yang dihabiskan menunggu dalam antrian.

WLM mencoba untuk melompat CREATE TABLE AS (CTAS) pernyataan dan kueri read-only, seperti pernyataan SELECT. Kueri yang tidak dapat dilompati dibatalkan. Untuk informasi selengkapnya, lihat Melompat antrian kueri WLM.

Waktu tunggu WLM tidak berlaku untuk kueri yang telah mencapai status pengembalian. Untuk melihat status kueri, lihat tabel STV_WLM_QUERY_STATE sistem. Pernyataan COPY dan operasi pemeliharaan, seperti ALTER, ANALYZE dan VACUUM, tidak tunduk pada batas waktu WLM.

Fungsi batas waktu WLM mirip dengan parameter statement_timeout konfigurasi. Perbedaannya adalah, di mana parameter `statement_timeout` konfigurasi berlaku untuk seluruh cluster, batas waktu WLM khusus untuk satu antrian dalam konfigurasi WLM.

Jika juga statement_timeout ditentukan, yang lebih rendah dari statement_timeout dan batas waktu WLM (max_execution_time) digunakan.

Aturan pemantauan kueri

Aturan pemantauan kueri menentukan batas kinerja berbasis metrik untuk antrian WLM dan menentukan tindakan apa yang harus diambil ketika kueri melampaui batas tersebut. Misalnya, untuk antrian yang didedikasikan untuk kueri berjalan singkat, Anda dapat membuat aturan yang membatalkan kueri yang berjalan selama lebih dari 60 detik. Untuk melacak kueri yang dirancang dengan buruk, Anda mungkin memiliki aturan lain yang mencatat kueri yang berisi loop bersarang. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

Melompat antrian kueri WLM

Dengan Amazon Redshift, Anda dapat mengelola konkurensi beban kerja dan alokasi sumber daya dengan mengaktifkan lompatan antrian kueri WLM (Manajemen Beban Kerja). Fitur ini memungkinkan kueri untuk sementara “melompat” dari antrian yang ditetapkan ke antrian prioritas yang lebih tinggi ketika sumber daya tersedia, meningkatkan kinerja kueri secara keseluruhan dan pemanfaatan sistem. Bagian berikut memberikan panduan terperinci tentang mengonfigurasi dan memanfaatkan lompatan antrian kueri WLM di Amazon Redshift.

Kueri dapat dilompati karena batas waktu WLM atau tindakan hop aturan pemantauan kueri (QMR). Anda hanya dapat melompat kueri dalam konfigurasi WLM manual.

Saat kueri dilompat, WLM mencoba merutekan kueri ke antrian pencocokan berikutnya berdasarkan aturan penetapan antrian WLM. Jika kueri tidak cocok dengan definisi antrian lainnya, kueri dibatalkan. Itu tidak ditetapkan ke antrian default.

Tindakan batas waktu WLM

Tabel berikut merangkum perilaku berbagai jenis kueri dengan batas waktu WLM.

Jenis kueri	Tindakan
MASUKKAN, PERBARUI, dan HAPUS	Batalkan
User-defined fungsi (UDF)	Batalkan
MEMBONGKAR	Batalkan
MENYALIN	Lanjutkan eksekusi
Operasi pemeliharaan	Lanjutkan eksekusi
Read-only kueri dalam suatu `returning` negara	Lanjutkan eksekusi
Read-only kueri dalam suatu `running` negara	Tetapkan kembali atau mulai ulang
BUAT TABEL SEBAGAI (CTAS), PILIH KE	Tetapkan kembali atau mulai ulang

Melompat antrian batas waktu WLM

WLM melompat jenis kueri berikut saat waktunya habis:

- Read-only kueri, seperti pernyataan SELECT, yang berada dalam status WLM. `running` Untuk menemukan status WLM dari kueri, lihat kolom STATE pada tabel STV_WLM_QUERY_STATE sistem.
- BUAT TABLE AS (CTAS) pernyataan. Lompat antrian WLM mendukung pernyataan CTAS yang ditentukan pengguna dan yang dihasilkan sistem.

- PILIH PERNYATAAN INTO.

Kueri yang tidak tunduk pada batas waktu WLM terus berjalan di antrian asli hingga selesai. Jenis kueri berikut tidak tunduk pada batas waktu WLM:

- Pernyataan COPY
- Operasi pemeliharaan, seperti ALTER, ANALYSIS dan VACUUM
- Read-only kueri, seperti pernyataan SELECT, yang telah mencapai status WLM. `returning` Untuk menemukan status WLM dari kueri, lihat kolom STATE pada tabel STV_WLM_QUERY_STATE sistem.

Kueri yang tidak memenuhi syarat untuk lompat dengan batas waktu WLM dibatalkan saat waktunya habis. Jenis kueri berikut tidak memenuhi syarat untuk dilompat dengan batas waktu WLM:

- Pernyataan INSERT, UPDATE, dan DELETE
- Pernyataan UNLOAD
- User-defined fungsi (UDF)

Waktu tunggu WLM ditugaskan kembali dan memulai ulang kueri

Ketika kueri dilompati dan tidak ada antrian yang cocok ditemukan, kueri dibatalkan.

Ketika kueri dilompati dan antrian yang cocok ditemukan, WLM mencoba menetapkan kembali kueri ke antrian baru. Jika kueri tidak dapat ditugaskan kembali, kueri akan dimulai ulang dalam antrian baru, seperti yang dijelaskan berikut.

Kueri ditugaskan kembali hanya jika semua hal berikut benar:

- Antrian yang cocok ditemukan.
- Antrian baru memiliki cukup slot gratis untuk menjalankan kueri. Kueri mungkin memerlukan beberapa slot jika wlm_query_slot_count parameter disetel ke nilai yang lebih besar dari 1.
- Antrian baru memiliki setidaknya sebanyak memori yang tersedia seperti yang digunakan kueri saat ini.

Jika kueri ditugaskan kembali, kueri terus dieksekusi dalam antrian baru. Hasil menengah dipertahankan, sehingga ada efek minimal pada total waktu eksekusi.

Jika kueri tidak dapat ditugaskan kembali, kueri dibatalkan dan dimulai ulang dalam antrian baru. Hasil menengah dihapus. Kueri menunggu dalam antrian, lalu mulai berjalan ketika cukup slot tersedia.

Tindakan hop QMR

Tabel berikut merangkum perilaku berbagai jenis kueri dengan tindakan hop QMR.

Jenis kueri	Tindakan
MENYALIN	Lanjutkan eksekusi
Operasi pemeliharaan	Lanjutkan eksekusi
User-defined fungsi (UDF)	Lanjutkan eksekusi
MEMBONGKAR	Tetapkan kembali atau lanjutkan eksekusi
MASUKKAN, PERBARUI, dan HAPUS	Tetapkan kembali atau lanjutkan eksekusi
Read-only kueri dalam suatu `returning` negara	Tetapkan kembali atau lanjutkan eksekusi
Read-only kueri dalam suatu `running` negara	Tetapkan kembali atau mulai ulang
BUAT TABEL SEBAGAI (CTAS), PILIH KE	Tetapkan kembali atau mulai ulang

Untuk mengetahui apakah kueri yang dilompati oleh QMR ditugaskan kembali, dimulai ulang, atau dibatalkan, kueri tabel log sistem. STL_WLM_RULE_ACTION

Tindakan hop QMR ditugaskan ulang dan kueri dimulai ulang

Ketika kueri dilompati dan tidak ada antrian yang cocok ditemukan, kueri dibatalkan.

Ketika kueri dilompati dan antrian yang cocok ditemukan, WLM mencoba menetapkan kembali kueri ke antrian baru. Jika kueri tidak dapat ditugaskan kembali, kueri tersebut akan dimulai ulang di antrian baru atau melanjutkan eksekusi dalam antrian asli, seperti yang dijelaskan berikut.

Kueri ditugaskan kembali hanya jika semua hal berikut benar:

- Antrian yang cocok ditemukan.

- Antrian baru memiliki cukup slot gratis untuk menjalankan kueri. Kueri mungkin memerlukan beberapa slot jika wlm_query_slot_count parameter disetel ke nilai yang lebih besar dari 1.
- Antrian baru memiliki setidaknya sebanyak memori yang tersedia seperti yang digunakan kueri saat ini.

Jika kueri ditugaskan kembali, kueri terus dieksekusi dalam antrian baru. Hasil menengah dipertahankan, sehingga ada efek minimal pada total waktu eksekusi.

Jika kueri tidak dapat ditugaskan kembali, kueri akan dimulai ulang atau melanjutkan eksekusi dalam antrian asli. Jika kueri dimulai ulang, kueri dibatalkan dan dimulai kembali dalam antrian baru. Hasil menengah dihapus. Kueri menunggu dalam antrian, kemudian memulai eksekusi ketika cukup slot tersedia.

Tutorial: Mengkonfigurasi antrian manajemen beban kerja manual (WLM)

Dengan Amazon Redshift, Anda dapat mengonfigurasi antrian manajemen beban kerja manual (WLM) untuk memprioritaskan dan mengalokasikan sumber daya untuk berbagai jenis kueri dan pengguna. Antrian WLM manual memungkinkan Anda mengontrol pengaturan memori dan konkurensi untuk antrian tertentu, memastikan bahwa beban kerja kritis menerima sumber daya yang diperlukan sambil mencegah kueri prioritas rendah memonopoli sistem. Bagian berikut memandu Anda melalui proses pembuatan dan konfigurasi antrian WLM manual di Amazon Redshift untuk memenuhi persyaratan manajemen beban kerja Anda.

Gambaran umum

Sebaiknya konfigurasi manajemen beban kerja otomatis (WLM) di Amazon Redshift. Untuk informasi selengkapnya tentang WLM otomatis, lihatManajemen beban kerja. Namun, jika Anda memerlukan beberapa antrian WLM, tutorial ini memandu Anda melalui proses mengonfigurasi manajemen beban kerja manual (WLM) di Amazon Redshift. Dengan mengonfigurasi WLM manual, Anda dapat meningkatkan kinerja kueri dan alokasi sumber daya di cluster Anda.

Amazon Redshift merutekan kueri pengguna ke antrian untuk diproses. WLM mendefinisikan bagaimana kueri tersebut dialihkan ke antrian. Secara default, Amazon Redshift memiliki dua antrian yang tersedia untuk kueri: satu untuk pengguna super, dan satu untuk pengguna. Antrian superuser tidak dapat dikonfigurasi dan hanya dapat memproses satu kueri pada satu waktu. Anda harus memesan antrian ini untuk tujuan pemecahan masalah saja. Antrian pengguna dapat memproses hingga lima kueri sekaligus, tetapi Anda dapat mengonfigurasinya dengan mengubah tingkat konkurensi antrian jika diperlukan.

Ketika Anda memiliki beberapa pengguna yang menjalankan kueri terhadap database, Anda mungkin menemukan konfigurasi lain yang lebih efisien. Misalnya, jika beberapa pengguna menjalankan operasi padat sumber daya, seperti VACUUM, ini mungkin berdampak negatif pada kueri yang kurang intensif, seperti laporan. Anda dapat mempertimbangkan untuk menambahkan antrian tambahan dan mengonfigurasinya untuk beban kerja yang berbeda.

Perkiraan waktu: 75 menit

Perkiraan biaya: 50 sen

Prasyarat

Anda memerlukan cluster Amazon Redshift, database TICKIT sampel, dan alat klien Amazon Redshift RSQL. Jika Anda belum menyiapkan ini, buka Panduan Memulai Amazon Redshift dan Amazon Redshift RSQL.

Bagian-bagian

- Bagian 1: Memahami perilaku pemrosesan antrian default
- Bagian 2: Memodifikasi konfigurasi antrian kueri WLM
- Bagian 3: Merutekan kueri ke antrian berdasarkan grup pengguna dan grup kueri
- Bagian 4: Menggunakan wlm_query_slot_count untuk sementara mengganti level konkurensi dalam antrian
- Bagian 5: Membersihkan sumber daya Anda

Bagian 1: Memahami perilaku pemrosesan antrian default

Sebelum Anda mulai mengonfigurasi WLM manual, penting untuk memahami perilaku default pemrosesan antrian di Amazon Redshift. Di bagian ini, Anda membuat dua tampilan database yang mengembalikan informasi dari beberapa tabel sistem. Kemudian Anda menjalankan beberapa kueri uji untuk melihat bagaimana kueri dirutekan secara default. Untuk informasi selengkapnya tentang tabel sistem, lihatTabel sistem dan referensi tampilan.

Langkah 1: Buat tampilan WLM_QUEUE_STATE_VW

Pada langkah ini, Anda membuat tampilan yang disebut WLM_QUEUE_STATE_VW. Tampilan ini mengembalikan informasi dari tabel sistem berikut.

- STV_WLM_CLASSICATION_CONFIG

- STV_WLM_SERVICE_CLASS_CONFIG
- STV_WLM_SERVICE_CLASS_STATE

Anda menggunakan tampilan ini di seluruh tutorial untuk memantau apa yang terjadi pada antrian setelah Anda mengubah konfigurasi WLM. Tabel berikut menjelaskan data yang dikembalikan tampilan WLM_QUEUE_STATE_VW.

Kolom	Deskripsi
antrean	Nomor yang terkait dengan baris yang mewakili antrian. Nomor antrian menentukan urutan antrian dalam database.
deskripsi	Nilai yang menjelaskan apakah antrian hanya tersedia untuk grup pengguna tertentu, grup kueri tertentu, atau semua jenis kueri.
slot	Jumlah slot yang dialokasikan untuk antrian.
memek	Jumlah memori, dalam MB per slot, dialokasikan ke antrian.
waktu_eksekusi maksimal	Jumlah waktu kueri diizinkan untuk dijalankan sebelum diakhiri.
pengguna_*	Nilai yang menunjukkan apakah karakter wildcard diperbolehkan dalam konfigurasi WLM untuk mencocokkan grup pengguna.
pertanyaan_*	Nilai yang menunjukkan apakah karakter wildcard diperbolehkan dalam konfigurasi WLM untuk mencocokkan grup kueri.
mengantri	Jumlah query yang menunggu dalam antrian untuk diproses.
mengeksekusi	Jumlah query yang saat ini sedang berjalan.
dieksekusi	Jumlah query yang telah dijalankan.

Untuk membuat tampilan WLM_QUEUE_STATE_VW

1. Buka Amazon Redshift RSQL dan sambungkan ke database sampel TICKIT Anda. Jika Anda tidak memiliki database ini, lihatPrasyarat.

2. Jalankan query berikut untuk membuat tampilan WLM_QUEUE_STATE_VW.

```
create view WLM_QUEUE_STATE_VW as
select (config.service_class-5) as queue
, trim (class.condition) as description
, config.num_query_tasks as slots
, config.query_working_mem as mem
, config.max_execution_time as max_time
, config.user_group_wild_card as "user_*"
, config.query_group_wild_card as "query_*"
, state.num_queued_queries queued
, state.num_executing_queries executing
, state.num_executed_queries executed
from
STV_WLM_CLASSIFICATION_CONFIG class,
STV_WLM_SERVICE_CLASS_CONFIG config,
STV_WLM_SERVICE_CLASS_STATE state
where
class.action_service_class = config.service_class
and class.action_service_class = state.service_class
and config.service_class > 4
order by config.service_class;
```

3. Jalankan kueri berikut untuk melihat informasi yang berisi tampilan.

```
select * from wlm_queue_state_vw;
```

Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (querytype:any)                           |     5 | 836 |        0 |  false |
 false   |      0 |         1 |      160
```

Langkah 2: Buat tampilan WLM_QUERY_STATE_VW

Pada langkah ini, Anda membuat tampilan yang disebut WLM_QUERY_STATE_VW. Tampilan ini mengembalikan informasi dari tabel STV_WLM_QUERY_STATE sistem.

Anda menggunakan tampilan ini di seluruh tutorial untuk memantau kueri yang sedang berjalan. Tabel berikut menjelaskan data yang dikembalikan tampilan WLM_QUERY_STATE_VW.

Kolom	Deskripsi
kueri	ID kueri.
antrean	Nomor antrian.
Slot_count	Jumlah slot yang dialokasikan untuk kueri.
waktu mulai	Waktu permintaan dimulai.
status	Status query, seperti mengeksekusi.
antrian_waktu	Jumlah mikrodetik yang telah dihabiskan kueri dalam antrian.
waktu eksekusi	Jumlah mikrodetik yang telah dijalankan kueri.

Untuk membuat tampilan WLM_QUERY_STATE_VW

1. Di RSQL, jalankan query berikut untuk membuat tampilan WLM_QUERY_STATE_VW.

```
create view WLM_QUERY_STATE_VW as
select query, (service_class-5) as queue, slot_count, trim(wlm_start_time) as
 start_time, trim(state) as state, trim(queue_time) as queue_time, trim(exec_time) as
 exec_time
from stv_wlm_query_state;
```

2. Jalankan kueri berikut untuk melihat informasi yang berisi tampilan.

```
select * from wlm_query_state_vw;
```

   Berikut ini adalah contoh hasil.

```
query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 1249 |     1 |          1 | 2014-09-24 22:19:16 | Executing | 0          | 516
```

Langkah 3: Jalankan kueri uji

Pada langkah ini, Anda menjalankan kueri dari beberapa koneksi di RSQL dan meninjau tabel sistem untuk menentukan bagaimana kueri dirutekan untuk diproses.

Untuk langkah ini, Anda perlu membuka dua jendela RSQL:

- Di jendela RSQL 1, Anda menjalankan kueri yang memantau keadaan antrian dan kueri menggunakan tampilan yang sudah Anda buat dalam tutorial ini.
- Di jendela RSQL 2, Anda menjalankan kueri yang berjalan lama untuk mengubah hasil yang Anda temukan di jendela RSQL 1.

Untuk menjalankan kueri uji

1. Buka dua jendela RSQL. Jika Anda sudah memiliki satu jendela terbuka, Anda hanya perlu membuka jendela kedua. Anda dapat menggunakan akun pengguna yang sama untuk kedua koneksi ini.
2. Di jendela RSQL 1, jalankan query berikut.

   ```
   select * from wlm_query_state_vw;
   ```

   Berikut ini adalah contoh hasil.

   ```
   query | queue | slot_count | start_time          | state     | queue_time | exec_time
   ------+-------+------------+---------------------+-----------+------------
   +-----------
    1258 |     1 |          1 | 2014-09-24 22:21:03 | Executing | 0          | 549
   ```

   Kueri ini mengembalikan hasil referensial diri. Kueri yang sedang berjalan adalah pernyataan SELECT dari tampilan ini. Kueri pada tampilan ini selalu mengembalikan setidaknya satu hasil. Bandingkan hasil ini dengan hasil yang terjadi setelah memulai kueri yang berjalan lama di langkah berikutnya.

3. Di jendela RSQL 2, jalankan kueri dari database sampel TICKIT. Kueri ini harus berjalan selama kurang lebih satu menit sehingga Anda memiliki waktu untuk menjelajahi hasil tampilan WLM_QUEUE_STATE_VW dan tampilan WLM_QUERY_STATE_VW yang Anda buat sebelumnya. Dalam beberapa kasus, Anda mungkin menemukan bahwa kueri tidak berjalan cukup lama bagi Anda untuk menanyakan kedua tampilan. Dalam kasus ini, Anda dapat meningkatkan nilai filter `l.listid` agar berjalan lebih lama.

   > **Note**
   >
   > Untuk mengurangi waktu eksekusi kueri dan meningkatkan kinerja sistem, Amazon Redshift menyimpan hasil jenis kueri tertentu dalam memori pada node pemimpin. Ketika hasil caching diaktifkan, kueri berikutnya berjalan jauh lebih cepat. Untuk mencegah kueri berjalan dengan cepat, nonaktifkan cache hasil untuk sesi saat ini.

   Untuk mematikan cache hasil untuk sesi saat ini, atur mengaktifkan_hasil_cache_untuk_sesi parameter ke`off`, seperti yang ditunjukkan berikut ini.

   ```
   set enable_result_cache_for_session to off;
   ```

   Di jendela RSQL 2, jalankan query berikut.

   ```
   select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <
    100000;
   ```

4. Di jendela RSQL 1, kueri WLM_QUEUE_STATE_VW dan WLM_QUERY_STATE_VW dan bandingkan hasilnya dengan hasil sebelumnya.

   ```
   select * from wlm_queue_state_vw;
   select * from wlm_query_state_vw;
   ```

   Berikut ini adalah contoh hasil.

   ```
   query | description                                | slots | mem | max_time | user_* |
    query_* | queued | executing | executed
   ------+--------------------------------------------+-------+-----+----------+--------
   +---------+--------+-----------+----------
       0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
    false   |      0 |         0 |        0
   ```

```
    1 | (querytype:any)                          |     5 | 836 |        0 |  false |
 false   |      0 |         2 |      163

query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 1267 |     1 |          1 | 2014-09-24 22:22:30 | Executing | 0          | 684
 1265 |     1 |          1 | 2014-09-24 22:22:36 | Executing | 0          | 4080859
```

Perhatikan perbedaan berikut antara kueri Anda sebelumnya dan hasil dalam langkah ini:

- Ada dua baris sekarang di WLM_QUERY_STATE_VW. Salah satu hasilnya adalah kueri referensial diri untuk menjalankan operasi SELECT pada tampilan ini. Hasil kedua adalah kueri yang berjalan lama dari langkah sebelumnya.
- Kolom eksekusi di WLM_QUEUE_STATE_VW telah meningkat dari 1 menjadi 2. Entri kolom ini berarti bahwa ada dua kueri yang berjalan dalam antrian.
- Kolom yang dieksekusi bertambah setiap kali Anda menjalankan kueri dalam antrian.

Tampilan WLM_QUEUE_STATE_VW berguna untuk mendapatkan tampilan keseluruhan antrian dan berapa banyak kueri yang sedang diproses di setiap antrian. Tampilan WLM_QUERY_STATE_VW berguna untuk mendapatkan tampilan yang lebih rinci dari kueri individu yang saat ini sedang berjalan.

Bagian 2: Memodifikasi konfigurasi antrian kueri WLM

Sekarang setelah Anda memahami cara kerja antrian secara default, Anda dapat mempelajari cara mengonfigurasi antrian kueri menggunakan WLM manual. Di bagian ini, Anda membuat dan mengkonfigurasi grup parameter baru untuk cluster Anda. Anda membuat dua antrian pengguna tambahan dan mengonfigurasinya untuk menerima kueri berdasarkan grup pengguna kueri atau label grup kueri. Setiap kueri yang tidak dirutekan ke salah satu dari dua antrian ini dialihkan ke antrian default saat runtime.

Untuk membuat konfigurasi WLM manual dalam grup parameter

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Konfigurasi, lalu pilih Manajemen beban kerja untuk menampilkan halaman manajemen beban kerja.

3. Pilih Buat untuk menampilkan jendela Create parameter group.
4. Masukkan **`WLMTutorial`** nama grup Parameter dan Deskripsi, lalu pilih Buat untuk membuat grup parameter.

   > **Note**
   >
   > Nama grup Parameter dikonversi ke semua format huruf kecil saat dibuat.

5. Pada halaman manajemen beban kerja, pilih grup parameter **`wlmtutorial`** untuk menampilkan halaman detail dengan tab untuk Parameter dan manajemen beban kerja.
6. Konfirmasikan bahwa Anda berada di tab Manajemen beban kerja, lalu pilih Beralih mode WLM untuk menampilkan jendela pengaturan Konkurensi.
7. Pilih Manual WLM, lalu pilih S impan untuk beralih ke WLM manual.
8. Pilih Edit antrian beban kerja.
9. Pilih Tambahkan antrean dua kali untuk menambahkan dua antrian. Sekarang ada tiga antrian: Antri an 1, Antri an 2, dan Antrian Default.
10. Masukkan informasi untuk setiap antrian sebagai berikut:

    - Untuk Antrian 1, masukkan **`30`** untuk Memori (%), **`2`** untuk Konkurensi pada main, dan **`test`** untuk grup Ku eri. Tinggalkan pengaturan lain dengan nilai defaultnya.
    - Untuk Antrian 2, masukkan **`40`** untuk Memori (%), **`3`** untuk Konkurensi pada main, dan **`admin`** untuk Grup Pengguna. Tinggalkan pengaturan lain dengan nilai defaultnya.
    - Tetapkan Konkurensi pada nilai utama untuk antrian default ke nilai yang lebih besar dari atau sama dengan 1. Jangan membuat perubahan lain pada antrian Default. WLM menetapkan memori yang tidak terisi ke antrian default.

11. Pilih Simpan untuk menyimpan pengaturan Anda.

Selanjutnya, kaitkan grup parameter yang memiliki konfigurasi WLM manual dengan cluster.

Untuk mengaitkan grup parameter dengan konfigurasi WLM manual dengan cluster

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Redshift di https://console.aws.amazon.com/redshiftv2/.
2. Pada menu navigasi, pilih Cluster, lalu pilih Cluster untuk menampilkan daftar cluster Anda.

3. Pilih cluster Anda, seperti `examplecluster` untuk menampilkan detail cluster. Kemudian pilih tab Properties untuk menampilkan properti cluster itu.
4. Di bagian Konfigurasi database, pilih Edit, Edit grup parameter untuk menampilkan jendela grup parameter.
5. Untuk grup Parameter pilih grup **`wlmtutorial`** parameter yang Anda buat sebelumnya.
6. Pilih Simpan perubahan untuk mengaitkan grup parameter.

   Cluster dimodifikasi dengan grup parameter yang diubah. Namun, Anda perlu me-reboot cluster agar perubahan juga diterapkan ke database.
7. Pilih cluster Anda, lalu pilih Reboot untuk Tind akan.

Setelah cluster di-boot ulang, statusnya kembali ke Tersedia.

Bagian 3: Merutekan kueri ke antrian berdasarkan grup pengguna dan grup kueri

Sekarang Anda memiliki cluster Anda terkait dengan grup parameter baru dan Anda telah mengkonfigurasi WLM. Selanjutnya, jalankan beberapa kueri untuk melihat bagaimana Amazon Redshift merutekan kueri ke dalam antrian untuk diproses.

Langkah 1: Lihat konfigurasi antrian kueri di database

Pertama, verifikasi bahwa database memiliki konfigurasi WLM yang Anda harapkan.

Untuk melihat konfigurasi antrian kueri

1. Buka RSQL dan jalankan query berikut. Kueri menggunakan tampilan WLM_QUEUE_STATE_VW yang Anda buat. Langkah 1: Buat tampilan WLM_QUEUE_STATE_VW Jika Anda sudah memiliki sesi yang terhubung ke database sebelum reboot cluster, Anda perlu menyambung kembali.

   ```
   select * from wlm_queue_state_vw;
   ```

   Berikut ini adalah contoh hasil.

   ```
   query | description                                | slots | mem | max_time | user_* |
    query_* | queued | executing | executed
   ------+--------------------------------------------+-------+-----+----------+--------
   +---------+--------+-----------+----------
       0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
    false   |      0 |         0 |        0
   ```

```
   1 | (query group: test)                        |     2 | 627 |        0 |  false |
false   |      0 |         0 |        0
   2 | (suser group: admin)                       |     3 | 557 |        0 |  false |
false   |      0 |         0 |        0
   3 | (querytype:any)                            |     5 | 250 |        0 |  false |
false   |      0 |         1 |        0
```

Bandingkan hasil ini dengan hasil yang Anda terimaLangkah 1: Buat tampilan WLM_QUEUE_STATE_VW. Perhatikan bahwa sekarang ada dua antrian tambahan. Antrian 1 sekarang menjadi antrian untuk grup kueri uji, dan antrian 2 adalah antrian untuk grup pengguna admin.

Antrian 3 sekarang menjadi antrian default. Antrian terakhir dalam daftar selalu merupakan antrian default. Itulah antrian ke mana kueri dirutekan secara default jika tidak ada grup pengguna atau grup kueri yang ditentukan dalam kueri.

2. Jalankan kueri berikut untuk mengonfirmasi bahwa kueri Anda sekarang berjalan di antrian 3.

```
select * from wlm_query_state_vw;
```

Berikut ini adalah contoh hasil.

```
query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 2144 |     3 |          1 | 2014-09-24 23:49:59 | Executing | 0          | 550430
```

Langkah 2: Jalankan kueri menggunakan antrian grup kueri

Untuk menjalankan kueri menggunakan antrian grup kueri

1. Jalankan kueri berikut untuk merutekan ke `test` grup kueri.

```
set query_group to test;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. Dari jendela RSQL lainnya, jalankan query berikut.

```
select * from wlm_query_state_vw;
```

Berikut ini adalah contoh hasil.

```
query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 2168 |     1 |          1 | 2014-09-24 23:54:18 | Executing | 0          | 6343309
 2170 |     3 |          1 | 2014-09-24 23:54:24 | Executing | 0          | 847
```

Kueri diarahkan ke grup kueri uji, yang sekarang adalah antrian 1.

3. Pilih semua dari tampilan status antrian.

```
select * from wlm_queue_state_vw;
```

Anda melihat hasil yang mirip dengan berikut ini.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      0 |         1 |        0
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         0 |        0
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         1 |        0
```

4. Sekarang, atur ulang grup kueri dan jalankan kueri panjang lagi:

```
reset query_group;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

5. Jalankan kueri terhadap tampilan untuk melihat hasilnya.

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      0 |         0 |        1
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         0 |        0
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         2 |        5

query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 2186 |     3 |          1 | 2014-09-24 23:57:52 | Executing | 0          | 649
 2184 |     3 |          1 | 2014-09-24 23:57:48 | Executing | 0          | 4137349
```

Hasilnya seharusnya kueri sekarang berjalan di antrian 3 lagi.

Langkah 3: Buat pengguna dan grup database

Sebelum Anda dapat menjalankan kueri dalam antrian ini, Anda perlu membuat grup pengguna di database dan menambahkan pengguna ke grup. Kemudian Anda masuk dengan RSQL menggunakan kredenSIAL pengguna baru dan menjalankan kueri. Anda perlu menjalankan query sebagai superuser, seperti pengguna admin, untuk membuat pengguna database.

Untuk membuat pengguna database baru dan grup pengguna

1. Dalam database, buat pengguna database baru bernama `adminwlm` dengan menjalankan perintah berikut di jendela RSQL.

```
create user adminwlm createuser password '123Admin';
```

2. Kemudian, jalankan perintah berikut untuk membuat grup pengguna baru dan menambahkan `adminwlm` pengguna baru Anda ke dalamnya.

```
create group admin;
alter group admin add user adminwlm;
```

Langkah 4: Jalankan kueri menggunakan antrean grup pengguna

Selanjutnya Anda menjalankan kueri dan merutekan ke antrian grup pengguna. Anda melakukan ini ketika Anda ingin merutekan kueri Anda ke antrian yang dikonfigurasi untuk menangani jenis kueri yang ingin Anda jalankan.

Untuk menjalankan kueri menggunakan antrean grup pengguna

1. Di jendela RSQL 2, jalankan kueri berikut untuk beralih ke `adminwlm` akun dan menjalankan kueri sebagai pengguna itu.

```
set session authorization 'adminwlm';
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. Di jendela RSQL 1, jalankan kueri berikut untuk melihat antrian kueri yang diarahkan ke kueri.

```
select * from wlm_query_state_vw;
select * from wlm_queue_state_vw;
```

   Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      0 |         0 |        1
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         1 |        0
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         1 |        8

query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 2202 |     2 |          1 | 2014-09-25 00:01:38 | Executing | 0          | 4885796
 2204 |     3 |          1 | 2014-09-25 00:01:43 | Executing | 0          | 650
```

Antrian yang dijalankan kueri ini adalah antrian 2, antrian `admin` pengguna. Setiap kali Anda menjalankan kueri yang masuk sebagai pengguna ini, mereka berjalan dalam antrian 2 kecuali Anda menentukan grup kueri yang berbeda untuk digunakan. Antrian yang dipilih tergantung pada aturan penetapan antrian. Untuk informasi selengkapnya, lihat Aturan penetapan antrian WLM.

3. Sekarang jalankan query berikut dari jendela RSQL 2.

```
set query_group to test;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

4. Di jendela RSQL 1, jalankan kueri berikut untuk melihat antrian kueri yang diarahkan ke kueri.

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      0 |         1 |        1
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         0 |        1
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         1 |       10

query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 2218 |     1 |          1 | 2014-09-25 00:04:30 | Executing | 0          | 4819666
 2220 |     3 |          1 | 2014-09-25 00:04:35 | Executing | 0          | 685
```

5. Setelah selesai, atur ulang grup kueri.

```
reset query_group;
```

Bagian 4: Menggunakan wlm_query_slot_count untuk sementara mengganti level konkurensi dalam antrian

Terkadang, pengguna mungkin sementara membutuhkan lebih banyak sumber daya untuk kueri tertentu. Jika demikian, mereka dapat menggunakan pengaturan konfigurasi wlm_query_slot_count untuk sementara mengganti cara slot dialokasikan dalam antrian kueri. Slot adalah unit memori dan CPU yang digunakan untuk memproses query. Anda mungkin mengganti jumlah slot ketika Anda memiliki kueri sesekali yang membutuhkan banyak sumber daya di cluster, seperti saat Anda melakukan operasi VACUUM dalam database.

Anda mungkin menemukan bahwa pengguna sering perlu mengatur wlm_query_slot_count untuk jenis kueri tertentu. Jika demikian, pertimbangkan untuk menyesuaikan konfigurasi WLM dan memberi pengguna antrian yang lebih sesuai dengan kebutuhan kueri mereka. Untuk informasi selengkapnya tentang mengganti sementara level konkurensi dengan menggunakan jumlah slot, lihat. wlm_query_slot_count

Langkah 1: Ganti level konkurensi menggunakan wlm_query_slot_count

Untuk keperluan tutorial ini, jalankan kueri SELECT yang berjalan lama yang sama. Jalankan sebagai `adminwlm` pengguna dengan menggunakan wlm_query_slot_count untuk meningkatkan jumlah slot yang tersedia untuk kueri.

Untuk mengganti level konkurensi menggunakan wlm_query_slot_count

1. Tingkatkan batas pada kueri untuk memastikan bahwa Anda memiliki cukup waktu untuk menanyakan tampilan WLM_QUERY_STATE_VW dan melihat hasilnya.

   ```
   set wlm_query_slot_count to 3;
   select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
   ```

2. Sekarang, kueri WLM_QUERY_STATE_VW dengan pengguna admin untuk melihat bagaimana kueri berjalan.

   ```
   select * from wlm_query_state_vw;
   ```

   Berikut ini adalah contoh hasil.

   ```
   query | queue | slot_count | start_time          | state     | queue_time | exec_time
   ------+-------+------------+---------------------+-----------+------------
   +-----------
   ```

```
 2240 |     2 |          1 | 2014-09-25 00:08:45 | Executing | 0          | 3731414
 2242 |     3 |          1 | 2014-09-25 00:08:49 | Executing | 0          | 596
```

Perhatikan bahwa jumlah slot untuk kueri adalah 3. Hitungan ini berarti bahwa kueri menggunakan ketiga slot untuk memproses kueri, mengalokasikan semua sumber daya dalam antrian ke kueri itu.

3. Sekarang, jalankan query berikut.

```
select * from WLM_QUEUE_STATE_VW;
```

Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      0 |         0 |        4
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         1 |        3
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         1 |       25
```

Pengaturan konfigurasi wlm_query_slot_count hanya valid untuk sesi saat ini. Jika sesi itu berakhir, atau pengguna lain menjalankan kueri, konfigurasi WLM digunakan.

4. Setel ulang jumlah slot dan jalankan kembali tes.

```
reset wlm_query_slot_count;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
```

```
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      0 |         0 |        2
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         1 |        2
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         1 |       14

query | queue | slot_count | start_time          | state     | queue_time | exec_time
------+-------+------------+---------------------+-----------+------------
+-----------
 2260 |     2 |          1 | 2014-09-25 00:12:11 | Executing | 0          | 4042618
 2262 |     3 |          1 | 2014-09-25 00:12:15 | Executing | 0          | 680
```

Langkah 2: Jalankan kueri dari sesi yang berbeda

Selanjutnya, jalankan kueri dari sesi yang berbeda.

Untuk menjalankan kueri dari sesi yang berbeda

1. Di jendela RSQL 1 dan 2, jalankan yang berikut ini untuk menggunakan grup kueri uji.

```
set query_group to test;
```

2. Di jendela RSQL 1, jalankan kueri yang berjalan lama berikut.

```
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

3. Karena kueri yang berjalan lama masih berjalan di jendela RSQL 1, jalankan yang berikut ini. Perintah ini meningkatkan jumlah slot untuk menggunakan semua slot untuk antrian dan kemudian mulai menjalankan kueri yang berjalan lama.

```
set wlm_query_slot_count to 2;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

4. Buka jendela RSQL ketiga dan kueri tampilan untuk melihat hasilnya.

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

Berikut ini adalah contoh hasil.

```
query | description                                | slots | mem | max_time | user_* |
 query_* | queued | executing | executed
------+--------------------------------------------+-------+-----+----------+--------
+---------+--------+-----------+----------
    0 | (super user) and (query group: superuser) |     1 | 357 |        0 |  false |
 false   |      0 |         0 |        0
    1 | (query group: test)                       |     2 | 627 |        0 |  false |
 false   |      1 |         1 |        2
    2 | (suser group: admin)                      |     3 | 557 |        0 |  false |
 false   |      0 |         0 |        3
    3 | (querytype:any)                           |     5 | 250 |        0 |  false |
 false   |      0 |         1 |       18

query | queue | slot_count | start_time          | state         | queue_time |
 exec_time
------+-------+------------+---------------------+---------------+------------
+-----------
 2286 |     1 |          2 | 2014-09-25 00:16:48 | QueuedWaiting | 3758950    | 0
 2282 |     1 |          1 | 2014-09-25 00:16:33 | Executing     | 0          |
 19335850
 2288 |     3 |          1 | 2014-09-25 00:16:52 | Executing     | 0          | 666
```

Perhatikan bahwa kueri pertama menggunakan salah satu slot yang dialokasikan ke antrian 1 untuk menjalankan kueri. Selain itu, perhatikan bahwa ada satu kueri yang menunggu dalam antrian (di mana `queued` ada `1` dan `state` berada`QueuedWaiting`). Setelah kueri pertama selesai, yang kedua mulai berjalan. Eksekusi ini terjadi karena kedua kueri dialihkan ke grup `test` kueri, dan kueri kedua harus menunggu slot yang cukup untuk mulai diproses.

Bagian 5: Membersihkan sumber daya Anda

Cluster Anda terus menambah biaya selama masih berjalan. Setelah menyelesaikan tutorial ini, kembalikan lingkungan Anda ke keadaan sebelumnya dengan mengikuti langkah-langkah di Temukan Sumber Daya Tambahan dan Setel Ulang Lingkungan Anda di Panduan Memulai Amazon Redshift.

Untuk informasi lebih lanjut tentang WLM, lihatManajemen beban kerja.

Penskalaan konkurensi

Dengan fitur Penskalaan Konkurensi, Anda dapat mendukung ribuan pengguna bersamaan dan kueri bersamaan, dengan kinerja kueri cepat secara konsisten. Saat Anda mengaktifkan penskalaan konkurensi, Amazon Redshift secara otomatis menambahkan kapasitas cluster tambahan untuk memproses peningkatan kueri baca dan tulis. Pengguna melihat data terbaru, apakah kueri berjalan di cluster utama atau cluster penskalaan bersamaan.

Anda dapat mengelola kueri mana yang dikirim ke cluster penskalaan bersamaan dengan mengonfigurasi antrian WLM. Saat Anda mengaktifkan penskalaan konkurensi, kueri yang memenuhi syarat dikirim ke cluster penskalaan bersamaan alih-alih menunggu dalam antrian.

Anda dikenakan biaya untuk cluster penskalaan bersamaan hanya selama mereka menjalankan kueri secara aktif. Untuk informasi selengkapnya tentang penetapan harga, termasuk bagaimana biaya bertambah dan biaya minimum, lihat Harga Penskal aan Konkurensi.

Topik

- Kemampuan penskalaan konkurensi
- Batasan untuk penskalaan konkurensi
- Wilayah AWS untuk penskalaan konkurensi
- Kandidat penskalaan konkurensi
- Mengkonfigurasi antrian penskalaan konkurensi
- Memantau penskalaan konkurensi
- Tampilan sistem penskalaan konkurensi

Kemampuan penskalaan konkurensi

Saat Anda mengaktifkan penskalaan konkurensi untuk antrian WLM, ini berfungsi untuk operasi baca, seperti kueri dasbor. Ini juga berfungsi untuk operasi tulis yang umum digunakan, seperti pernyataan untuk konsumsi dan pemrosesan data.

Kemampuan penskalaan konkurensi untuk operasi penulisan

Penskalaan konkurensi mendukung operasi tulis yang sering digunakan, seperti pernyataan ekstrak, transformasi, dan muat (ETL). Penskalaan konkurensi untuk operasi penulisan sangat berguna ketika Anda ingin mempertahankan waktu respons yang konsisten ketika cluster Anda menerima sejumlah

besar permintaan. Ini meningkatkan throughput untuk operasi penulisan yang bersaing untuk sumber daya pada cluster utama.

Penskalaan konkurensi mendukung pernyataan COPY, INSERT, DELETE, UPDATE, CREATE TABLE AS (CTAS), dan VACUUM. Selain itu, penskalaan konkurensi mendukung penyegaran manual tampilan terwujud (MV) dan operasi vakum otomatis. Pernyataan bahasa manipulasi data (DML) lainnya dan pernyataan bahasa definisi data (DDL) tidak didukung. Ketika pernyataan tulis yang tidak didukung, seperti CREATE tanpa TABLE AS, disertakan dalam transaksi eksplisit sebelum pernyataan tulis yang didukung, tidak ada pernyataan tulis yang akan berjalan pada cluster penskalaan konkurensi.

Ketika Anda memperoleh kredit untuk penskalaan konkurensi, akrual kredit ini berlaku untuk operasi baca dan tulis.

Batasan untuk penskalaan konkurensi

Berikut ini adalah batasan untuk menggunakan penskalaan konkurensi Amazon Redshift:

- Itu tidak mendukung kueri pada tabel yang menggunakan kunci pengurutan yang disisipkan.
- Itu tidak mendukung kueri pada tabel sementara.
- Ini tidak mendukung kueri yang mengakses sumber daya eksternal yang dilindungi oleh jaringan terbatas atau konfigurasi cloud pribadi virtual (VPC).
- Itu tidak mendukung kueri yang berisi fungsi yang ditentukan pengguna Python (UDF) dan UDF Lambda.
- Ini tidak mendukung kueri yang mengakses tabel sistem, tabel katalog PostgreSQL, atau tabel tanpa cadangan.
- Ini tidak mendukung kueri COPY atau UNLOAD yang mengakses sumber daya eksternal saat izin kebijakan IAM terbatas diterapkan. Ini termasuk izin yang diterapkan baik ke sumber daya, seperti bucket Amazon S3 atau tabel DynamoDB, atau ke sumber. Sumber IAM dapat mencakup yang berikut:
  - `aws:sourceVpc`— Sumber VPC.
  - `aws:sourceVpce`— Titik akhir VPC sumber.
  - `aws:sourceIp`— Alamat IP sumber.

  Dalam beberapa kasus, Anda mungkin perlu menghapus izin yang membatasi sumber daya atau sumber, sehingga kueri COPY dan UNLOAD yang mengakses sumber daya dikirim ke cluster penskalaan bersamaan.

Untuk informasi selengkapnya tentang kebijakan sumber daya, lihat Jenis kebijakan di panduan AWS Identity and Access Management pengguna dan Mengon trol akses dari titik akhir VPC dengan kebijakan bucket.

- Penskalaan konkurensi Amazon Redshift untuk operasi penulisan tidak didukung untuk sebagian besar operasi DDL, seperti CREATE TABLE.
- Itu tidak mendukung ANALYZE untuk perintah COPY.
- Itu tidak mendukung operasi tulis pada tabel target di mana DISTSTYLE diatur ke ALL.
- Itu tidak mendukung operasi tulis pada tabel dengan kolom identitas.
- Amazon Redshift mendukung penskalaan konkurensi untuk operasi penulisan hanya pada node Amazon Redshift RG dan RA3. Penskalaan konkurensi untuk operasi penulisan tidak didukung pada jenis node lain.

Wilayah AWS untuk penskalaan konkurensi

Dengan Amazon Redshift, Anda dapat menggunakan penskalaan konkurensi untuk mengelola tuntutan beban kerja bersamaan di seluruh cluster Redshift. Topik ini merinci Wilayah mana Anda dapat menggunakan penskalaan konkurensi dengan Amazon Redshift.

Penskalaan konkurensi tersedia di Wil AWS ayah berikut:

- Wilayah AS Timur (Virginia Utara) (us-timur-1)
- Wilayah AS Timur (Ohio) (us-east-2)
- Wilayah AS Barat (California Utara) (us-barat-1)
- Wilayah AS Barat (Oregon) (us-barat-2)
- Wilayah Afrika (Cape Town) (af-south-1)
- Wilayah Asia Pasifik (Mumbai) (ap-south-1)
- Wilayah Asia Pasifik (Hyderabad) (ap-south-2)
- Wilayah Asia Pasifik (Seoul) (ap-northeast-2)
- Wilayah Asia Pasifik (Osaka) (ap-northeast-3)
- Wilayah Asia Pasifik (Singapura) (ap-southeast-1)
- Wilayah Asia Pasifik (Sydney) (ap-southeast-2)
- Wilayah Asia Pasifik (Jakarta) (ap-southeast-3)

- Wilayah Asia Pasifik (Malaysia) (ap-southeast-5)
- Wilayah Asia Pasifik (Selandia Baru) (ap-southeast-6)
- Wilayah Asia Pasifik (Thailand) (ap-southeast-7)
- Wilayah Asia Pasifik (Hong Kong) (ap-east-1)
- Wilayah Asia Pasifik (Taipei) (ap-east-2)
- Wilayah Asia Pasifik (Tokyo) (ap-northeast-1)
- Wilayah Kanada (Tengah) (ca-central-1)
- Wilayah Kanada Barat (Calgary) (ca-west-1)
- Wilayah China (Beijing) (cn-north-1)
- Wilayah China (Ningxia) (cn-northwest-1)
- Wilayah Eropa (Frankfurt) (eu-central-1)
- Wilayah Eropa (Irlandia) (eu-west-1)
- Wilayah Eropa (London) (eu-west-2)
- Wilayah Eropa (Paris) (eu-west-3)
- Wilayah Eropa (Stockholm) (eu-utara-1)
- Wilayah Eropa (Zurich) (eu-central-2)
- Wilayah Eropa (Milan) (eu-south-1)
- Eropa (Spanyol) Wilayah (eu-south-2)
- Wilayah Israel (Tel Aviv) (il-sentral-1)
- Wilayah Timur Tengah (Bahrain) (me-south-1)
- Wilayah Meksiko (Tengah) (mx-central-1)
- Wilayah Amerika Selatan (São Paulo) (sa-east-1)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Kandidat penskalaan konkurensi

Dengan Amazon Redshift, Anda dapat menskalakan pemrosesan kueri untuk mempercepat eksekusi kueri bersamaan. Topik berikut menjelaskan kriteria yang digunakan Amazon Redshift untuk menentukan kueri apa yang akan diarahkan ke penskalaan konkurensi.

Kueri dialihkan ke cluster penskalaan konkurensi hanya jika cluster utama memenuhi persyaratan berikut:

- EC2-VPC platform.
- Jenis simpul harus dc2.8xlarge, dc2.large, rg.large, rg.xlarge, rg.4xlarge, rg.12xlarge, ra3.large, ra3.xlplus, ra3.4xlarge, atau ra3.16xlarge. Penskalaan konkurensi untuk operasi penulisan hanya didukung pada node Amazon Redshift RG dan RA3.
- Maksimal 32 node komputasi untuk cluster dengan tipe node rg.large, rg.xlarge, rg.4xlarge, rg.12xlarge, ra3.xlplus, ra3.4xlarge, atau ra3.16xlarge. Selain itu, jumlah node cluster utama tidak boleh lebih besar dari 32 node ketika cluster awalnya dibuat. Misalnya, bahkan jika cluster saat ini memiliki 20 node, tetapi awalnya dibuat dengan 40, itu tidak memenuhi persyaratan untuk penskalaan konkurensi. Sebaliknya, jika cluster DC2 saat ini memiliki 40 node, tetapi awalnya dibuat dengan 20, itu memenuhi persyaratan untuk penskalaan konkurensi.
- Bukan cluster simpul tunggal.

Mengkonfigurasi antrian penskalaan konkurensi

Dengan Amazon Redshift, Anda dapat mengelola konkurensi dan sumber daya sistem dengan mengonfigurasi penskalaan konkurensi. Dengan antrian penskalaan konkurensi, Anda dapat menetapkan batas jumlah kueri atau sesi pengguna yang dapat dijalankan secara bersamaan. Bagian berikut memberikan petunjuk tentang cara mengaktifkan antrian penskalaan konkurensi di Amazon Redshift, sehingga Anda dapat menangani kueri bersamaan dan sesi pengguna secara efektif.

Anda merutekan kueri ke cluster penskalaan konkurensi dengan mengaktifkan penskalaan konkurensi dalam antrian pengelola beban kerja (WLM). Untuk mengaktifkan penskalaan konkurensi untuk antrian, atur nilai mode Penskalaan Concurrency ke auto.

Ketika jumlah kueri yang dialihkan ke antrian dengan penskalaan konkurensi diaktifkan melebihi kapasitas konkurensi antrian, apakah kapasitas dikonfigurasi secara manual atau ditentukan secara otomatis, kueri yang memenuhi syarat dikirim ke cluster penskalaan konkurensi. Ketika slot antrian tersedia di cluster utama, kueri dialihkan ke dan dijalankan di cluster utama. Seperti halnya antrian WLM lainnya, Anda merutekan kueri ke antrian penskalaan konkurensi berdasarkan grup pengguna, atau dengan memberi label kueri dengan label grup kueri, atau sesuai dengan kondisi pencocokan yang ditentukan dalam Menetapkan kueri ke antrian. Anda juga dapat merutekan kueri dengan mendefinisikanAturan pemantauan kueri WLM. Misalnya, Anda dapat merutekan semua kueri yang membutuhkan waktu lebih dari 5 detik ke antrian penskalaan konkurensi. Perlu diingat bahwa

perilaku antrian dapat bervariasi, tergantung pada apakah Anda menggunakan WLM otomatis atau WLM manual. Untuk informasi selengkapnya, lihat Mener apkan WLM otomatis atau Menerapkan W LM manual.

Jumlah default cluster penskalaan konkurensi adalah satu. Jumlah cluster penskalaan konkurensi yang dapat digunakan dikendalikan olehklaster penskalaan maksimum.

Memantau penskalaan konkurensi

Dengan Amazon Redshift, Anda dapat memantau dan mengelola penskalaan konkurensi untuk mengoptimalkan kinerja dan efisiensi biaya untuk beban kerja pergudangan data Anda. Penskalaan konkurensi memungkinkan Amazon Redshift menambahkan kapasitas cluster tambahan secara otomatis saat permintaan beban kerja meningkat, dan menghapus kapasitas tersebut saat permintaan menurun. Bagian berikut memberikan panduan untuk memantau penskalaan konkurensi untuk cluster Amazon Redshift Anda.

Anda dapat melihat apakah kueri sedang berjalan di cluster utama atau cluster penskalaan konkurensi dengan menavigasi ke Cluster di konsol Amazon Redshift dan memilih cluster. Kemudian pilih tab Pem antauan kueri dan Konkur ensi beban kerja untuk melihat informasi tentang menjalankan kueri dan kueri antri.

Untuk menemukan waktu eksekusi, kueri tabel STL_QUERY dan filter pada kolom. `concurrency_scaling_status` Kueri berikut membandingkan waktu antrian dan waktu eksekusi untuk kueri yang dijalankan pada cluster penskalaan konkurensi dan kueri yang dijalankan pada cluster utama.

```
SELECT w.service_class AS queue
, CASE WHEN q.concurrency_scaling_status = 1 THEN 'concurrency scaling cluster' ELSE
 'main cluster' END as concurrency_scaling_status
, COUNT( * ) AS queries
, SUM( q.aborted ) AS aborted
, SUM( ROUND( total_queue_time::NUMERIC / 1000000,2) ) AS queue_secs
, SUM( ROUND( total_exec_time::NUMERIC / 1000000,2) ) AS exec_secs
FROM stl_query q
JOIN stl_wlm_query w
USING (userid,query)
WHERE q.userid > 1
AND q.starttime > '2019-01-04 16:38:00'
AND q.endtime < '2019-01-04 17:40:00'
GROUP BY 1,2
ORDER BY 1,2;
```

Sesuaikan `starttime` dan `endtime` nilai sesuai dengan kebutuhan Anda.

Tampilan sistem penskalaan konkurensi

Dengan Amazon Redshift, Anda dapat menggunakan tampilan sistem penskalaan Concurrency untuk memantau dan mengelola aktivitas penskalaan konkurensi di cluster Anda. Bagian berikut menjelaskan menanyakan tampilan sistem ini dan menafsirkan hasilnya untuk secara efektif memanfaatkan penskalaan konkurensi di lingkungan Amazon Redshift Anda.

Satu set tampilan sistem dengan awalan SVCS memberikan rincian dari tabel log sistem tentang kueri pada cluster penskalaan utama dan konkurensi.

Tampilan berikut memiliki informasi serupa dengan tampilan STL atau tampilan SVL yang sesuai:

- SVCS_ALERT_EVENT_LOG
- SVCS_KOMPILASI
- SVCS_JELASKAN
- SVCS_PLAN_INFO
- SVCS_QUERY_RINGKASAN
- SVCS_STREAM_SEGS

Tampilan berikut khusus untuk penskalaan konkurensi.

- SVCS_CONCURRENCY_SCALING_PEMAKAIAN

Untuk informasi selengkapnya tentang penskalaan konkurensi, lihat topik berikut di Panduan Manajemen Amazon Redshift.

- Melihat Data Penskalaan Konkurensi
- Melihat Kinerja Cluster Selama Eksekusi Kueri
- Melihat Detail Kueri

Akselerasi kueri singkat

Akselerasi kueri pendek (SQA) memprioritaskan kueri berjalan pendek yang dipilih di depan kueri yang berjalan lebih lama. SQA menjalankan kueri yang berjalan singkat di ruang khusus, sehingga kueri SQA tidak dipaksa menunggu dalam antrian di belakang kueri yang lebih lama. SQA hanya

memprioritaskan kueri yang berjalan singkat dan berada dalam antrian yang ditentukan pengguna. Dengan SQA, kueri yang berjalan pendek mulai berjalan lebih cepat dan pengguna melihat hasilnya lebih cepat.

Jika Anda mengaktifkan SQA, Anda dapat mengurangi antrian manajemen beban kerja (WLM) yang didedikasikan untuk menjalankan kueri pendek. Selain itu, kueri yang berjalan lama tidak perlu bersaing dengan kueri pendek untuk slot dalam antrian, sehingga Anda dapat mengonfigurasi antrian WLM Anda untuk menggunakan slot kueri yang lebih sedikit. Bila Anda menggunakan konkurensi yang lebih rendah, throughput kueri meningkat dan kinerja sistem secara keseluruhan ditingkatkan untuk sebagian besar beban kerja.

CREATE TABLE ASPernyataan (CTAS) dan kueri baca-saja, seperti SELECT pernyataan, memenuhi syarat untuk SQA.

Amazon Redshift menggunakan algoritma pembelajaran mesin untuk menganalisis setiap kueri yang memenuhi syarat dan memprediksi waktu eksekusi kueri. Secara default, WLM secara dinamis menetapkan nilai untuk runtime maksimum SQA berdasarkan analisis beban kerja cluster Anda. Atau, Anda dapat menentukan nilai tetap 1—20 detik. Jika waktu berjalan yang diprediksi kueri kurang dari runtime maksimum SQA yang ditentukan atau ditetapkan secara dinamis dan kueri sedang menunggu dalam antrian WLM, SQA memisahkan kueri dari antrian WLM dan menjadwalkannya untuk eksekusi prioritas. Jika kueri berjalan lebih lama dari runtime maksimum SQA, WLM memindahkan kueri ke antrian WLM yang cocok pertama berdasarkan aturan penetapan antrian WLM. Seiring waktu, prediksi meningkat saat SQA belajar dari pola kueri Anda.

SQA diaktifkan secara default di grup parameter default dan untuk semua grup parameter baru. Untuk menonaktifkan SQA di konsol Amazon Redshift, edit konfigurasi WLM untuk grup parameter dan hapus pilihan Aktifkan akselerasi kueri pendek. Sebagai praktik terbaik, sebaiknya gunakan jumlah slot kueri WLM 15 atau kurang untuk mempertahankan kinerja sistem secara keseluruhan yang optimal. Untuk informasi tentang memodifikasi konfigurasi WLM, lihat Mengon figurasi Manajemen Beb an Kerja di Panduan Manajemen Amazon Redshift.

Waktu proses maksimum untuk kueri pendek

Saat Anda mengaktifkan SQA, WLM menetapkan runtime maksimum untuk kueri pendek menjadi dinamis secara default. Sebaiknya simpan pengaturan dinamis untuk runtime maksimum SQA. Anda dapat mengganti pengaturan default dengan menentukan nilai tetap 1—20 detik.

Dalam beberapa kasus, Anda dapat mempertimbangkan untuk menggunakan nilai yang berbeda untuk nilai runtime maksimum SQA untuk meningkatkan kinerja sistem Anda. Dalam kasus seperti

itu, analisis beban kerja Anda untuk menemukan waktu eksekusi maksimum untuk sebagian besar kueri jangka pendek Anda. Kueri berikut mengembalikan runtime maksimum untuk kueri pada sekitar persentil ke-70.

```
select least(greatest(percentile_cont(0.7)
within group (order by total_exec_time / 1000000) + 2, 2), 20)
from stl_wlm_query
where userid >= 100
and final_state = 'Completed';
```

Setelah mengidentifikasi nilai runtime maksimum yang berfungsi dengan baik untuk beban kerja Anda, Anda tidak perlu mengubahnya kecuali beban kerja Anda berubah secara signifikan.

Pemantauan SQA

Untuk memeriksa apakah SQA diaktifkan, jalankan query berikut. Jika kueri mengembalikan baris, maka SQA diaktifkan.

```
select * from stv_wlm_service_class_config
where service_class = 14;
```

Query berikut menunjukkan jumlah query yang melewati setiap antrian kueri (kelas layanan). Ini juga menunjukkan waktu eksekusi rata-rata, jumlah kueri dengan waktu tunggu pada persentil ke-90, dan waktu tunggu rata-rata. Kueri SQA digunakan di kelas layanan 14.

```
select final_state, service_class, count(*), avg(total_exec_time),
percentile_cont(0.9) within group (order by total_queue_time), avg(total_queue_time)
from stl_wlm_query where userid >= 100 group by 1,2 order by 2,1;
```

Untuk menemukan kueri mana yang diambil oleh SQA dan berhasil diselesaikan, jalankan kueri berikut.

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class = 14 and a.final_state = 'Completed'
order by b.query desc limit 5;
```

Untuk menemukan kueri yang diambil SQA tetapi waktunya habis, jalankan kueri berikut.

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class = 14 and a.final_state = 'Evicted'
order by b.query desc limit 5;
```

Untuk informasi selengkapnya tentang kueri yang diusir dan, secara lebih umum, tindakan berbasis aturan yang dapat diambil pada kueri, lihat. Aturan pemantauan kueri WLM

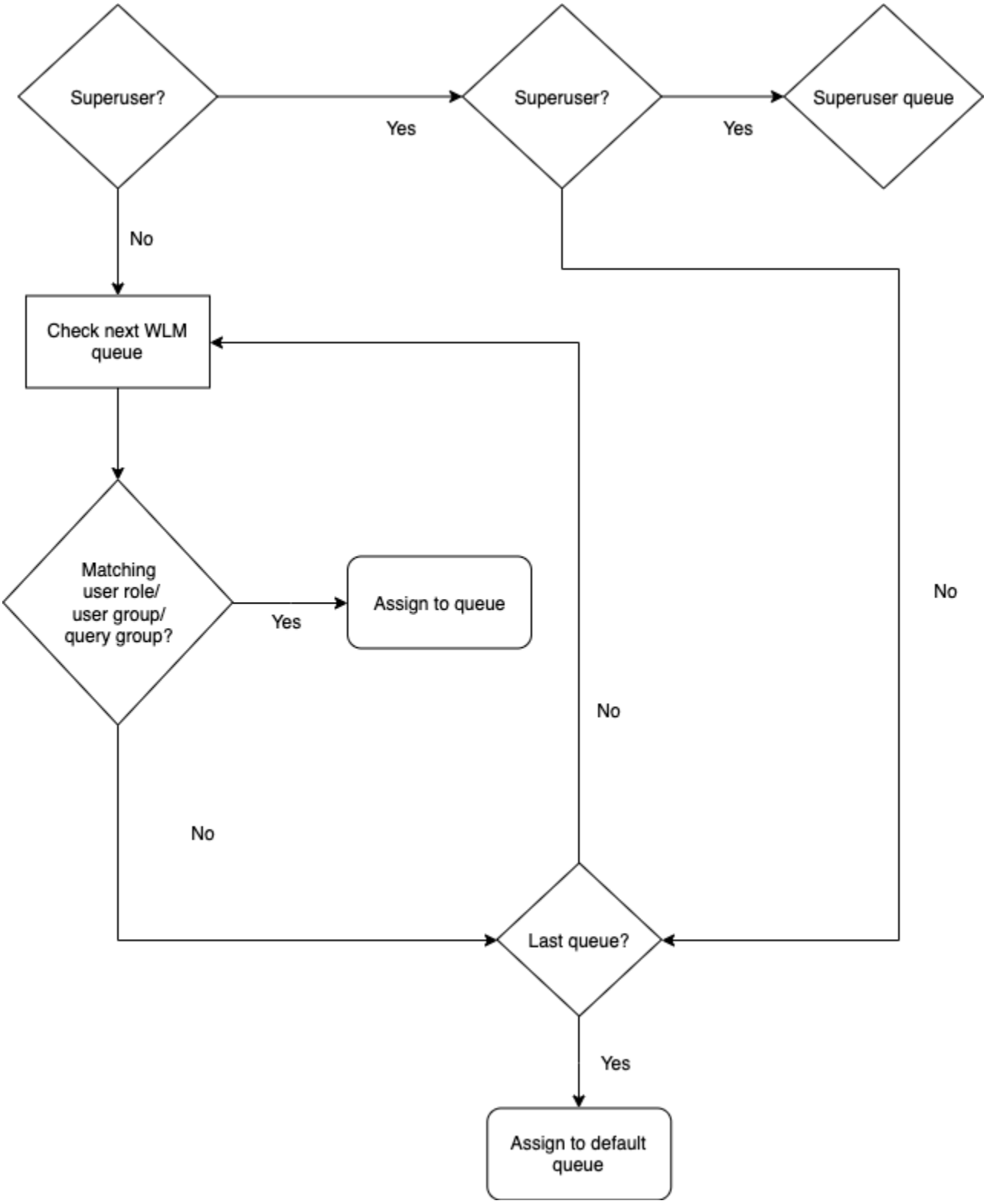
Aturan penetapan antrian WLM

Dengan Amazon Redshift, Anda dapat mengontrol alokasi memori dan sumber daya CPU ke kueri pengguna dengan menentukan aturan penetapan antrian dalam konfigurasi manajemen beban kerja (WLM). Bagian berikut menjelaskan membuat dan mengelola aturan penetapan antrian WLM untuk mencapai alokasi sumber daya yang efisien dan memenuhi perjanjian tingkat layanan untuk beragam beban kerja di Amazon Redshift.

Saat pengguna menjalankan kueri, WLM menetapkan kueri ke antrian pencocokan pertama, berdasarkan aturan penetapan antrian WLM:

1. Jika pengguna masuk sebagai superuser dan menjalankan kueri dalam grup kueri berlabel superuser, kueri ditetapkan ke antrian superuser.
2. Jika pengguna adalah bagian dari peran, milik grup pengguna terdaftar, atau menjalankan kueri dalam grup kueri terdaftar, kueri ditetapkan ke antrian pencocokan pertama.
3. Jika kueri tidak memenuhi kriteria apa pun, kueri ditetapkan ke antrian default, yang merupakan antrian terakhir yang ditentukan dalam konfigurasi WLM.

Diagram berikut menggambarkan bagaimana aturan ini bekerja.

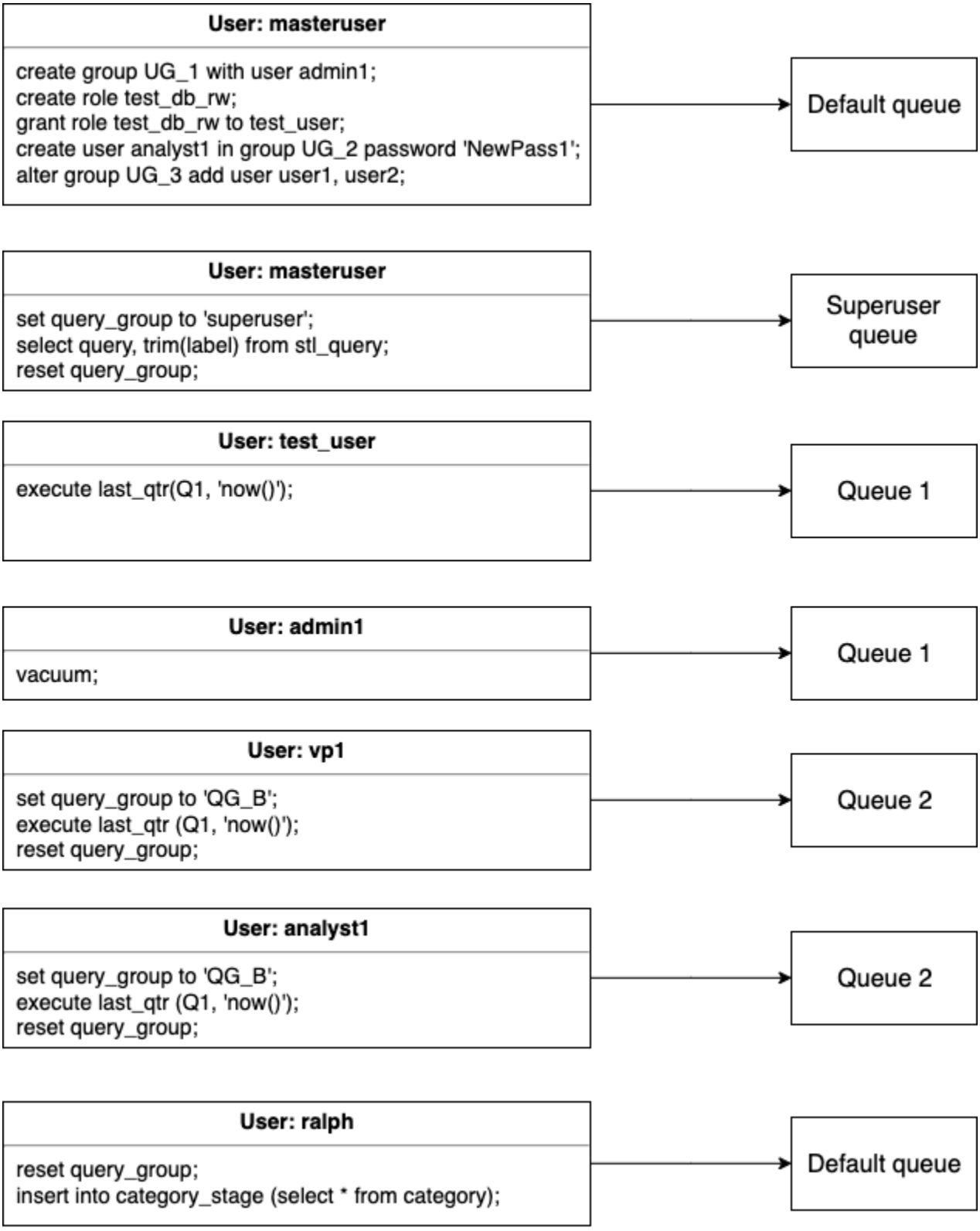

Superuser?
Yes
Superuser?
Yes
Superuser queue
No
Check next WLM queue
Matching user role/ user group/ query group?
Yes
Assign to queue
No
No
No
Last queue?
Yes
Assign to default queue

Contoh tugas antrian

Tabel berikut menunjukkan konfigurasi WLM dengan antrian superuser dan empat antrian yang ditentukan pengguna.

Antrean	Bersamaan	Peran Pengguna	Grup Pengguna	Grup Kueri
Superuser	1			pengguna super
1	5	test_db_rw	UG_1	
2	5			QG_B
3	5		UG_2	QG_C
Default	5			

Ilustrasi berikut menunjukkan bagaimana kueri ditetapkan ke antrian di tabel sebelumnya sesuai dengan grup pengguna dan grup kueri. Untuk informasi tentang cara menetapkan kueri ke grup pengguna dan grup kueri saat runtime, lihat Menetapkan kueri ke antrian nanti di bagian ini.



Dalam contoh ini, WLM membuat tugas berikut:

1. Kumpulan pernyataan pertama menunjukkan tiga cara untuk menetapkan pengguna ke grup pengguna. Pernyataan dijalankan oleh penggunaadminuser, yang bukan anggota grup pengguna yang terdaftar dalam antrian WLM apa pun. Tidak ada grup kueri yang disetel, sehingga pernyataan dialihkan ke antrian default.
2. Pengguna adminuser adalah superuser dan grup kueri diatur ke'superuser', sehingga kueri ditetapkan ke antrian superuser.
3. Pengguna diberi peran test_user yang test_db_rw tercantum dalam antrian 1, sehingga kueri ditugaskan ke antrian 1.
4. Pengguna admin1 adalah anggota grup pengguna yang tercantum dalam antrian 1, sehingga kueri ditugaskan ke antrian 1.
5. Pengguna vp1 bukan anggota grup pengguna yang terdaftar. Grup kueri disetel ke'QG_B', sehingga kueri ditugaskan ke antrian 2.
6. Pengguna analyst1 adalah anggota grup pengguna yang tercantum dalam antrian 3, tetapi 'QG_B' cocok dengan antrian 2, sehingga kueri ditugaskan ke antrian 2.
7. Pengguna bukan ralph anggota grup pengguna yang terdaftar dan grup kueri telah diatur ulang, jadi tidak ada antrian yang cocok. Kueri ditetapkan ke antrian default.

Menetapkan kueri ke antrian

Dengan Amazon Redshift, Anda dapat mengelola konkurensi beban kerja dan memprioritaskan kueri dengan menugaskannya ke antrian. Antrian memungkinkan Anda mengalokasikan sumber daya seperti memori dan CPU untuk berbagai jenis kueri atau pengguna, memastikan kueri penting diprioritaskan daripada yang kurang penting. Bagian berikut menjelaskan cara membuat antrian, mengonfigurasi propertinya, dan menetapkan kueri masuk berdasarkan kriteria yang Anda tentukan.

Contoh berikut menetapkan kueri ke antrian sesuai dengan peran pengguna, grup pengguna, dan grup kueri.

Menetapkan kueri ke antrian berdasarkan peran pengguna

Jika pengguna ditugaskan ke peran dan peran tersebut dilampirkan ke antrian, maka kueri yang dijalankan oleh pengguna tersebut ditetapkan ke antrian tersebut. Contoh berikut membuat peran pengguna bernama sales_rw dan menetapkan pengguna test_user ke peran itu.

```
create role sales_rw;
grant role sales_rw to test_user;
```

Anda juga dapat menggabungkan izin dari dua peran dengan secara eksplisit memberikan satu peran ke peran lain. Menetapkan peran bersarang ke pengguna memberikan izin kedua peran kepada pengguna.

```
create role sales_rw;
create role sales_ro;
grant role sales_ro to role sales_rw;
grant role sales_rw to test_user;
```

Untuk melihat daftar pengguna yang telah diberikan peran dalam cluster, kueri tabel SVV_USER_GRANTS. Untuk melihat daftar peran yang telah diberikan peran dalam cluster, kueri tabel SVV_ROLE_GRANTS.

```
select * from svv_user_grants;
select * from svv_role_grants;
```

Menetapkan kueri ke antrian berdasarkan grup pengguna

Jika nama grup pengguna terdaftar dalam definisi antrian, kueri yang dijalankan oleh anggota grup pengguna tersebut ditetapkan ke antrian yang sesuai. Contoh berikut membuat grup pengguna dan menambahkan pengguna ke grup dengan menggunakan perintah SQLBUAT PENGGUNA,BUAT GRUP, danMENGUBAH KELOMPOK.

```
create group admin_group with user admin246, admin135, sec555;
create user vp1234 in group ad_hoc_group password 'vpPass1234';
alter group admin_group add user analyst44, analyst45, analyst46;
```

Menetapkan kueri ke grup kueri

Anda dapat menetapkan kueri ke antrian saat runtime dengan menetapkan kueri Anda ke grup kueri yang sesuai. Gunakan perintah SET untuk memulai grup kueri.

```
SET query_group TO group_label
```

Di sini, *group_label* adalah label grup kueri yang terdaftar dalam konfigurasi WLM.

Semua kueri yang Anda jalankan setelah SET query_group perintah dijalankan sebagai anggota grup kueri yang ditentukan sampai Anda mengatur ulang grup kueri atau mengakhiri sesi login Anda

saat ini. Untuk informasi tentang menyetel dan mengatur ulang objek Amazon Redshift, lihat SET dan MENGATUR ULANG di Referensi Perintah SQL.

Label grup kueri yang Anda tentukan harus disertakan dalam konfigurasi WLM saat ini; jika tidak, perintah SET query_group tidak berpengaruh pada antrian kueri.

Label yang ditentukan dalam klausa TO ditangkap dalam log kueri sehingga Anda dapat menggunakan label untuk pemecahan masalah. Untuk informasi tentang parameter konfigurasi query_group, lihat Query_group di Referensi Konfigurasi.

Contoh berikut menjalankan dua kueri sebagai bagian dari grup kueri 'prioritas' dan kemudian mengatur ulang grup kueri.

```
set query_group to 'priority';
select count(*)from stv_blocklist;
select query, elapsed, substring from svl_qlog order by query desc limit 5;
reset query_group;
```

Menetapkan kueri ke antrian superuser

Untuk menetapkan kueri ke antrian superuser, masuk ke Amazon Redshift sebagai pengguna super dan kemudian jalankan kueri di grup superuser. Setelah selesai, atur ulang grup kueri sehingga kueri berikutnya tidak berjalan dalam antrian superuser.

Contoh berikut menetapkan dua perintah untuk dijalankan dalam antrian superuser.

```
set query_group to 'superuser';

analyze;
vacuum;
reset query_group;
```

Untuk melihat daftar pengguna super, kueri tabel katalog sistem PG_USER.

```
select * from pg_user where usesuper = 'true';
```

Properti konfigurasi dinamis dan statis WLM

Properti konfigurasi WLM bersifat dinamis atau statis. Anda dapat menerapkan properti dinamis ke database tanpa reboot cluster, tetapi properti statis memerlukan reboot cluster agar perubahan

diterapkan. Namun, jika Anda mengubah properti dinamis dan statis pada saat yang sama, maka Anda harus me-reboot cluster agar semua perubahan properti berlaku. Ini benar apakah properti yang diubah dinamis atau statis.

Saat properti dinamis sedang diterapkan, status cluster Anda adalah`modifying`. Beralih antara WLM otomatis dan WLM manual adalah perubahan statis dan memerlukan reboot cluster untuk berlaku.

Tabel berikut menunjukkan properti WLM mana yang dinamis atau statis saat menggunakan WLM otomatis atau WLM manual.

Properti WLM	WLM otomatis	WLM manual
Grup kueri	Dinamis	Statis
Karakter pengganti grup kueri	Dinamis	Statis
Grup pengguna	Dinamis	Statis
Wildcard grup pengguna	Dinamis	Statis
Peran pengguna	Dinamis	Statis
Karakter pengganti peran pengguna	Dinamis	Statis
Konkurensi pada main	Tidak berlaku	Dinamis
Mode penskalaan konkurensi	Dinamis	Dinamis
Aktifkan akselerasi kueri singkat	Tidak berlaku	Dinamis
Waktu proses maksimum untuk kueri pendek	Dinamis	Dinamis
Persentase memori yang digunakan	Tidak berlaku	Dinamis
Batas waktu	Tidak berlaku	Dinamis

Properti WLM	WLM otomatis	WLM manual
Prioritas	Dinamis	Tidak berlaku
Menambahkan atau menghapus antrian	Dinamis	Statis

Jika Anda menambahkan aturan pemantauan kueri (QMR), atau memodifikasi atau menghapus QMR yang ada, perubahan terjadi secara otomatis tanpa perlu memulai ulang cluster.

> **Note**
>
> Saat menggunakan WLM manual, jika nilai batas waktu diubah, nilai baru diterapkan ke kueri apa pun yang mulai berjalan setelah nilai diubah. Jika konkurensi atau persen memori yang digunakan diubah, Amazon Redshift berubah ke konfigurasi baru secara dinamis. Dengan demikian, kueri yang sedang berjalan tidak terpengaruh oleh perubahan. Untuk informasi selengkapnya, lihat Alokasi Memori Dinamis WLM.

Topik

- Alokasi memori dinamis WLM
- Contoh WLM dinamis

Alokasi memori dinamis WLM

Dalam setiap antrian, WLM membuat sejumlah slot kueri sama dengan tingkat konkurensi antrian. Jumlah memori yang dialokasikan ke slot query sama dengan persentase memori yang dialokasikan ke antrian dibagi dengan jumlah slot. Jika Anda mengubah alokasi memori atau konkurensi, Amazon Redshift secara dinamis mengelola transisi ke konfigurasi WLM baru. Dengan demikian, kueri aktif dapat berjalan hingga selesai menggunakan jumlah memori yang dialokasikan saat ini. Pada saat yang sama, Amazon Redshift memastikan bahwa total penggunaan memori tidak pernah melebihi 100 persen dari memori yang tersedia.

Manajer beban kerja menggunakan proses berikut untuk mengelola transisi:

1. WLM menghitung ulang alokasi memori untuk setiap slot query baru.

2. Jika slot kueri tidak aktif digunakan oleh kueri yang sedang berjalan, WLM menghapus slot, yang membuat memori itu tersedia untuk slot baru.
3. Jika slot kueri aktif digunakan, WLM menunggu kueri selesai.
4. Saat kueri aktif selesai, slot kosong dihapus dan memori terkait dibebaskan.
5. Ketika memori yang cukup tersedia untuk menambahkan satu atau lebih slot, slot baru ditambahkan.
6. Ketika semua kueri yang sedang berjalan pada saat perubahan selesai, jumlah slot sama dengan level konkurensi baru, dan transisi ke konfigurasi WLM baru selesai.

Akibatnya, kueri yang berjalan saat perubahan terjadi terus menggunakan alokasi memori asli. Kueri yang diantri saat perubahan terjadi dialihkan ke slot baru saat tersedia.

Jika properti dinamis WLM diubah selama proses transisi, WLM segera mulai beralih ke konfigurasi baru, mulai dari keadaan saat ini. Untuk melihat status transisi, kueri tabel STV_WLM_SERVICE_CLASS_CONFIG sistem.

Contoh WLM dinamis

Dengan Amazon Redshift, Anda dapat secara otomatis mengelola distribusi beban kerja dan alokasi sumber daya di seluruh cluster Amazon Redshift menggunakan Dynamic WLM (Manajemen Beban Kerja). Dynamic WLM adalah contoh konfigurasi Manajemen Beban Kerja (WLM) yang secara dinamis menyesuaikan alokasi memori berdasarkan permintaan beban kerja, memungkinkan konkurensi dan kinerja yang optimal. Bagian berikut memberikan detail tentang mengimplementasikan dan mengonfigurasi Dynamic WLM untuk cluster Amazon Redshift Anda.

Misalkan bahwa cluster WLM Anda dikonfigurasi dengan dua antrian, menggunakan properti dinamis berikut.

Antrean	Bersamaan	% Memori untuk Digunakan
1	4	50%
2	4	50%

Sekarang anggaplah cluster Anda memiliki memori 200 GB yang tersedia untuk pemrosesan kueri. (Nomor ini sewenang-wenang dan hanya digunakan untuk ilustrasi.) Seperti yang ditunjukkan persamaan berikut, setiap slot dialokasikan 25 GB.

```
(200 GB * 50% ) / 4 slots  = 25 GB
```

Selanjutnya, Anda mengubah WLM Anda untuk menggunakan properti dinamis berikut.

Antrean	Bersamaan	% Memori untuk Digunakan
1	3	75%
2	4	25%

Seperti yang ditunjukkan persamaan berikut, alokasi memori baru untuk setiap slot dalam antrian 1 adalah 50 GB.

```
(200 GB * 75% ) / 3 slots = 50 GB
```

Misalkan kueri A1, A2, A3, dan A4 sedang berjalan ketika konfigurasi baru diterapkan, dan kueri B1, B2, B3, dan B4 mengantri. WLM secara dinamis mengkonfigurasi ulang slot kueri sebagai berikut.

Langkah	Kueri Berjalan	Jumlah Slot Saat Ini	Jumlah Slot Target	Memori yang dialokasikan	Memori Tersedia
1	A1, A2, A3, A4	4	0	100GB	50 GB
2	A2, A3, A4	3	0	75 GB	75 GB
3	A3, A4	2	0	50 GB	100GB
4	A3, A4, B1	2	1	100GB	50 GB
5	A4, B1	1	1	75 GB	75 GB
6	A4, B1, B2	1	2	125GB	25 GB
7	B1, B2	0	2	100GB	50 GB
8	B1, B2, B3	0	3	150GB	0 GB

1. WLM menghitung ulang alokasi memori untuk setiap slot query. Awalnya, antrian 1 dialokasikan 100 GB. Antrian baru memiliki alokasi total 150 GB, sehingga antrian baru segera memiliki 50 GB yang tersedia. Antrian 1 sekarang menggunakan empat slot, dan level konkurensi baru adalah tiga slot, jadi tidak ada slot baru yang ditambahkan.
2. Ketika satu kueri selesai, slot dihapus dan 25 GB dibebaskan. Antrian 1 sekarang memiliki tiga slot dan 75 GB memori yang tersedia. Konfigurasi baru membutuhkan 50 GB untuk setiap slot baru, tetapi tingkat konkurensi baru adalah tiga slot, jadi tidak ada slot baru yang ditambahkan.
3. Ketika kueri kedua selesai, slot dihapus, dan 25 GB dibebaskan. Antrian 1 sekarang memiliki dua slot dan 100 GB memori bebas.
4. Slot baru ditambahkan menggunakan 50 GB memori bebas. Antrian 1 sekarang memiliki tiga slot, dan memori bebas 50 GB. Kueri antrian sekarang dapat dialihkan ke slot baru.
5. Ketika kueri ketiga selesai, slot dihapus, dan 25 GB dibebaskan. Antrian 1 sekarang memiliki dua slot, dan 75 GB memori bebas.
6. Slot baru ditambahkan menggunakan 50 GB memori bebas. Antrian 1 sekarang memiliki tiga slot, dan memori bebas 25 GB. Kueri antrian sekarang dapat dialihkan ke slot baru.
7. Ketika kueri keempat selesai, slot dihapus, dan 25 GB dibebaskan. Antrian 1 sekarang memiliki dua slot dan 50 GB memori bebas.
8. Slot baru ditambahkan menggunakan memori bebas 50 GB. Antrian 1 sekarang memiliki tiga slot dengan masing-masing 50 GB dan semua memori yang tersedia telah dialokasikan.

Transisi selesai dan semua slot kueri tersedia untuk kueri yang antri.

Aturan pemantauan kueri WLM

Di manajemen beban kerja Amazon Redshift (WLM), aturan pemantauan kueri menentukan batas kinerja berbasis metrik untuk antrian WLM dan menentukan tindakan apa yang harus diambil ketika kueri melampaui batas tersebut. Misalnya, untuk antrian yang didedikasikan untuk kueri berjalan singkat, Anda dapat membuat aturan yang membatalkan kueri yang berjalan selama lebih dari 60 detik. Untuk melacak kueri yang dirancang dengan buruk, Anda mungkin memiliki aturan lain yang mencatat kueri yang berisi loop bersarang.

Anda menentukan aturan pemantauan kueri sebagai bagian dari konfigurasi manajemen beban kerja (WLM) Anda. Anda dapat menentukan hingga 25 aturan untuk setiap antrian, dengan batas 25 aturan untuk semua antrian. Setiap aturan mencakup hingga tiga kondisi, atau predikat, dan satu tindakan. P redikat terdiri dari metrik, kondisi perbandingan (=, <, or >), dan nilai. Jika semua predikat untuk

aturan apa pun terpenuhi, tindakan aturan itu dipicu. Tindakan aturan yang mungkin adalah log, hop, dan abort, seperti yang dibahas berikut.

Aturan dalam antrian tertentu hanya berlaku untuk kueri yang berjalan dalam antrian itu. Aturan tidak tergantung pada aturan lain.

WLM mengevaluasi metrik setiap 10 detik. Amazon Redshift menerapkan aturan pemantauan kueri di tingkat kueri anak saat kueri ditulis ulang secara otomatis. Jika lebih dari satu aturan dipicu selama periode yang sama, WLM memilih aturan dengan tindakan paling parah. Jika tindakan untuk dua aturan memiliki tingkat keparahan yang sama, WLM menjalankan aturan dalam urutan abjad, berdasarkan nama aturan. Jika tindakan tersebut adalah hop atau abort, tindakan dicatat dan kueri dikeluarkan dari antrian. Jika tindakan adalah log, kueri terus berjalan dalam antrian. WLM memulai hanya satu tindakan log per kueri per aturan. Jika antrian berisi aturan lain, aturan tersebut tetap berlaku. Jika tindakan adalah hop dan kueri dialihkan ke antrian lain, aturan untuk antrian baru berlaku. Untuk informasi selengkapnya tentang pemantauan kueri dan tindakan pelacakan yang dilakukan pada kueri tertentu, lihat pengumpulan sampel diAkselerasi kueri singkat.

Ketika semua predikat aturan terpenuhi, WLM menulis baris ke tabel STL_WLM_RULE_ACTION sistem. Selain itu, Amazon Redshift mencatat metrik kueri untuk kueri yang sedang berjalan keSTV_QUERY_METRICS. Metrik untuk kueri yang diselesaikan disimpan diSTL_QUERY_METRICS.

> **Note**
>
> Untuk Amazon Redshift Serverless, Anda dapat mengonfigurasi antrian kueri dan aturan pemantauan menggunakan parameter. `wlm_json_configuration` Ini memungkinkan Anda untuk membuat beberapa antrian dengan peran pengguna yang berbeda, grup kueri, dan aturan pemantauan. Untuk informasi selengkapnya tentang mengonfigurasi antrian kueri tanpa server, lihat Menyet el antrian kueri di Panduan Manajemen Amazon Redshift.

Mendefinisikan aturan pemantauan kueri

Anda membuat aturan pemantauan kueri sebagai bagian dari konfigurasi WLM Anda, yang Anda tentukan sebagai bagian dari definisi grup parameter cluster Anda.

Anda dapat membuat aturan menggunakan Konsol Manajemen AWS atau secara terprogram menggunakan JSON.

> **Note**
>
> Jika Anda memilih untuk membuat aturan secara terprogram, kami sangat menyarankan menggunakan konsol untuk menghasilkan JSON yang Anda sertakan dalam definisi grup parameter. Untuk informasi selengkapnya, lihat Membuat aturan pemantauan kueri dan Meng konfigurasi Nilai AWS CLI Parameter Menggunakan Panduan Manajemen Amazon Redshift.

Untuk menentukan aturan pemantauan kueri, Anda menentukan elemen berikut:

- Nama aturan — Nama aturan harus unik dalam konfigurasi WLM. Nama aturan dapat terdiri dari hingga 32 karakter alfanumerik atau garis bawah, dan tidak dapat berisi spasi atau tanda kutip. Anda dapat memiliki hingga 25 aturan per antrian, dan batas total untuk semua antrian adalah 25 aturan.
- Satu atau lebih predikat — Anda dapat memiliki hingga tiga predikat per aturan. Jika semua predikat untuk aturan apa pun terpenuhi, tindakan terkait dipicu. Predikat didefinisikan oleh nama metrik, operator (=, <, or >), dan nilai. Contohnya adalah `query_cpu_time > 100000`. Untuk daftar metrik dan contoh nilai untuk metrik yang berbeda, lihat Metrik pemantauan kueri untuk Amazon Redshift disediakan berikut di bagian ini.
- Tindakan — Jika lebih dari satu aturan dipicu, WLM memilih aturan dengan tindakan paling parah. Tindakan yang mungkin, dalam urutan keparahan yang menaik, adalah:
  - Log — Merekam informasi tentang kueri dalam tabel sistem STL_WLM_RULE_ACTION. Gunakan tindakan Log jika Anda hanya ingin menulis catatan log. WLM membuat paling banyak satu log per kueri, per aturan. Setelah tindakan log, aturan lain tetap berlaku dan WLM terus memantau kueri.
  - Hop (hanya tersedia dengan WLM manual) — Catat tindakan dan lompat kueri ke antrian pencocokan berikutnya. Jika tidak ada antrian pencocokan lain, kueri dibatalkan. QMR hanya melompat pernyataan CREATE TABLE AS (CTAS) dan kueri read-only, seperti pernyataan SELECT. Untuk informasi selengkapnya, lihat Melompat antrian kueri WLM.
  - Batalkan — Catat tindakan dan batalkan kueri. QMR tidak menghentikan pernyataan COPY dan operasi pemeliharaan, seperti ALTER, ANALYZE, dan VACUUM.
  - Ubah prioritas (hanya tersedia dengan WLM otomatis) — Ubah prioritas kueri.

Untuk membatasi runtime kueri, sebaiknya buat aturan pemantauan kueri alih-alih menggunakan batas waktu WLM. Misalnya, Anda dapat mengatur `max_execution_time` ke 50.000 milidetik seperti yang ditunjukkan dalam cuplikan JSON berikut.

```
"max_execution_time": 50000
```

Namun sebaiknya Anda mendefinisikan aturan pemantauan kueri yang setara. Contoh berikut menunjukkan aturan pemantauan kueri yang ditetapkan `query_execution_time` ke 50 detik:

```
"rules":
[
    {
        "rule_name": "rule_query_execution",
        "predicate": [
            {
                "metric_name": "query_execution_time",
                "operator": ">",
                "value": 50
            }
        ],
        "action": "abort"
    }
]
```

Untuk langkah-langkah membuat atau memodifikasi aturan pemantauan kueri, lihat Membuat aturan pemantauan kueri dan Pro perti di Parameter wlm_json_configuration di Panduan Manajemen Amazon Redshift.

Anda dapat menemukan informasi selengkapnya tentang aturan pemantauan kueri dalam topik berikut:

- Metrik pemantauan kueri untuk Amazon Redshift disediakan
- Templat aturan pemantauan kueri
- Membuat aturan pemantauan kueri
- Mengkonfigurasi Manajemen Beban Kerja
- Tabel sistem dan tampilan untuk aturan pemantauan kueri

Metrik pemantauan kueri untuk Amazon Redshift disediakan

Tabel berikut menjelaskan metrik yang digunakan dalam aturan pemantauan kueri. (Metrik ini berbeda dari metrik yang disimpan dalam tabel STV_QUERY_METRICS dan STL_QUERY_METRICS sistem.)

Untuk metrik tertentu, ambang kinerja dilacak baik di tingkat kueri atau tingkat segmen. Untuk informasi selengkapnya tentang segmen dan langkah, lihatPerencanaan kueri dan alur kerja pelaksanaan.

> **Note**
>
> Batas waktu WLMParameter berbeda dari aturan pemantauan kueri.

Metrik	Nama	Deskripsi
Kueri waktu CPU	`query_cpu_time`	Waktu CPU yang digunakan oleh kueri, dalam detik. `CPU time`berbeda dari`Query execution time`. Nilai yang valid adalah 0-999.999.
Blok dibaca	`query_blocks_read`	Jumlah blok data 1 MB yang dibaca oleh kueri. Nilai yang valid adalah 0-1,048,575.
Pindai jumlah baris	`scan_row_count`	Jumlah baris dalam langkah pemindaian. Jumlah baris adalah jumlah total baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna. Nilai yang valid adalah 0-999.999.999.999.999.
Waktu eksekusi kueri	`query_execution_time`	Waktu eksekusi yang telah berlalu untuk kueri, dalam detik. Waktu eksekusi tidak termasuk

Metrik	Nama	Deskripsi
		waktu yang dihabiskan menunggu dalam antrian. Nilai yang valid adalah 0-86.399.
Waktu antrian kueri	`query_queue_time`	Waktu yang dihabiskan menunggu dalam antrian, dalam hitungan detik. Nilai yang valid adalah 0-86.399.
Penggunaan CPU	`query_cpu_usage_percent`	Persentase kapasitas CPU yang digunakan oleh query. Nilai yang valid adalah 0-6,399.
Memori ke disk	`query_temp_blocks_to_disk`	Ruang disk sementara digunakan untuk menulis hasil perantara, dalam blok 1 MB. Nilai yang valid adalah 0—319.815.679.
Kemiringan CPU	`cpu_skew`	Rasio penggunaan CPU maksimum untuk irisan apa pun terhadap penggunaan CPU rata-rata untuk semua irisan. Metrik ini didefinisikan pada tingkat segmen. Nilai yang valid adalah 0-99.
I/O miring	`io_skew`	Rasio blok maksimum read (I/O) untuk setiap irisan terhadap blok rata-rata yang dibaca untuk semua irisan. Metrik ini didefinisikan pada tingkat segmen. Nilai yang valid adalah 0-99.
Baris bergabung	`join_row_count`	Jumlah baris yang diproses dalam langkah gabungan. Nilai yang valid adalah 0-999.999.999.999.999.

Metrik	Nama	Deskripsi
Jumlah baris bergabung dengan loop bersarang	`nested_loop_join_row_count`	Jumlah atau baris dalam loop bersarang bergabung. Nilai yang valid adalah 0-999.999.999.999.999.
Kembalikan jumlah baris	`return_row_count`	Jumlah baris yang dikembalikan oleh query. Nilai yang valid adalah 0-999.999.999.999.999.
Waktu eksekusi segmen	`segment_execution_time`	Waktu eksekusi yang telah berlalu untuk satu segmen, dalam hitungan detik. Untuk menghindari atau mengurangi kesalahan pengambilan sampel, sertakan `segment_execution_time > 10` dalam aturan Anda. Nilai yang valid adalah 0-86.388.
Jumlah baris pemindaian spektrum	`spectrum_scan_row_count`	Jumlah baris data di Amazon S3 yang dipindai oleh kueri Amazon Redshift Spectrum. Nilai yang valid adalah 0-999.999.999.999.999.
Ukuran pemindaian spektrum	`spectrum_scan_size_mb`	Ukuran data di Amazon S3, dalam MB, dipindai oleh kueri Amazon Redshift Spectrum. Nilai yang valid adalah 0-999.999.999.999.999.
Prioritas kueri	`query_priority`	Prioritas kueri. Nilai yang valid adalah `HIGHEST`, `HIGH`, `NORMAL`, `LOW`, dan `LOWEST`. Ketika membandingkan `query_priority` menggunakan operator lebih besar dari (**>**) dan kurang dari (**<**), `HIGHEST` lebih besar dari`HIGH`, `HIGH` lebih besar dari`NORMAL`, dan seterusnya.

> **Note**
>
> - Tindakan hop tidak didukung dengan `query_queue_time` predikat. Artinya, aturan yang didefinisikan untuk melompat ketika `query_queue_time` predikat terpenuhi diabaikan.
> - Waktu eksekusi segmen yang singkat dapat mengakibatkan kesalahan pengambilan sampel dengan beberapa metrik, seperti `io_skew` dan`query_cpu_usage_percent`. Untuk menghindari atau mengurangi kesalahan pengambilan sampel, sertakan waktu eksekusi segmen dalam aturan Anda. Titik awal yang baik adalah`segment_execution_time > 10`.

T SVL_QUERY_METRICS ampilan menunjukkan metrik untuk kueri yang diselesaikan. T SVL_QUERY_METRICS_SUMMARY ampilan menunjukkan nilai metrik maksimum untuk kueri yang diselesaikan. Gunakan nilai-nilai dalam tampilan ini sebagai bantuan untuk menentukan nilai ambang batas untuk menentukan aturan pemantauan kueri.

Metrik pemantauan kueri untuk Amazon Redshift Serverless

Tabel berikut menjelaskan metrik yang digunakan dalam aturan pemantauan kueri untuk Amazon Redshift Serverless.

Metrik	Nama Predikat WLM	Nama	Deskripsi
Blok dibaca	`query_blocks_read`	`max_query_blocks_read`	Jumlah blok data 1 MB yang dibaca oleh kueri. Nilai yang valid adalah 0-1,048,575.
Pindai jumlah baris	`scan_row_count`	`max_scan_row_count`	Jumlah baris dalam langkah pemindaian. Jumlah baris adalah jumlah total baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna.

Metrik	Nama Predikat WLM	Nama	Deskripsi
			Nilai yang valid adalah 0-999.999 .999.999.999.
Waktu eksekusi kueri	`query_execution_time`	waktu_kueri_eksekusi maksimal	Waktu eksekusi yang telah berlalu untuk kueri, dalam detik. Waktu eksekusi tidak termasuk waktu yang dihabiskan menunggu dalam antrian. Jika kueri melebihi waktu eksekusi yang ditetapkan, Amazon Redshift Serverless menghentikan kueri. Nilai yang valid adalah 0-86.399.
Waktu antrian kueri	`query_queue_time`	`max_query_queue_time`	Waktu yang dihabiskan menunggu dalam antrian, dalam hitungan detik. Nilai yang valid adalah 0-86.399.
Memori ke disk	`query_temp_blocks_to_disk`	`max_query_temp_blocks_to_disk`	Ruang disk sementara digunakan untuk menulis hasil perantara, dalam blok 1 MB. Nilai yang valid adalah 0—319.815 .679.
Baris bergabung	`join_row_count`	`max_join_row_count`	Jumlah baris yang diproses dalam langkah gabungan. Nilai yang valid adalah 0-999.999 .999.999.999.
Jumlah baris bergabung dengan loop bersarang	`nested_loop_join_row_count`	`max_nested_loop_join_row_count`	Jumlah atau baris dalam loop bersarang bergabung. Nilai yang valid adalah 0-999.999 .999.999.999.

> **Note**
>
> - Aksi hop tidak didukung dengan `max_query_queue_time` predikat. Artinya, aturan yang didefinisikan untuk melompat ketika `max_query_queue_time` predikat terpenuhi diabaikan.
> - Waktu eksekusi segmen yang singkat dapat mengakibatkan kesalahan pengambilan sampel dengan beberapa metrik, seperti `max_io_skew` dan`max_query_cpu_usage_percent`.

Untuk Amazon Redshift Serverless, Anda dapat mengonfigurasi antrian kueri dan aturan pemantauan menggunakan parameter. `wlm_json_configuration` Ini memungkinkan Anda membuat beberapa antrian dengan peran pengguna, grup kueri, dan aturan pemantauan yang berbeda menggunakan metrik yang tercantum di atas. Untuk informasi selengkapnya tentang mengonfigurasi antrian kueri tanpa server, lihat Struktur konfigurasi JSON WLM di Panduan Manajemen Amazon Redshift.

Templat aturan pemantauan kueri

Saat menambahkan aturan menggunakan konsol Amazon Redshift, Anda dapat memilih untuk membuat aturan dari template yang telah ditentukan sebelumnya. Amazon Redshift membuat aturan baru dengan serangkaian predikat dan mengisi predikat dengan nilai default. Tindakan default adalah log. Anda dapat memodifikasi predikat dan tindakan untuk memenuhi kasus penggunaan Anda.

Tabel berikut mencantumkan template yang tersedia.

Nama Templat	Predikat	Deskripsi
Gabung loop bersarang	`nested_loop_join_row_count > 100`	Gabungan loop bersarang mungkin menunjukkan predikat gabungan yang tidak lengkap, yang sering menghasilkan set pengembalian yang sangat besar (produk Cartesian). Gunakan jumlah baris rendah untuk menemukan kueri yang berpotensi gagal lebih awal.
Query mengembalikan jumlah baris yang tinggi	`return_row_count > 1000000`	Jika Anda mendedikasikan antrian untuk kueri sederhana yang berjalan singkat, Anda mungkin menyertakan aturan yang menemukan

Nama Templat	Predikat	Deskripsi
		kueri yang mengembalikan jumlah baris tinggi. Template menggunakan default 1 juta baris. Untuk beberapa sistem, Anda mungkin menganggap satu juta baris tinggi, atau dalam sistem yang lebih besar, satu miliar baris atau lebih mungkin tinggi.
Bergabunglah dengan jumlah baris yang tinggi	`join_row_count > 1000000000`	Langkah gabungan yang melibatkan jumlah baris yang luar biasa tinggi mungkin menunjukkan perlunya filter yang lebih ketat. Template menggunakan default 1 miliar baris. Untuk antrian ad hoc (satu kali) yang ditujukan untuk kueri cepat dan sederhana, Anda dapat menggunakan angka yang lebih rendah.
Penggunaan disk yang tinggi saat menulis hasil menengah	`query_temp_blocks_to_disk > 100000`	Saat menjalankan kueri menggunakan lebih dari RAM sistem yang tersedia, mesin eksekusi kueri menulis hasil perantara ke disk (memori tumpah). Biasanya, kondisi ini adalah hasil dari kueri nakal, yang biasanya juga merupakan kueri yang menggunakan ruang disk paling banyak. Ambang batas yang dapat diterima untuk penggunaan disk bervariasi berdasarkan jenis node cluster dan jumlah node. Template menggunakan default 100.000 blok, atau 100 GB. Untuk cluster kecil, Anda dapat menggunakan angka yang lebih rendah.

Nama Templat	Predikat	Deskripsi
Kueri yang berjalan lama dengan kemiringan I/O tinggi	`segment_execution_time > 120` dan `io_skew > 1.30`	I/O kemiringan terjadi ketika satu irisan simpul memiliki I/O tingkat yang jauh lebih tinggi daripada irisan lainnya. Sebagai titik awal, kemiringan 1,30 (1,3 kali rata-rata) dianggap tinggi. Kem I/O iringan tinggi tidak selalu menjadi masalah, tetapi ketika dikombinasikan dengan waktu kueri yang berjalan lama, itu mungkin menunjukkan masalah dengan gaya distribusi atau kunci pengurutan.

Tabel sistem dan tampilan untuk aturan pemantauan kueri

Ketika semua predikat aturan terpenuhi, WLM menulis baris ke tabel STL_WLM_RULE_ACTION sistem. Baris ini berisi detail untuk kueri yang memicu aturan dan tindakan yang dihasilkan.

Selain itu, Amazon Redshift mencatat metrik kueri tabel dan tampilan sistem berikut.

- STV_QUERY_METRICSTabel menampilkan metrik untuk kueri yang sedang berjalan.
- STL_QUERY_METRICSTabel mencatat metrik untuk kueri yang diselesaikan.
- T SVL_QUERY_METRICS ampilan menunjukkan metrik untuk kueri yang diselesaikan.
- T SVL_QUERY_METRICS_SUMMARY ampilan menunjukkan nilai metrik maksimum untuk kueri yang diselesaikan.

Tabel dan tampilan sistem WLM

WLM mengonfigurasi antrian kueri menurut kelas layanan WLM, yang didefinisikan secara internal. Amazon Redshift membuat beberapa antrian internal sesuai dengan kelas layanan ini bersama dengan antrian yang ditentukan dalam konfigurasi WLM. Istil ah antrean dan kelas layanan sering digunakan secara bergantian dalam tabel sistem. Antrian superuser menggunakan kelas layanan 5. User-defined antrian menggunakan kelas layanan 6 dan lebih besar.

Anda dapat melihat status kueri, antrian, dan kelas layanan dengan menggunakan tabel WLM-specific sistem. Kueri tabel sistem berikut untuk melakukan hal berikut:

- Lihat kueri mana yang sedang dilacak dan sumber daya apa yang dialokasikan oleh pengelola beban kerja.
- Lihat antrian mana yang ditugaskan kueri.
- Melihat status kueri yang sedang dilacak oleh pengelola beban kerja.

Nama Tabel	Deskripsi
STL_WLM_ERROR	Berisi log peristiwa WLM-related kesalahan.
STL_WLM_QUERY	Daftar kueri yang sedang dilacak oleh WLM.
STV_WLM_CLASSICATION_CONFIG	Menunjukkan aturan klasifikasi saat ini untuk WLM.
STV_WLM_QUERY_QUEUE_STATE	Merekam keadaan antrian kueri saat ini.
STV_WLM_QUERY_STATE	Menyediakan snapshot dari keadaan kueri saat ini yang sedang dilacak oleh WLM.
STV_WLM_QUERY_TASKING_STATE	Berisi status tugas kueri saat ini.
STV_WLM_SERVICE_CLASS_CONFIG	Merekam konfigurasi kelas layanan untuk WLM.
STV_WLM_SERVICE_CLASS_STATE	Berisi status kelas layanan saat ini.
STL_WLM_RULE_ACTION	Merekam detail tentang tindakan yang dihasilkan dari aturan pemantauan kueri WLM yang terkait dengan antrian yang ditentukan pengguna.
STV_WLM_QMR_CONFIG	Merekam konfigurasi untuk aturan pemantauan kueri WLM (QMR).

Anda menggunakan ID tugas untuk melacak kueri dalam tabel sistem. Contoh berikut menunjukkan cara mendapatkan ID tugas dari kueri pengguna yang terakhir dikirimkan:

```
select task from stl_wlm_query where exec_start_time =(select max(exec_start_time) from
 stl_wlm_query);

task
------
137
(1 row)
```

Contoh berikut menampilkan kueri yang saat ini sedang dijalankan atau menunggu di berbagai kelas layanan (antrian). Kueri ini berguna dalam melacak keseluruhan beban kerja bersamaan untuk Amazon Redshift:

```
select * from stv_wlm_query_state order by query;


xid |task|query|service_| wlm_start_  |  state  |queue_ | exec_
    |    |     |class   | time        |         |time   | time
----+----+-----+--------+-------------+---------+-------+--------
2645| 84 | 98  | 3      | 2010-10-... |Returning|   0   | 3438369
2650| 85 | 100 | 3      | 2010-10-... |Waiting  |   0   | 1645879
2660| 87 | 101 | 2      | 2010-10-... |Executing|   0   | 916046
2661| 88 | 102 | 1      | 2010-10-... |Executing|   0   | 13291
(4 rows)
```

ID kelas layanan WLM

Tabel berikut mencantumkan ID yang ditetapkan untuk kelas layanan.

ID	Kelas layanan
1—4	Dicadangkan untuk penggunaan sistem.
5	Digunakan oleh antrian superuser.
6—13	Digunakan oleh antrian WLM manual yang ditentukan dalam konfigurasi WLM.

ID	Kelas layanan
14	Digunakan oleh akselerasi kueri pendek.
15	Dicadangkan untuk kegiatan pemeliharaan yang dijalankan oleh Amazon Redshift.
100—107	Digunakan oleh antrian WLM otomatis ketika auto_w lm benar.

Keamanan database

Anda mengelola keamanan database dengan mengontrol pengguna mana yang memiliki akses ke objek database mana. Pengguna dapat diberi peran atau grup, dan izin yang Anda berikan kepada pengguna, peran, atau grup menentukan objek database mana yang dapat mereka akses.

Topik

- Ikhtisar keamanan Amazon Redshift
- Izin pengguna basis data default
- Superuser
- Pengguna
- Grup
- Skema
- Role-based kontrol akses (RBAC)
- Row-level keamanan
- Keamanan metadata
- Penyembunyian data dinamis
- Izin terlingkup

Akses ke objek database tergantung pada izin yang Anda berikan kepada pengguna atau peran. Panduan berikut merangkum cara kerja keamanan database:

- Secara default, izin diberikan hanya kepada pemilik objek.
- Pengguna database Amazon Redshift adalah pengguna bernama yang dapat terhubung ke database. Pengguna diberikan izin dalam dua cara: secara eksplisit, dengan memiliki izin tersebut ditetapkan langsung ke akun, atau secara implisit, dengan menjadi anggota grup yang diberikan izin.
- Grup adalah kumpulan pengguna yang secara kolektif dapat diberi izin untuk pemeliharaan keamanan yang efisien.
- Skema adalah kumpulan tabel database dan objek database lainnya. Skema mirip dengan direktori sistem file, kecuali bahwa skema tidak dapat bersarang. Pengguna dapat diberikan akses ke skema tunggal atau beberapa skema.

Selain itu, Amazon Redshift menggunakan fitur-fitur berikut untuk memberi Anda kontrol yang lebih baik atas pengguna mana yang memiliki akses ke objek database mana:

- Dengan kontrol akses berbasis peran (RBAC), Anda dapat menetapkan izin ke peran yang kemudian dapat Anda terapkan ke pengguna, memungkinkan Anda mengontrol izin untuk kelompok besar pengguna. Tidak seperti grup, peran dapat mewarisi izin dari peran lain.

  Dengan keamanan tingkat baris (RLS), Anda dapat menentukan kebijakan yang membatasi akses ke baris pilihan Anda, lalu menerapkan kebijakan tersebut ke pengguna atau grup.

  Dynamic data masking (DDM) lebih melindungi data Anda dengan mengubahnya saat runtime kueri sehingga Anda dapat mengizinkan pengguna mengakses data tanpa mengekspos detail sensitif.

Untuk contoh implementasi keamanan, lihatContoh untuk mengontrol akses pengguna dan grup.

Untuk informasi selengkapnya tentang melindungi data Anda, lihat Keamanan di Amazon Redshift di Panduan Manajemen Amazon Redshift.

Ikhtisar keamanan Amazon Redshift

Keamanan database Amazon Redshift berbeda dari jenis keamanan Amazon Redshift lainnya. Selain keamanan database, yang dijelaskan di bagian ini, Amazon Redshift menyediakan fitur-fitur ini untuk mengelola keamanan:

- Sign-in kredenSIAL — Akses ke Amazon Redshift AWS Management Console dikendalikan oleh izin AWS akun Anda. Untuk informasi selengkapnya, lihat Sign-in kredensi.
- Manajemen akses — Untuk mengontrol akses ke sumber daya Amazon Redshift tertentu, Anda menentukan akun AWS Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat Meng ontrol akses ke sumber daya Amazon Redshift.
- Grup keamanan cluster — Untuk memberi pengguna lain akses masuk ke cluster Amazon Redshift, Anda menentukan grup keamanan cluster dan mengaitkannya dengan cluster. Untuk informasi selengkapnya, lihat grup keamanan cluster Amazon Redshift.
- VPC — Untuk melindungi akses ke cluster Anda dengan menggunakan lingkungan jaringan virtual, Anda dapat meluncurkan cluster Anda di Amazon Virtual Private Cloud (VPC). Untuk informasi selengkapnya, lihat Mengelola cluster di Virtual Private Cloud (VPC).

- Enkripsi cluster — Untuk mengenkripsi data di semua tabel yang dibuat pengguna, Anda dapat mengaktifkan enkripsi cluster saat meluncurkan cluster. Untuk informasi selengkapnya, lihat cluster Amazon Redshift.
- Koneksi SSL — Untuk mengenkripsi koneksi antara klien SQL Anda dan cluster Anda, Anda dapat menggunakan enkripsi Secure Sockets Layer (SSL). Untuk informasi selengkapnya, lihat Menghubungkan ke cluster menggunakan SSL.
- Muat enkripsi data — Untuk mengenkripsi file data beban tabel saat Anda mengunggahnya ke Amazon S3, Anda dapat menggunakan enkripsi sisi server atau enkripsi sisi klien. Saat Anda memuat dari data terenkripsi sisi server, Amazon S3 menangani dekripsi secara transparan. Saat Anda memuat dari data terenkripsi sisi klien, perintah Amazon Redshift COPY mendekripsi data saat memuat tabel. Untuk informasi selengkapnya, lihat Mengunggah data terenkripsi ke Amazon S3.
- Data dalam perjalanan — Untuk melindungi data Anda dalam transit di dalam AWS Cloud, Amazon Redshift menggunakan SSL yang dipercepat perangkat keras untuk berkomunikasi dengan Amazon S3 atau Amazon DynamoDB untuk operasi COPY, UNLOAD, backup, dan restore.
- Column-level kontrol akses — Untuk memiliki kontrol akses tingkat kolom untuk data di Amazon Redshift, gunakan pernyataan hibah dan cabut tingkat kolom tanpa harus menerapkan kontrol akses berbasis tampilan atau menggunakan sistem lain.
- Row-level kontrol keamanan — Untuk memiliki kontrol keamanan tingkat baris untuk data di Amazon Redshift, buat dan lampirkan kebijakan ke peran atau pengguna yang membatasi akses ke baris yang ditentukan dalam kebijakan.

Izin pengguna basis data default

Ketika Anda membuat objek database, Anda adalah pemiliknya. Secara default, hanya pengguna super atau pemilik objek yang dapat menanyakan, memodifikasi, atau memberikan izin pada objek. Agar pengguna dapat menggunakan objek, Anda harus memberikan izin yang diperlukan kepada pengguna atau grup yang berisi pengguna. Superuser database memiliki izin yang sama dengan pemilik database.

Amazon Redshift mendukung izin berikut: PILIH, SISIPKAN, PERBARUI, HAPUS, REFERENSI, BUAT, SEMENTARA, dan PEMAKAIAN. Izin yang berbeda dikaitkan dengan jenis objek yang berbeda. Untuk informasi tentang izin objek database yang didukung oleh Amazon Redshift, lihat HIBAH perintah.

Hanya pemilik yang memiliki izin untuk memodifikasi atau menghancurkan objek.

Secara default, semua pengguna memiliki izin CREATE dan USAGE pada skema PUBLIK database. Untuk melarang pengguna membuat objek dalam skema PUBLIK database, gunakan perintah REVOKE untuk menghapus izin tersebut.

Untuk mencabut izin yang sebelumnya diberikan, gunakan MENCABUT perintah. Izin pemilik objek, seperti izin DROP, GRANT, dan REVOKE, bersifat implisit dan tidak dapat diberikan atau dicabut. Pemilik objek dapat mencabut izin biasa mereka sendiri, misalnya, untuk membuat tabel hanya dapat dibaca untuk diri mereka sendiri dan orang lain. Pengguna super mempertahankan semua izin terlepas dari perintah GRANT dan REVOKE.

Superuser

Superuser database memiliki izin yang sama dengan pemilik database untuk semua database.

Pengguna admin, yang merupakan pengguna yang Anda buat saat meluncurkan cluster, adalah pengguna super.

Anda harus menjadi superuser untuk membuat superuser.

Tabel sistem Amazon Redshift dan tampilan sistem hanya terlihat oleh pengguna super atau terlihat oleh semua pengguna. Hanya pengguna super yang dapat menanyakan tabel sistem dan tampilan sistem yang ditetapkan “terlihat oleh pengguna super.” Untuk informasi, lihat Tampilan pemantauan SYS.

Superuser dapat melihat semua tabel katalog. Untuk informasi, lihat Tabel katalog sistem.

Superuser database melewati semua pemeriksaan izin. Pengguna super mempertahankan semua izin terlepas dari perintah GRANT dan REVOKE. Hati-hati saat menggunakan peran superuser. Kami menyarankan Anda melakukan sebagian besar pekerjaan Anda sebagai peran yang bukan superuser. Anda dapat membuat peran administrator dengan izin yang lebih ketat. Untuk informasi selengkapnya tentang membuat peran, lihat Role-based kontrol akses (RBAC)

Untuk membuat superuser database baru, masuk ke database sebagai superuser dan keluarkan perintah CREATE USER atau perintah ALTER USER dengan izin CREATEUSER.

```
CREATE USER adminuser CREATEUSER PASSWORD '1234Admin';
ALTER USER adminuser CREATEUSER;
```

Untuk membuat, mengubah, atau menghapus superuser, gunakan perintah yang sama untuk mengelola pengguna. Untuk informasi selengkapnya, lihat Membuat, mengubah, dan menghapus pengguna.

Pengguna

Anda dapat membuat dan mengelola pengguna database menggunakan perintah Amazon Redshift SQL CREATE USER dan ALTER USER. Atau Anda dapat mengonfigurasi klien SQL Anda dengan driver Amazon Redshift JDBC atau ODBC khusus. Ini mengelola proses pembuatan pengguna basis data dan kata sandi sementara sebagai bagian dari proses logon database.

Driver mengotentikasi pengguna basis data berdasarkan otentik AWS Identity and Access Management asi (IAM). Jika sudah mengelola identitas pengguna di luar AWS, Anda dapat menggunakan penyedia identitas (IdP) yang sesuai dengan SAML 2.0 untuk mengelola akses ke sumber daya Amazon Redshift. Anda menggunakan peran IAM untuk mengonfigurasi IdP dan mengiz AWS inkan pengguna federasi menghasilkan kredentif basis data sementara dan masuk ke database Amazon Redshift. Untuk informasi selengkapnya, lihat Menggunakan otentikasi IAM untuk menghasilkan kredenSIAL pengguna database.

Pengguna Amazon Redshift hanya dapat dibuat dan dijatuhkan oleh superuser database. Pengguna diautentikasi saat mereka masuk ke Amazon Redshift. Mereka dapat memiliki database dan objek database (misalnya, tabel). Mereka juga dapat memberikan izin pada objek tersebut kepada pengguna, grup, dan skema untuk mengontrol siapa yang memiliki akses ke objek mana. Pengguna dengan hak CREATE DATABASE dapat membuat database dan memberikan izin ke database tersebut. Superuser memiliki izin kepemilikan database untuk semua database.

Membuat, mengubah, dan menghapus pengguna

Pengguna database bersifat global di seluruh cluster gudang data (dan bukan untuk setiap database individu).

- Untuk membuat pengguna, gunakan BUAT PENGGUNA perintah.
- Untuk membuat superuser, gunakan BUAT PENGGUNA perintah dengan opsi CREATEUSER.
- Untuk menghapus pengguna yang ada, gunakan JATUHKAN PENGGUNA perintah.
- Untuk mengubah pengguna, misalnya mengubah kata sandi, gunakan ALTER USER perintah.
- Untuk melihat daftar pengguna, kueri tabel katalog PG_USER.

```
select * from pg_user;
```

```
  usename   | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil |
 useconfig
------------+----------+-------------+----------+-----------+----------+----------
+-----------
 rdsdb      |        1 | t           | t        | t         | ******** |          |
 masteruser |      100 | t           | t        | f         | ******** |          |
 dwuser     |      101 | f           | f        | f         | ******** |          |
 simpleuser |      102 | f           | f        | f         | ******** |          |
 poweruser  |      103 | f           | t        | f         | ******** |          |
 dbuser     |      104 | t           | f        | f         | ******** |          |
(6 rows)
```

Penguncian pengguna untuk pengguna database

Dengan Amazon Redshift, Anda dapat melindungi pengguna basis data berbasis kata sandi dari serangan kata sandi brute-force. Amazon Redshift melacak upaya masuk kata sandi yang gagal berturut-turut untuk setiap pengguna yang Anda buat. Ketika jumlah upaya yang gagal mencapai ambang batas, Amazon Redshift mengunci pengguna dan menolak upaya koneksi lebih lanjut sampai pengguna super atau pengguna dengan hak istimewa ALTER USER membuka kunci pengguna. Masuk yang berhasil mengatur ulang konter. Perlindungan ini diaktifkan secara default.

Seorang superuser menetapkan ambang batas untuk cluster dengan parameter upayan_login_gagal maksimal konfigurasi. Defaultnya adalah 5 upaya gagal berturut-turut, dan Anda dapat mengatur nilai dari 2 hingga 50. Anda mengatur parameter ini dengan menggunakan perintah ALTER SYSTEM SET, seperti yang ditunjukkan dalam contoh berikut.

```
ALTER SYSTEM SET max_failed_login_attempts = 10;
```

Penguncian pengguna hanya berlaku untuk pengguna basis data berbasis kata sandi. Pengguna federasi tidak terpengaruh, termasuk pengguna yang melakukan otentikasi melalui AWS Identity and Access Management (IAM), Pusat Identitas AWS IAM, atau penyedia identitas asli (IdP). Amazon Redshift juga tidak secara otomatis mengunci pengguna admin.

Mengunci dan membuka kunci pengguna

Selain penguncian otomatis, pengguna super atau pengguna dengan hak istimewa ALTER USER dapat mengunci dan membuka kunci pengguna secara manual.

- Untuk mengunci pengguna secara manual, gunakan ALTER USER dengan opsi NOLOGIN. Amazon Redshift kemudian menolak upaya koneksi pengguna tersebut. Mengunci pengguna tidak mengakhiri sesi pengguna yang ada.

```
ALTER USER data_analyst NOLOGIN;
```

- Untuk membuka kunci pengguna, gunakan ALTER USER dengan opsi LOGIN PASSWORD dan berikan kata sandi baru. Membuka kunci pengguna akan mengatur ulang penghitung login gagal pengguna tersebut.

```
ALTER USER data_analyst LOGIN PASSWORD 'NewStr0ngP@ssword';
```

Untuk informasi selengkapnya tentang opsi ini, lihat ALTER USER.

Mengaudit pengguna yang terkunci

Untuk meninjau pengguna mana yang saat ini terkunci, lihat apakah setiap penguncian dilakukan secara otomatis atau manual, dan tinjau aktivitas login gagal baru-baru ini, superuser dapat menjalankan perintah SHOW USER LOCKOUT. Untuk informasi selengkapnya, lihat TAMPILKAN PENGUNCIAN PENGGUNA.

Mendapatkan kembali akses jika pengguna admin terkunci

Seorang superuser atau pengguna dengan hak istimewa ALTER USER membuka kunci pengguna yang terkunci dengan perintah ALTER USER LOGIN PASSWORD. Jika pengguna admin terkunci dan tidak ada superuser lain yang dapat masuk, Anda dapat mengatur ulang kata sandi admin untuk mendapatkan kembali akses. Menyetel ulang kata sandi admin mengembalikan kemampuan pengguna admin untuk masuk.

- Untuk cluster yang disediakan, gunakan operasi ModifyCluster API, atau AWS CLI perintah Konsol Manajemen AWS atau yang setara, untuk menetapkan kata sandi admin baru.
- Untuk Amazon Redshift Serverless, gunakan operasi UpdateNamespace API, atau AWS CLI perintah Konsol Manajemen AWS atau yang setara, untuk menetapkan kata sandi admin baru. Anda harus memberikan keduanya `adminUsername` dan `adminUserPassword` dalam permintaan yang sama.

Grup

Grup adalah kumpulan pengguna yang semuanya diberikan izin apa pun yang terkait dengan grup. Anda dapat menggunakan grup untuk menetapkan izin. Misalnya, Anda dapat membuat grup yang berbeda untuk penjualan, administrasi, dan dukungan dan memberikan pengguna di setiap grup akses yang sesuai ke data yang mereka butuhkan untuk pekerjaan mereka. Anda dapat memberikan atau mencabut izin di tingkat grup, dan perubahan tersebut akan berlaku untuk semua anggota grup, kecuali untuk pengguna super.

Untuk melihat semua grup pengguna, kueri tabel katalog sistem PG_GROUP:

```
select * from pg_group;
```

Misalnya, untuk mencantumkan semua pengguna database berdasarkan grup, jalankan SQL berikut.

```
SELECT u.usesysid
,g.groname
,u.usename
FROM pg_user u
LEFT JOIN pg_group g ON u.usesysid = ANY (g.grolist)
```

Membuat, mengubah, dan menghapus grup

Hanya pengguna super yang dapat membuat, mengubah, atau menghapus grup.

Anda dapat melakukan tindakan berikut:

- Untuk membuat grup, gunakan BUAT GRUP perintah.
- Untuk menambahkan pengguna ke atau menghapus pengguna dari grup yang ada, gunakan MENGUBAH KELOMPOK perintah.
- Untuk menghapus grup, gunakan KELOMPOK DROP perintah. Perintah ini hanya menghapus grup, bukan pengguna anggotanya.

Contoh untuk mengontrol akses pengguna dan grup

Contoh ini membuat grup pengguna dan pengguna dan kemudian memberi mereka berbagai izin untuk database Amazon Redshift yang terhubung ke klien aplikasi web. Contoh ini mengasumsikan

tiga kelompok pengguna: pengguna reguler aplikasi web, pengguna kuat aplikasi web, dan pengembang web.

Untuk informasi tentang cara menghapus pengguna dari grup, lihatMENGUBAH KELOMPOK.

1. Buat grup tempat pengguna akan ditugaskan. Kumpulan perintah berikut membuat tiga grup pengguna yang berbeda:

```
create group webappusers;

create group webpowerusers;

create group webdevusers;
```

2. Buat beberapa pengguna database dengan izin yang berbeda dan tambahkan mereka ke grup.

   a. Buat dua pengguna dan tambahkan mereka ke grup WEBAPPUSERS:

```
create user webappuser1 password 'webAppuser1pass'
in group webappusers;

create user webappuser2 password 'webAppuser2pass'
in group webappusers;
```

   b. Buat pengguna pengembang web dan tambahkan ke grup WEBDEVUSERS:

```
create user webdevuser1 password 'webDevuser2pass'
in group webdevusers;
```

   c. Buat superuser. Pengguna ini akan memiliki hak administratif untuk membuat pengguna lain:

```
create user webappadmin  password 'webAppadminpass1'
createuser;
```

3. Buat skema untuk dikaitkan dengan tabel database yang digunakan oleh aplikasi web, dan berikan berbagai kelompok pengguna akses ke skema ini:

   a. Buat skema WEBAPP:

```
create schema webapp;
```

   b. Berikan izin USAGE ke grup WEBAPPUSERS:

```
grant usage on schema webapp to group webappusers;
```

c. Berikan izin USAGE ke grup WEBPOWERUSERS:

```
grant usage on schema webapp to group webpowerusers;
```

d. Berikan SEMUA izin ke grup WEBDEVUSERS:

```
grant all on schema webapp to group webdevusers;
```

Pengguna dan grup dasar sekarang disiapkan. Anda sekarang dapat mengubah pengguna dan grup.

4. Misalnya, perintah berikut mengubah parameter search_path untuk WEBAPPUSER1.

```
alter user webappuser1 set search_path to webapp, public;
```

SEARCH_PATH menentukan urutan pencarian skema untuk objek database, seperti tabel dan fungsi, ketika objek direferensikan oleh nama sederhana tanpa skema yang ditentukan.

5. Anda juga dapat menambahkan pengguna ke grup setelah membuat grup, seperti menambahkan WEBAPPUSER2 ke grup WEBPOWERUSERS:

```
alter group webpowerusers add user webappuser2;
```

Skema

Database berisi satu atau lebih skema bernama. Setiap skema dalam database berisi tabel dan jenis objek bernama lainnya. Secara default, database memiliki skema tunggal, yang bernama PUBLIC. Anda dapat menggunakan skema untuk mengelompokkan objek database di bawah nama umum. Skema mirip dengan direktori sistem file, kecuali bahwa skema tidak dapat bersarang.

Nama objek database yang identik dapat digunakan dalam skema yang berbeda dalam database yang sama tanpa konflik. Misalnya, MY_SCHEMA dan YOUR_SCHEMA dapat berisi tabel bernama MYTABLE. Pengguna dengan izin yang diperlukan dapat mengakses objek di beberapa skema dalam database.

Secara default, sebuah objek dibuat dalam skema pertama di jalur pencarian database. Untuk informasi, lihat Jalur pencarian nanti di bagian ini.

Skema dapat membantu masalah organisasi dan konkurensi di lingkungan multipengguna dengan cara berikut:

- Untuk membiarkan banyak pengembang bekerja dalam database yang sama tanpa mengganggu satu sama lain.
- Untuk mengatur objek database ke dalam kelompok logis untuk membuatnya lebih mudah dikelola.
- Untuk memberikan aplikasi kemampuan untuk menempatkan objek mereka ke dalam skema terpisah sehingga nama mereka tidak akan bertabrakan dengan nama objek yang digunakan oleh aplikasi lain.

Jalur pencarian

Jalur pencarian didefinisikan dalam parameter search_path dengan daftar nama skema yang dipisahkan koma. Jalur pencarian menentukan urutan di mana skema dicari ketika objek, seperti tabel atau fungsi, direferensikan oleh nama sederhana yang tidak menyertakan kualifier skema.

Jika sebuah objek dibuat tanpa menentukan skema target, objek ditambahkan ke skema pertama yang terdaftar di jalur pencarian. Ketika objek dengan nama yang identik ada dalam skema yang berbeda, nama objek yang tidak menentukan skema akan merujuk ke skema pertama di jalur pencarian yang berisi objek dengan nama itu.

Untuk mengubah skema default untuk sesi saat ini, gunakan SET perintah.

Untuk informasi selengkapnya, lihat search_path deskripsi di Referensi Konfigurasi.

Membuat, mengubah, dan menghapus skema

Setiap pengguna dapat membuat skema dan mengubah atau menghapus skema yang mereka miliki.

Anda dapat melakukan tindakan berikut:

- Untuk membuat skema, gunakan BUAT SKEMA perintah.
- Untuk mengubah pemilik skema, gunakan ALTER SCHEMA perintah.
- Untuk menghapus skema dan objeknya, gunakan DROP SCHEMA perintah.

- Untuk membuat tabel dalam skema, buat tabel dengan format schema_ name.table_name.

Untuk melihat daftar semua skema, kueri tabel katalog sistem PG_NAMESPACE:

```
select * from pg_namespace;
```

Untuk melihat daftar tabel yang termasuk dalam skema, kueri tabel katalog sistem PG_TABLE_DEF. Misalnya, kueri berikut mengembalikan daftar tabel dalam skema PG_CATALOG.

```
select distinct(tablename) from pg_table_def
where schemaname = 'pg_catalog';
```

Schema-based izin

Schema-based izin ditentukan oleh pemilik skema:

- Secara default, semua pengguna memiliki izin CREATE dan USAGE pada skema PUBLIK database. Untuk melarang pengguna membuat objek dalam skema PUBLIK database, gunakan MENCABUT perintah untuk menghapus izin tersebut.
- Kecuali mereka diberikan izin USAGE oleh pemilik objek, pengguna tidak dapat mengakses objek apa pun dalam skema yang tidak mereka miliki.
- Jika pengguna telah diberikan izin CREATE ke skema yang dibuat oleh pengguna lain, pengguna tersebut dapat membuat objek dalam skema tersebut.

Role-based kontrol akses (RBAC)

Dengan menggunakan kontrol akses berbasis peran (RBAC) untuk mengelola izin database di Amazon Redshift, Anda dapat menyederhanakan pengelolaan izin keamanan di Amazon Redshift. Anda dapat mengamankan akses ke data sensitif dengan mengontrol apa yang dapat dilakukan pengguna baik pada tingkat yang luas atau baik. Anda juga dapat mengontrol akses pengguna ke tugas yang biasanya dibatasi untuk pengguna super. Dengan menetapkan izin yang berbeda untuk peran yang berbeda dan menetapkannya ke pengguna yang berbeda, Anda dapat memiliki kontrol akses pengguna yang lebih terperinci.

Pengguna dengan peran yang ditugaskan hanya dapat melakukan tugas yang ditentukan oleh peran yang ditugaskan untuk mereka otorisasi. Misalnya, pengguna dengan peran yang ditugaskan yang memiliki izin CREATE TABLE dan DROP TABLE hanya diizinkan untuk melakukan tugas-tugas

tersebut. Anda dapat mengontrol akses pengguna dengan memberikan izin keamanan tingkat yang berbeda kepada pengguna yang berbeda untuk mengakses data yang mereka butuhkan untuk pekerjaan mereka.

RBAC menerapkan prinsip izin paling sedikit kepada pengguna berdasarkan persyaratan peran mereka, terlepas dari jenis objek yang terlibat. Pemberian dan pencabutan izin dilakukan pada tingkat peran, tanpa perlu memperbarui izin pada objek database individual.

Dengan RBAC, Anda dapat membuat peran dengan izin untuk menjalankan perintah yang digunakan untuk memerlukan izin superuser. Pengguna dapat menjalankan perintah ini, selama mereka diotorisasi dengan peran yang menyertakan izin ini. Demikian pula, Anda juga dapat membuat peran untuk membatasi akses ke perintah tertentu, dan menetapkan peran untuk pengguna super atau pengguna yang telah diotorisasi dengan peran tersebut.

Untuk mempelajari cara kerja Amazon Redshift RBAC, tonton video berikut: Memperkenalkan kontrol Role-based akses (RBAC) di Amazon Redshift.

Hirarki peran

Per an adalah kumpulan izin yang dapat Anda tetapkan ke pengguna atau peran lain. Anda dapat menetapkan izin sistem atau database ke peran. Pengguna mewarisi izin dari peran yang ditetapkan.

Di RBAC, pengguna dapat memiliki peran bersarang. Anda dapat memberikan peran kepada pengguna dan peran. Saat memberikan peran kepada pengguna, Anda mengotorisasi pengguna dengan semua izin yang disertakan peran ini. Saat memberikan peran r1 kepada pengguna, Anda mengotorisasi pengguna dengan izin dari r1. Pengguna sekarang memiliki izin dari r1 dan juga izin yang sudah ada yang sudah mereka miliki.

Saat memberikan peran (r1) ke peran lain (r2), Anda mengotorisasi r2 dengan semua izin dari r1. Juga, ketika memberikan r2 ke peran lain (r3), izin r3 adalah kombinasi dari izin dari r1 dan r2. Hirarki peran memiliki r2 mewarisi izin dari r1. Amazon Redshift menyebarkan izin dengan setiap otorisasi peran. Memberikan r1 ke r2 dan kemudian r2 ke r3 mengotorisasi r3 dengan semua izin dari tiga peran. Dengan demikian, dengan memberikan r3 kepada pengguna, pengguna memiliki semua izin dari tiga peran.

Amazon Redshift tidak mengizinkan pembuatan siklus otorisasi peran. Siklus otorisasi peran terjadi ketika peran bersarang ditugaskan kembali ke peran sebelumnya dalam hierarki peran, seperti r3 ditugaskan kembali ke r1. Untuk informasi selengkapnya tentang cara membuat peran dan mengelola penugasan peran, lihatMengelola peran di RBAC.

Penetapan peran

Pengguna super dan pengguna biasa dengan izin CREATE ROLE dapat menggunakan pernyataan CREATE ROLE untuk membuat peran. Pengguna super dan administrator peran dapat menggunakan pernyataan GRANT ROLE untuk memberikan peran kepada orang lain. Mereka dapat menggunakan pernyataan REVOKE ROLE untuk mencabut peran dari orang lain, dan pernyataan DROP ROLE untuk menghapus peran. Administrator peran mencakup pemilik peran dan pengguna yang telah diberikan peran dengan izin ADMIN OPTION.

Hanya pengguna super atau administrator peran yang dapat memberikan dan mencabut peran. Anda dapat memberikan atau mencabut satu atau lebih peran ke atau dari satu atau lebih peran atau pengguna. Gunakan opsi WITH ADMIN OPTION dalam pernyataan GRANT ROLE untuk menyediakan opsi administrasi untuk semua peran yang diberikan kepada semua penerima hibah.

Amazon Redshift mendukung kombinasi penugasan peran yang berbeda, seperti pemberian beberapa peran atau memiliki beberapa penerima hibah. OPSI DENGAN ADMIN hanya berlaku untuk pengguna dan bukan untuk peran. Demikian pula, gunakan opsi WITH ADMIN OPTION dalam pernyataan REVOKE ROLE untuk menghapus peran dan otorisasi administratif dari penerima hibah. Saat digunakan dengan ADMIN OPTION, hanya otorisasi administratif yang dicabut dari peran.

Contoh berikut mencabut otorisasi administratif `sample_role2` peran dari. `user2`

```
REVOKE ADMIN OPTION FOR sample_role2 FROM user2;
```

Untuk informasi selengkapnya tentang cara membuat peran dan mengelola penugasan peran, lihatMengelola peran di RBAC.

Peran yang ditentukan sistem Amazon Redshift

Amazon Redshift menyediakan beberapa peran yang ditentukan sistem yang ditentukan dengan izin tertentu. System-specific peran dimulai dengan `sys:` awalan. Hanya pengguna dengan akses yang sesuai yang dapat mengubah peran yang ditentukan sistem atau membuat peran yang ditentukan sistem khusus. Anda tidak dapat menggunakan `sys:` awalan untuk peran yang ditentukan sistem khusus.

Tabel berikut merangkum peran dan izinnya.

Nama peran	Deskripsi		
`sys:monitor`	Peran ini memiliki izin untuk mengakses katalog atau tabel sistem.		
`sys:operator`	Peran ini memiliki izin untuk mengakses katalog atau tabel sistem, menganalisis, menyedot, atau membatalkan kueri.		
`sys:dba`	Peran ini memiliki izin untuk membuat skema, membuat tabel, melepas skema, melepas tabel, dan memotong tabel. Ini memiliki izin untuk membuat atau mengganti prosedur yang disimpan, menjatuhkan prosedur, membuat atau mengganti fungsi, membuat atau mengganti fungsi eksternal, membuat tampilan, dan melepaskan tampilan. Juga, peran ini mewarisi semua izin dari peran sys:operator.		

Nama peran	Deskripsi		
`sys:superuser`	Peran ini memiliki semua izin sistem yang didukung yang ditentukan diIzin sistem untuk RBAC.		

Nama peran	Deskripsi		
`sys:secadmin`	• Peran ini memiliki izin untuk membuat pengguna, mengubah pengguna, menghapus pengguna, membuat peran, menghapus peran, dan memberikan peran. • Peran ini memiliki izin untuk mengaktifkan atau menonaktifkan RLS pada relasi dan izin untuk mengelola kebijakan RLS dan DDM (CREATE, DROP, ATTACH, DETACH, dan ALTER). Juga, perhatikan bahwa izin EXPLAIN RLS, IGNORE RLS, dan EXPLAIN MASKING diberikan ke peran ini secara default. • Peran ini dapat memiliki akses ke tabel pengguna hanya jika izin secara eksplisit		

Nama peran	Deskripsi		
	diberikan ke peran tersebut.		

System-defined peran dan pengguna untuk berbagi data

Amazon Redshift menciptakan peran dan pengguna untuk penggunaan internal yang sesuai dengan data share dan konsumen datashare. Setiap nama peran internal dan nama pengguna memiliki awalan namespace yang dicadangkan. `ds:` Mereka memiliki format berikut:

Nama	Deskripsi		
`ds:`*`sharename`*	Peran sistem yang sesuai dengan datashare.		
`ds:`*`sharename_consumer`*	Pengguna sistem yang berhubungan dengan konsumen datashare.		

Peran berbagi data dibuat untuk setiap datashare. Ini memegang semua izin yang saat ini diberikan ke datashare. Pengguna berbagi data dibuat untuk setiap konsumen datashare. Ini diberikan izin untuk peran berbagi data tunggal. Konsumen yang ditambahkan ke beberapa datashares akan memiliki pengguna berbagi data yang dibuat untuk setiap datashare.

Pengguna dan peran ini diperlukan agar berbagi data berfungsi dengan baik. Mereka tidak dapat dimodifikasi atau dijatuhkan dan mereka tidak dapat diakses atau digunakan untuk tugas apa pun yang dijalankan oleh pelanggan. Anda dapat dengan aman mengabaikannya. Untuk informasi selengkapnya tentang berbagi data, lihat Berbagi data lintas cluster di Amazon Redshift.

> **Note**
>
> Anda tidak dapat menggunakan `ds:` awalan untuk membuat peran atau pengguna yang ditentukan pengguna.

Izin sistem untuk RBAC

Berikut ini adalah daftar izin sistem yang dapat Anda berikan atau cabut dari peran.

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
CREATE ROLE	• Pengguna super. • Pengguna dengan izin CREATE ROLE.		
DROP ROLE	• Pengguna super. • Pemilik peran yang merupakan pengguna yang membuat peran atau pengguna yang telah diberikan peran dengan izin WITH ADMIN OPTION.		
BUAT PENGGUNA	• Pengguna super. • Pengguna dengan izin CREATE USER. Pengguna ini tidak dapat membuat pengguna super.		
JATUHKAN PENGGUNA	• Pengguna super. • Pengguna dengan izin DROP USER.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
ALTER USER	• Pengguna super. • Pengguna dengan izin ALTER USER. Pengguna ini tidak dapat mengubah pengguna menjadi pengguna super atau mengubah pengguna super menjadi pengguna. • Pengguna saat ini yang ingin mengubah kata sandi mereka sendiri.		
BUAT SKEMA	• Pengguna super. • Pengguna dengan izin CREATE SCHEMA.		
DROP SCHEMA	• Pengguna super. • Pengguna dengan izin DROP SCHEMA. • Pemilik skema.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
UBAH HAK ISTIMEWA DEFAULT	• Pengguna super. • Pengguna dengan izin ALTER DEFAULT PRIVILEGES. • Pengguna mengubah izin akses default mereka sendiri. • Pengguna mengatur izin untuk skema yang mereka miliki izin aksesnya.		
AKSES KATALOG	• Pengguna super. • Pengguna dengan izin ACCESS CATALOG.		
TABEL SISTEM AKSES	• Pengguna super. • Pengguna dengan izin TABEL SISTEM AKSES.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
CREATE TABLE	• Pengguna super. • Pengguna dengan izin CREATE TABLE. • Pengguna dengan izin CREATE pada skema.		
MEJA DROP	• Pengguna super. • Pengguna dengan izin DROP TABLE. • Pemilik tabel dengan izin USAGE pada skema.		
ALTER TABLE	• Pengguna super. • Pengguna dengan izin ALTER TABLE. • Pemilik tabel dengan izin USAGE pada skema.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		

MEMBUAT ATAU MENGGANTI FUNGSI	• Untuk FUNGSI BUAT:
	  • Pengguna super.
	  • Pengguna dengan izin CREATE OR REPLACE FUNCTION.
	  • Pengguna dengan izin USAGE pada bahasa.
	• Untuk FUNGSI GANTI:
	  • Pengguna super.
	  • Pengguna dengan izin CREATE OR REPLACE FUNCTION.
	  • Pemilik fungsi.		

MEMBUAT ATAU MENGGANTI FUNGSI EKSTERNAL	• Pengguna super. • Pengguna dengan izin CREATE OR REPLACE EXTERNAL FUNCTION.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
FUNGSI DROP	• Pengguna super. • Pengguna dengan izin DROP FUNCTION. • Pemilik fungsi.		

MEMBUAT ATAU MENGGANTI PROSEDUR	• Untuk MEMBUAT PROSEDUR:
	  • Pengguna super.
	  • Pengguna dengan izin BUAT ATAU GANTI PROSEDUR.
	  • Pengguna dengan izin USAGE pada bahasa.
	• Untuk PROSEDUR GANTI:
	  • Pengguna super.
	  • Pengguna dengan izin BUAT ATAU GANTI PROSEDUR.
	  • Pemilik prosedur.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
PROSEDUR DROP	• Pengguna super. • Pengguna dengan izin DROP PROCEDURE. • Pemilik prosedur.		

MEMBUAT ATAU MENGGANTI TAMPILAN	• Untuk CREATE VIEW:
	  • Pengguna super.
	  • Pengguna dengan izin CREATE OR REPLACE VIEW.
	  • Pengguna dengan izin CREATE pada skema.
	• Untuk TAMPILAN GANTI:
	  • Pengguna super.
	  • Pengguna dengan izin CREATE OR REPLACE VIEW.
	  • Lihat pemilik.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
MENJATUHKAN TAMPILAN	• Pengguna super. • Pengguna dengan izin DROP VIEW. • Lihat pemilik.		
BUAT MODEL	• Pengguna super. • Pengguna dengan izin sistem CREATE MODEL, yang seharusnya dapat membaca relasi CREATE MODEL. • Pengguna dengan izin CREATE MODEL.		
MODEL JATUHKAN	• Pengguna super. • Pengguna dengan izin DROP MODEL. • Pemilik model. • Pemilik skema.		
BUAT DATASHARE	• Pengguna super. • Pengguna dengan izin CREATE DATASHARE. • Pemilik database.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
ALTER DATA	• Pengguna super. • Pengguna dengan izin ALTER DATASHARE. • Pengguna yang memiliki izin ALTER atau ALL pada datashare. • Untuk menambahkan objek tertentu ke datashare, pengguna ini harus memiliki izin pada objek. Pengguna harus menjadi pemilik objek atau memiliki izin SELECT, USAGE, atau ALL pada objek.		
DROP DATA	• Pengguna super. • Pengguna dengan izin DROP DATASHARE. • Pemilik database.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
BUAT PERPUSTAKAAN	• Pengguna super. • Pengguna dengan izin CREATE LIBRARY atau dengan izin bahasa yang ditentukan.		
JATUHKAN PERPUSTAKAAN	• Pengguna super. • Pengguna dengan izin DROP LIBRARY. • Pemilik perpustakaan.		
MENGANALISA	• Pengguna super. • Pengguna dengan izin ANALYSIS. • Pemilik hubungan. • Pemilik database kepada siapa tabel dibagikan.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
CANCEL (BATALKAN)	• Superuser membatalkan kueri mereka sendiri. • Superuser membatalkan kueri pengguna. • Pengguna dengan izin BATAL membatalkan kueri pengguna. • Pengguna membatalkan kueri mereka sendiri.		
MEMOTONG TABEL	• Pengguna super. • Pengguna dengan izin TRUNCATE TABLE. • Pemilik meja.		
VAKUM	• Pengguna super. • Pengguna dengan izin VACUUM. • Pemilik meja. • Pemilik database kepada siapa tabel dibagikan.		

Perintah	Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan perintah		
ABAIKAN RLS	• Pengguna super. • Pengguna dalam `sys:secadmin` peran.		
JELASKAN RLS	• Pengguna super. • Pengguna dalam `sys:secadmin` peran.		
JELASKAN MASKING	• Pengguna super. • Pengguna dalam `sys:secadmin` peran.		

Izin objek database

Terlepas dari izin sistem, Amazon Redshift menyertakan izin objek database yang menentukan opsi akses. Ini termasuk opsi seperti kemampuan untuk membaca data dalam tabel dan tampilan, menulis data, membuat tabel, dan melepas tabel. Untuk informasi selengkapnya, lihat HIBAH.

Dengan menggunakan RBAC, Anda dapat menetapkan izin objek database ke peran, mirip dengan bagaimana Anda dapat dengan izin sistem. Kemudian Anda dapat menetapkan peran kepada pengguna, mengotorisasi pengguna dengan izin sistem, dan mengotorisasi pengguna dengan izin database.

UBAH HAK ISTIMEWA STANDAR untuk RBAC

Gunakan pernyataan ALTER DEFAULT PRIVILEGES untuk menentukan kumpulan izin akses default yang akan diterapkan ke objek yang dibuat di masa mendatang oleh pengguna yang ditentukan. Secara default, pengguna hanya dapat mengubah izin akses default mereka sendiri. Dengan RBAC,

Anda dapat mengatur izin akses default untuk peran. Untuk informasi selengkapnya, lihat UBAH HAK ISTIMEWA DEFAULT perintah.

Dengan RBAC, Anda dapat menetapkan izin objek database ke peran, mirip dengan izin sistem. Kemudian Anda dapat menetapkan peran kepada pengguna, mengotorisasi pengguna dengan izin and/or database sistem.

Pertimbangan untuk penggunaan peran di RBAC

Saat bekerja dengan peran RBAC, pertimbangkan hal berikut:

- Amazon Redshift tidak mengizinkan siklus otorisasi peran. Anda tidak dapat memberikan r1 ke r2 dan kemudian memberikan r2 ke r1.
- RBAC berfungsi untuk objek Amazon Redshift asli dan tabel Amazon Redshift Spectrum.
- Sebagai administrator Amazon Redshift, Anda dapat mengaktifkan RBAC dengan memutakhirkan cluster Anda ke patch pemeliharaan terbaru untuk memulai.
- Hanya pengguna super dan pengguna dengan izin sistem CREATE ROLE yang dapat membuat peran.
- Hanya pengguna super dan administrator peran yang dapat memodifikasi atau menghapus peran.
- Nama peran tidak bisa sama dengan nama pengguna.
- Nama peran tidak dapat berisi karakter yang tidak valid, seperti “:/\ n.”
- Nama peran tidak boleh menjadi kata yang dicadangkan, seperti PUBLIK.
- Nama peran tidak dapat dimulai dengan awalan yang dicadangkan untuk peran default,`sys:`.
- Anda tidak dapat menghapus peran yang memiliki parameter RESTRICT saat diberikan ke peran lain. Pengaturan default adalah RESTRICT. Amazon Redshift menampilkan kesalahan saat Anda mencoba menghapus peran yang telah mewarisi peran lain.
- Pengguna yang tidak memiliki izin admin pada peran tidak dapat memberikan atau mencabut peran.
- RBAC tidak sepenuhnya didukung untuk tabel sistem dan tampilan. Izin RBAC untuk tabel sistem dan tampilan tidak bertahan melalui peningkatan, penurunan, atau perubahan ukuran. Sebaiknya gunakan Peran yang ditentukan sistem Amazon Redshift untuk mengelola tabel sistem dan melihat izin. Untuk informasi lebih lanjut tentang tabel sistem, bukaTabel sistem dan referensi tampilan.

Mengelola peran di RBAC

Untuk melakukan tindakan berikut, gunakan perintah berikut:

- Untuk membuat peran, gunakan CREATE ROLE perintah.
- Untuk mengganti nama peran atau mengubah pemilik peran, gunakan MENGUBAH PERAN perintah.
- Untuk menghapus peran, gunakan DROP ROLE perintah.
- Untuk memberikan peran kepada pengguna, gunakan HIBAH perintah.
- Untuk mencabut peran dari pengguna, gunakan MENCABUT perintah.
- Untuk memberikan izin sistem ke peran, gunakan HIBAH perintah.
- Untuk mencabut izin sistem dari peran, gunakan MENCABUT perintah.

Untuk melihat daftar peran dalam cluster atau kelompok kerja Anda, lihatSVV_ROLE.

Tutorial: Membuat peran dan menanyakan dengan RBAC

Dengan RBAC, Anda dapat membuat peran dengan izin untuk menjalankan perintah yang dulunya memerlukan izin superuser. Pengguna dapat menjalankan perintah ini, selama mereka diotorisasi dengan peran yang menyertakan izin ini.

Dalam tutorial ini, Anda menggunakan kontrol akses berbasis peran (RBAC) untuk mengelola izin dalam database yang Anda buat. Anda kemudian terhubung ke database dan menanyakan database dari dua peran yang berbeda untuk menguji fungsionalitas RBAC.

Dua peran yang Anda buat dan gunakan untuk menanyakan database adalah `sales_ro` dan`sales_rw`. Anda membuat `sales_ro` peran dan data kueri sebagai pengguna dengan `sales_ro` peran tersebut. Pengguna `sales_ro` hanya dapat menggunakan perintah SELECT tetapi tidak dapat menggunakan perintah UPDATE. Kemudian, Anda membuat `sales_rw` peran dan kueri data sebagai pengguna dengan `sales_rw` peran tersebut. `sales_rw`Pengguna dapat menggunakan perintah SELECT dan perintah UPDATE.

Selain itu, Anda dapat membuat peran untuk membatasi akses ke perintah tertentu, dan menetapkan peran untuk pengguna super atau pengguna.

Tugas

- Prasyarat

- Langkah 1: Buat pengguna administrator
- Langkah 2: Siapkan skema
- Langkah 3: Buat pengguna read-only
- Langkah 4: Kueri data sebagai pengguna read-only
- Langkah 5: Buat pengguna baca-tulis
- Langkah 6: Kueri data sebagai pengguna dengan peran read-only yang diwariskan
- Langkah 7: Berikan pembaruan dan masukkan izin ke peran baca-tulis
- Langkah 8: Kueri data sebagai pengguna baca-tulis
- Langkah 9: Analisis dan vakum tabel dalam database sebagai pengguna administrator
- Langkah 10: Memotong tabel sebagai pengguna baca-tulis
- Fungsi sistem untuk RBAC (opsional)
- Tampilan sistem untuk RBAC (opsional)
- Gunakan keamanan tingkat baris dengan RBAC (opsional)

Prasyarat

- Buat cluster Amazon Redshift atau kelompok kerja tanpa server yang dimuat dengan database sampel TICKIT. Untuk membuat workgroup tanpa server, lihat Mem ulai dengan gudang data Redshift Serverless. Untuk membuat cluster, lihat Membuat cluster Amazon Redshift sampel. Untuk informasi lebih lanjut tentang database sampel TICKIT, lihatDatabase sampel.
- Memiliki akses ke pengguna dengan izin superuser atau administrator peran. Hanya pengguna super atau administrator peran yang dapat memberikan atau mencabut peran. Untuk informasi selengkapnya tentang izin yang diperlukan untuk RBAC, lihatIzin sistem untuk RBAC.
- TinjauPertimbangan untuk penggunaan peran di RBAC.

Langkah 1: Buat pengguna administrator

Untuk mengatur tutorial ini, Anda membuat peran admin database dan melampirkannya ke pengguna administrator database dalam langkah ini. Anda harus membuat administrator database sebagai superuser atau administrator peran.

Jalankan semua kueri di editor kueri Amazon Redshift v2.

1. Untuk membuat peran administrator db_admin, gunakan contoh berikut.

```
CREATE ROLE db_admin;
```

2. Untuk membuat pengguna database bernama dbadmin, gunakan contoh berikut.

```
CREATE USER dbadmin PASSWORD 'Test12345';
```

3. Untuk memberikan peran yang ditentukan sistem bernama sys:dba ke peran db_admin, gunakan contoh berikut. Ketika diberikan peran sys:dba, pengguna dbadmin dapat membuat skema dan tabel. Untuk informasi selengkapnya, lihat Peran yang ditentukan sistem Amazon Redshift.

Langkah 2: Siapkan skema

Pada langkah ini, Anda terhubung ke database Anda sebagai administrator database. Kemudian, Anda membuat dua skema dan menambahkan data ke dalamnya.

1. Hubungkan ke database dev sebagai pengguna dbadmin menggunakan editor kueri v2. Untuk informasi selengkapnya tentang menghubungkan ke database, lihat Bek erja dengan editor kueri v2.
2. Untuk membuat skema database penjualan dan pemasaran, gunakan contoh berikut.

```
CREATE SCHEMA sales;
CREATE SCHEMA marketing;
```

3. Untuk membuat dan menyisipkan nilai ke dalam tabel dalam skema penjualan, gunakan contoh berikut.

```
CREATE TABLE sales.cat(
catid smallint,
catgroup varchar(10),
catname varchar(10),
catdesc varchar(50)
);
INSERT INTO sales.cat(SELECT * FROM category);

CREATE TABLE sales.dates(
dateid smallint,
caldate date,
day char(3),
week smallint,
month char(5),
```

```
qtr char(5),
year smallint,
holiday boolean
);
INSERT INTO sales.dates(SELECT * FROM date);

CREATE TABLE sales.events(
eventid integer,
venueid smallint,
catid smallint,
dateid smallint,
eventname varchar(200),
starttime timestamp
);
INSERT INTO sales.events(SELECT * FROM event);

 CREATE TABLE sales.sale(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp
);
INSERT INTO sales.sale(SELECT * FROM sales);
```

4. Untuk membuat dan memasukkan nilai ke dalam tabel dalam skema pemasaran, gunakan contoh berikut.

```
CREATE TABLE marketing.cat(
catid smallint,
catgroup varchar(10),
catname varchar(10),
catdesc varchar(50)
);
INSERT INTO marketing.cat(SELECT * FROM category);

CREATE TABLE marketing.dates(
dateid smallint,
```

```
caldate date,
day char(3),
week smallint,
month char(5),
qtr char(5),
year smallint,
holiday boolean
);
INSERT INTO marketing.dates(SELECT * FROM date);

CREATE TABLE marketing.events(
eventid integer,
venueid smallint,
catid smallint,
dateid smallint,
eventname varchar(200),
starttime timestamp
);
INSERT INTO marketing.events(SELECT * FROM event);

CREATE TABLE marketing.sale(
marketingid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp
);
INSERT INTO marketing.sale(SELECT * FROM marketing);
```

Langkah 3: Buat pengguna read-only

Pada langkah ini, Anda membuat peran read-only dan pengguna analis penjualan untuk peran read-only. Analis penjualan hanya memerlukan akses read-only ke tabel dalam skema penjualan untuk menyelesaikan tugas yang ditugaskan untuk menemukan peristiwa yang menghasilkan komisi terbesar.

1. Hubungkan ke database sebagai pengguna dbadmin.

2. Untuk membuat peran sales_ro, gunakan contoh berikut.

```
CREATE ROLE sales_ro;
```

3. Untuk membuat pengguna salesAnalyst, gunakan contoh berikut.

```
CREATE USER salesanalyst PASSWORD 'Test12345';
```

4. Untuk memberikan penggunaan peran sales_ro dan memilih akses ke objek skema penjualan, gunakan contoh berikut.

```
GRANT USAGE ON SCHEMA sales TO ROLE sales_ro;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO ROLE sales_ro;
```

5. Untuk memberikan peran sales_ro kepada pengguna salesanalis, gunakan contoh berikut.

```
GRANT ROLE sales_ro TO salesanalyst;
```

Langkah 4: Kueri data sebagai pengguna read-only

Pada langkah ini, pengguna salesAnalyst menanyakan data dari skema penjualan. Kemudian, pengguna salesAnalyst mencoba memperbarui tabel dan membaca tabel dalam skema pemasaran.

1. Hubungkan ke database sebagai pengguna analis penjualan.
2. Untuk menemukan 10 penjualan dengan komisi tertinggi, gunakan contoh berikut.

```
SET SEARCH_PATH TO sales;
SELECT DISTINCT events.dateid, sale.commission, cat.catname
FROM sale, events, dates, cat
WHERE events.dateid=dates.dateid AND events.dateid=sale.dateid AND events.catid =
 cat.catid
ORDER BY 2 DESC LIMIT 10;

+--------+------------+----------+
| dateid | commission | catname  |
+--------+------------+----------+
|   1880 |     1893.6 | Pop      |
|   1880 |     1893.6 | Opera    |
|   1880 |     1893.6 | Plays    |
|   1880 |     1893.6 | Musicals |
```

```
|   1861 |       1500 | Plays    |
|   2003 |       1500 | Pop      |
|   1861 |       1500 | Opera    |
|   2003 |       1500 | Plays    |
|   1861 |       1500 | Musicals |
|   1861 |       1500 | Pop      |
+--------+------------+----------+
```

3. Untuk memilih 10 peristiwa dari tabel peristiwa dalam skema penjualan, gunakan contoh berikut.

```
SELECT * FROM sales.events LIMIT 10;

+---------+---------+-------+--------+--------------------+---------------------+
| eventid | venueid | catid | dateid |     eventname      |      starttime      |
+---------+---------+-------+--------+--------------------+---------------------+
|    4836 |      73 |     9 |   1871 | Soulfest           | 2008-02-14 19:30:00 |
|    5739 |      41 |     9 |   1871 | Fab Faux           | 2008-02-14 19:30:00 |
|     627 |     229 |     6 |   1872 | High Society       | 2008-02-15 14:00:00 |
|    2563 |     246 |     7 |   1872 | Hamlet             | 2008-02-15 20:00:00 |
|    7703 |      78 |     9 |   1872 | Feist              | 2008-02-15 14:00:00 |
|    7903 |      90 |     9 |   1872 | Little Big Town    | 2008-02-15 19:30:00 |
|    7925 |     101 |     9 |   1872 | Spoon              | 2008-02-15 19:00:00 |
|    8113 |      17 |     9 |   1872 | Santana            | 2008-02-15 15:00:00 |
|     463 |     303 |     8 |   1873 | Tristan und Isolde | 2008-02-16 19:00:00 |
|     613 |     236 |     6 |   1873 | Pal Joey           | 2008-02-16 15:00:00 |
+---------+---------+-------+--------+--------------------+---------------------+
```

4. Untuk mencoba memperbarui nama acara untuk eventid 1, jalankan contoh berikut. Contoh ini akan mengakibatkan kesalahan izin ditolak karena pengguna salesAnalyst hanya memiliki izin SELECT pada tabel peristiwa dalam skema penjualan. Untuk memperbarui tabel peristiwa, Anda harus memberikan izin peran sales_ro untuk UPDATE. Untuk informasi selengkapnya tentang pemberian izin untuk memperbarui tabel, lihat parameter UPDATE untukHIBAH. Untuk informasi selengkapnya tentang perintah UPDATE, lihatUPDATE.

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;

ERROR: permission denied for relation events
```

5. Untuk mencoba memilih semua dari tabel peristiwa dalam skema pemasaran, gunakan contoh berikut. Contoh ini akan mengakibatkan kesalahan izin ditolak karena pengguna salesAnalyst

hanya memiliki izin SELECT untuk tabel peristiwa dalam skema penjualan. Untuk memilih data dari tabel peristiwa dalam skema pemasaran, Anda harus memberikan izin SELECT peran sales_ro pada tabel peristiwa dalam skema pemasaran.

```
SELECT * FROM marketing.events;

               ERROR: permission denied for schema marketing
```

Langkah 5: Buat pengguna baca-tulis

Pada langkah ini, insinyur penjualan yang bertanggung jawab untuk membangun pipeline ekstrak, transformasi, dan memuat (ETL) untuk pemrosesan data dalam skema penjualan akan diberikan akses read-only, tetapi nantinya akan diberikan akses baca dan tulis untuk melakukan tugas mereka.

1. Hubungkan ke database sebagai pengguna dbadmin.
2. Untuk membuat peran sales_rw dalam skema penjualan, gunakan contoh berikut.

```
CREATE ROLE sales_rw;
```

3. Untuk membuat pengguna salesengineer, gunakan contoh berikut.

```
CREATE USER salesengineer PASSWORD 'Test12345';
```

4. Untuk memberikan penggunaan peran sales_rw dan memilih akses ke objek skema penjualan dengan menetapkan peran sales_ro padanya, gunakan contoh berikut. Untuk informasi selengkapnya tentang bagaimana peran mewarisi izin di Amazon Redshift, lihatHirarki peran.

```
GRANT ROLE sales_ro TO ROLE sales_rw;
```

5. Untuk menetapkan peran sales_rw ke pengguna salesengineer, gunakan contoh berikut.

```
GRANT ROLE sales_rw TO salesengineer;
```

Langkah 6: Kueri data sebagai pengguna dengan peran read-only yang diwariskan

Pada langkah ini, pengguna salesengineer mencoba memperbarui tabel peristiwa sebelum mereka diberikan izin baca.

1. Hubungkan ke database sebagai pengguna salesengineer.
2. Pengguna salesengineer dapat berhasil membaca data dari tabel peristiwa skema penjualan. Untuk memilih acara dengan eventid 1 dari tabel peristiwa dalam skema penjualan, gunakan contoh berikut.

```
SELECT * FROM sales.events where eventid=1;

+---------+---------+-------+--------+-----------------+---------------------+
| eventid | venueid | catid | dateid |    eventname    |      starttime      |
+---------+---------+-------+--------+-----------------+---------------------+
|       1 |     305 |     8 |   1851 | Gotterdammerung | 2008-01-25 14:30:00 |
+---------+---------+-------+--------+-----------------+---------------------+
```

3. Untuk mencoba memilih semua dari tabel peristiwa dalam skema pemasaran, gunakan contoh berikut. Pengguna salesengineer tidak memiliki izin untuk tabel dalam skema pemasaran, jadi kueri ini akan menghasilkan kesalahan izin ditolak. Untuk memilih data dari tabel peristiwa dalam skema pemasaran, Anda harus memberikan izin SELECT peran sales_rw pada tabel peristiwa dalam skema pemasaran.

```
SELECT * FROM marketing.events;

ERROR: permission denied for schema marketing
```

4. Untuk mencoba memperbarui nama acara untuk eventid 1, jalankan contoh berikut. Contoh ini akan mengakibatkan kesalahan izin ditolak karena pengguna salesengineer hanya memiliki izin pilih pada tabel peristiwa dalam skema penjualan. Untuk memperbarui tabel peristiwa, Anda harus memberikan izin peran sales_rw untuk UPDATE.

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;

ERROR: permission denied for relation events
```

Langkah 7: Berikan pembaruan dan masukkan izin ke peran baca-tulis

Pada langkah ini, Anda memberikan pembaruan dan menyisipkan izin ke peran sales_rw.

1. Hubungkan ke database sebagai pengguna dbadmin.

2. Untuk memberikan izin UPDATE, INSERT, dan DELETE ke peran sales_rw, gunakan contoh berikut.

```
GRANT UPDATE, INSERT, ON ALL TABLES IN SCHEMA sales TO role sales_rw;
```

Langkah 8: Kueri data sebagai pengguna baca-tulis

Pada langkah ini, salesengineer berhasil memperbarui tabel setelah peran mereka diberikan izin menyisipkan dan memperbarui. Selanjutnya, salesengineer mencoba menganalisis dan menyedot debu tabel peristiwa tetapi gagal melakukannya.

1. Hubungkan ke database sebagai pengguna salesengineer.
2. Untuk memperbarui nama acara untuk eventid 1, jalankan contoh berikut.

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;
```

3. Untuk melihat perubahan yang dibuat dalam kueri sebelumnya, gunakan contoh berikut untuk memilih acara dengan eventid 1 dari tabel peristiwa dalam skema penjualan.

```
SELECT * FROM sales.events WHERE eventid=1;

+---------+---------+-------+--------+--------------+---------------------+
| eventid | venueid | catid | dateid |   eventname  |      starttime      |
+---------+---------+-------+--------+--------------+---------------------+
|       1 |     305 |     8 |   1851 | Comment event | 2008-01-25 14:30:00 |
+---------+---------+-------+--------+--------------+---------------------+
```

4. Untuk menganalisis tabel peristiwa yang diperbarui dalam skema penjualan, gunakan contoh berikut. Contoh ini akan mengakibatkan kesalahan izin ditolak karena pengguna salesengineer tidak memiliki izin yang diperlukan dan bukan pemilik tabel peristiwa dalam skema penjualan. Untuk menganalisis tabel peristiwa, Anda harus memberikan izin peran sales_rw untuk ANALYZE menggunakan perintah GRANT. Untuk informasi selengkapnya tentang perintah ANALYZE, lihatMENGANALISA.

```
ANALYZE sales.events;

               ERROR: skipping "events" --- only table or database owner can analyze
```

5. Untuk menyedot debu tabel peristiwa yang diperbarui, gunakan contoh berikut. Contoh ini akan mengakibatkan kesalahan izin ditolak karena pengguna salesengineer tidak memiliki izin yang diperlukan dan bukan pemilik tabel peristiwa dalam skema penjualan. Untuk menyedot debu tabel peristiwa, Anda harus memberikan izin peran sales_rw ke VACUUM menggunakan perintah GRANT. Untuk informasi selengkapnya tentang perintah VACUUM, lihatVAKUM.

```
VACUUM sales.events;

ERROR: skipping "events" --- only table or database owner can vacuum it
```

Langkah 9: Analisis dan vakum tabel dalam database sebagai pengguna administrator

Pada langkah ini, pengguna dbadmin menganalisis dan menyedot semua tabel. Pengguna memiliki izin administrator pada database ini, sehingga mereka dapat menjalankan perintah ini.

1. Hubungkan ke database sebagai pengguna dbadmin.
2. Untuk menganalisis tabel peristiwa dalam skema penjualan, gunakan contoh berikut.

```
ANALYZE sales.events;
```

3. Untuk menyedot tabel peristiwa dalam skema penjualan, gunakan contoh berikut.

```
VACUUM sales.events;
```

4. Untuk menganalisis tabel peristiwa dalam skema pemasaran, gunakan contoh berikut.

```
ANALYZE marketing.events;
```

5. Untuk menyedot tabel peristiwa dalam skema pemasaran, gunakan contoh berikut.

```
VACUUM marketing.events;
```

Langkah 10: Memotong tabel sebagai pengguna baca-tulis

Pada langkah ini, pengguna salesengineer mencoba memotong tabel peristiwa dalam skema penjualan, tetapi hanya berhasil jika diberikan izin potong oleh pengguna dbadmin.

1. Hubungkan ke database sebagai pengguna salesengineer.

2. Untuk mencoba menghapus semua baris dari tabel peristiwa dalam skema penjualan, gunakan contoh berikut. Contoh ini akan mengakibatkan kesalahan karena pengguna salesengineer tidak memiliki izin yang diperlukan dan bukan pemilik tabel peristiwa dalam skema penjualan. Untuk memotong tabel peristiwa, Anda harus memberikan izin peran sales_rw untuk TRUNCATE menggunakan perintah GRANT. Untuk informasi selengkapnya tentang perintah TRUNCATE, lihatMEMOTONG.

```
TRUNCATE sales.events;

ERROR: must be owner of relation events
```

3. Hubungkan ke database sebagai pengguna dbadmin.
4. Untuk memberikan hak istimewa tabel terpotong ke peran sales_rw, gunakan contoh berikut.

```
GRANT TRUNCATE TABLE TO role sales_rw;
```

5. Hubungkan ke database sebagai pengguna salesengineer menggunakan editor kueri v2.
6. Untuk membaca 10 peristiwa pertama dari tabel peristiwa dalam skema penjualan, gunakan contoh berikut.

```
SELECT * FROM sales.events ORDER BY eventid LIMIT 10;

+---------+---------+-------+--------+----------------------------
+---------------------+
| eventid | venueid | catid | dateid |          eventname          |      starttime
    |
+---------+---------+-------+--------+----------------------------
+---------------------+
|       1 |     305 |     8 |   1851 | Comment event               | 2008-01-25
 14:30:00 |
|       2 |     306 |     8 |   2114 | Boris Godunov               | 2008-10-15
 20:00:00 |
|       3 |     302 |     8 |   1935 | Salome                      | 2008-04-19
 14:30:00 |
|       4 |     309 |     8 |   2090 | La Cenerentola (Cinderella) | 2008-09-21
 14:30:00 |
|       5 |     302 |     8 |   1982 | Il Trovatore                | 2008-06-05
 19:00:00 |
|       6 |     308 |     8 |   2109 | L Elisir d Amore            | 2008-10-10
 19:30:00 |
```

```
|       7 |     309 |     8 |   1891 | Doctor Atomic               | 2008-03-06
 14:00:00 |
|       8 |     302 |     8 |   1832 | The Magic Flute             | 2008-01-06
 20:00:00 |
|       9 |     308 |     8 |   2087 | The Fly                     | 2008-09-18
 19:30:00 |
|      10 |     305 |     8 |   2079 | Rigoletto                   | 2008-09-10
 15:00:00 |
+---------+---------+-------+--------+-----------------------------
+---------------------+
```

7. Untuk memotong tabel peristiwa dalam skema penjualan, gunakan contoh berikut.

```
TRUNCATE sales.events;
```

8. Untuk membaca data dari tabel peristiwa yang diperbarui dalam skema penjualan, gunakan contoh berikut.

```
SELECT * FROM sales.events ORDER BY eventid LIMIT 10;

+---------+---------+-------+--------+-----------------------------
+---------------------+
| eventid | venueid | catid | dateid |          eventname          |      starttime
    |
+---------+---------+-------+--------+-----------------------------
+---------------------+
```

Buat peran baca-saja dan baca-tulis untuk skema pemasaran (opsional)

Pada langkah ini, Anda membuat peran baca-saja dan baca-tulis untuk skema pemasaran.

1. Hubungkan ke database sebagai pengguna dbadmin.
2. Untuk membuat peran baca-saja dan baca-tulis untuk skema pemasaran, gunakan contoh berikut.

```
CREATE ROLE marketing_ro;

CREATE ROLE marketing_rw;

GRANT USAGE ON SCHEMA marketing TO ROLE marketing_ro, ROLE marketing_rw;

GRANT SELECT ON ALL TABLES IN SCHEMA marketing TO ROLE marketing_ro;
```

```
GRANT ROLE marketing_ro TO ROLE marketing_rw;

GRANT INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA marketing TO ROLE marketing_rw;

CREATE USER marketinganalyst PASSWORD 'Test12345';

CREATE USER marketingengineer PASSWORD 'Test12345';

GRANT ROLE marketing_ro TO marketinganalyst;

GRANT ROLE marketing_rw TO marketingengineer;
```

Fungsi sistem untuk RBAC (opsional)

Amazon Redshift memiliki dua fungsi untuk memberikan informasi sistem tentang keanggotaan pengguna dan keanggotaan peran dalam grup atau peran tambahan: role_is_member_of dan user_is_member_of. Fungsi-fungsi ini tersedia untuk pengguna super dan pengguna biasa. Pengguna super dapat memeriksa semua keanggotaan peran. Pengguna biasa hanya dapat memeriksa keanggotaan untuk peran yang telah diberikan akses kepada mereka.

Untuk menggunakan fungsi role_is_member_of

1. Hubungkan ke database sebagai pengguna salesengineer.
2. Untuk memeriksa apakah peran sales_rw adalah anggota peran sales_ro, gunakan contoh berikut.

```
SELECT role_is_member_of('sales_rw', 'sales_ro');

+------------------+
| role_is_member_of |
+------------------+
| true             |
+------------------+
```

3. Untuk memeriksa apakah peran sales_ro adalah anggota dari peran sales_rw, gunakan contoh berikut.

```
SELECT role_is_member_of('sales_ro', 'sales_rw');
```

```
+-------------------+
| role_is_member_of |
+-------------------+
| false             |
+-------------------+
```

Untuk menggunakan fungsi user_is_member_of

1. Hubungkan ke database sebagai pengguna salesengineer.
2. Contoh berikut mencoba memeriksa keanggotaan pengguna untuk pengguna analis penjualan. Kueri ini menghasilkan kesalahan karena salesengineer tidak memiliki akses ke salesAnalyst. Untuk menjalankan perintah ini dengan sukses, sambungkan ke database sebagai pengguna salesAnalyst dan gunakan contoh.

```
SELECT user_is_member_of('salesanalyst', 'sales_ro');

ERROR
```

3. Hubungkan ke database sebagai superuser.
4. Untuk memeriksa keanggotaan pengguna salesAnalyst saat terhubung sebagai superuser, gunakan contoh berikut.

```
SELECT user_is_member_of('salesanalyst', 'sales_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| true              |
+-------------------+
```

5. Hubungkan ke database sebagai pengguna dbadmin.
6. Untuk memeriksa keanggotaan pengguna salesengineer, gunakan contoh berikut.

```
SELECT user_is_member_of('salesengineer', 'sales_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| true              |
+-------------------+
```

```
SELECT user_is_member_of('salesengineer', 'marketing_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| false             |
+-------------------+

SELECT user_is_member_of('marketinganalyst', 'sales_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| false             |
+-------------------+
```

Tampilan sistem untuk RBAC (opsional)

Untuk melihat peran, penetapan peran kepada pengguna, hierarki peran, dan hak istimewa untuk objek database melalui peran, gunakan tampilan sistem untuk Amazon Redshift. Tampilan ini tersedia untuk pengguna super dan pengguna biasa. Pengguna super dapat memeriksa semua detail peran. Pengguna biasa hanya dapat memeriksa detail untuk peran yang telah diberikan akses kepada mereka.

1. Untuk melihat daftar pengguna yang secara eksplisit diberikan peran dalam cluster, gunakan contoh berikut.

   ```
   SELECT * FROM svv_user_grants;
   ```

2. Untuk melihat daftar peran yang secara eksplisit diberikan peran dalam cluster, gunakan contoh berikut.

   ```
   SELECT * FROM svv_role_grants;
   ```

Untuk daftar lengkap tampilan sistem, lihatTampilan metadata SVV.

Gunakan keamanan tingkat baris dengan RBAC (opsional)

Untuk memiliki kontrol akses terperinci atas data sensitif Anda, gunakan keamanan tingkat baris (RLS). Untuk informasi lebih lanjut tentang RLS, lihatRow-level keamanan.

Di bagian ini, Anda membuat kebijakan RLS yang memberikan izin `salesengineer` pengguna untuk hanya melihat baris dalam `cat` tabel yang memiliki `catdesc` nilai Major League Baseball. Anda kemudian menanyakan database sebagai `salesengineer` pengguna.

1. Hubungkan ke database sebagai `salesengineer` pengguna.
2. Untuk melihat 5 entri pertama dalam `cat` tabel, gunakan contoh berikut.

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
|     1 | Sports   | MLB     | Major League Baseball           |
|     2 | Sports   | NHL     | National Hockey League          |
|     3 | Sports   | NFL     | National Football League        |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer             |
+-------+----------+---------+---------------------------------+
```

3. Hubungkan ke database sebagai `dbadmin` pengguna.
4. Untuk membuat kebijakan RLS untuk `catdesc` kolom dalam `cat` tabel, gunakan contoh berikut.

```
CREATE RLS POLICY policy_mlb_engineer
WITH (catdesc VARCHAR(50))
USING (catdesc = 'Major League Baseball');
```

5. Untuk melampirkan kebijakan RLS ke `sales_rw` peran, gunakan contoh berikut.

```
ATTACH RLS POLICY policy_mlb_engineer ON sales.cat TO ROLE sales_rw;
```

6. Untuk mengubah tabel untuk mengaktifkan RLS, gunakan contoh berikut.

```
ALTER TABLE sales.cat ROW LEVEL SECURITY ON;
```

7. Hubungkan ke database sebagai `salesengineer` pengguna.
8. Untuk mencoba melihat 5 entri pertama dalam `cat` tabel, gunakan contoh berikut. Perhatikan bahwa hanya entri yang hanya muncul ketika `catdesc` kolom ada`Major League Baseball`.

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+----------------------+
| catid | catgroup | catname |        catdesc       |
+-------+----------+---------+----------------------+
|     1 | Sports   | MLB     | Major League Baseball |
+-------+----------+---------+----------------------+
```

9. Hubungkan ke database sebagai `salesanalyst` pengguna.
10. Untuk mencoba melihat 5 entri pertama dalam `cat` tabel, gunakan contoh berikut. Perhatikan bahwa tidak ada entri yang muncul karena kebijakan tolak semua default diterapkan.

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+----------------------+
| catid | catgroup | catname |        catdesc       |
+-------+----------+---------+----------------------+
```

11. Hubungkan ke database sebagai `dbadmin` pengguna.
12. Untuk memberikan izin IGNORE RLS ke `sales_ro` peran, gunakan contoh berikut. Ini memberi `salesanalyst` pengguna izin untuk mengabaikan kebijakan RLS karena mereka adalah anggota `sales_ro` peran.

```
GRANT IGNORE RLS TO ROLE sales_ro;
```

13. Hubungkan ke database sebagai `salesanalyst` pengguna.
14. Untuk melihat 5 entri pertama dalam `cat` tabel, gunakan contoh berikut.

```
SELECT *
FROM sales.cat
```

```
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+--------------------------------+
| catid | catgroup | catname |             catdesc            |
+-------+----------+---------+--------------------------------+
|     1 | Sports   | MLB     | Major League Baseball          |
|     2 | Sports   | NHL     | National Hockey League         |
|     3 | Sports   | NFL     | National Football League       |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer            |
+-------+----------+---------+--------------------------------+
```

15. Hubungkan ke database sebagai `dbadmin` pengguna.
16. Untuk mencabut izin IGNORE RLS dari `sales_ro` peran, gunakan contoh berikut.

```
REVOKE IGNORE RLS FROM ROLE sales_ro;
```

17. Hubungkan ke database sebagai `salesanalyst` pengguna.
18. Untuk mencoba melihat 5 entri pertama dalam `cat` tabel, gunakan contoh berikut. Perhatikan bahwa tidak ada entri yang muncul karena kebijakan tolak semua default diterapkan.

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+----------------------+
| catid | catgroup | catname |        catdesc       |
+-------+----------+---------+----------------------+
```

19. Hubungkan ke database sebagai `dbadmin` pengguna.
20. Untuk melepaskan kebijakan RLS dari `cat` tabel, gunakan contoh berikut.

```
DETACH RLS POLICY policy_mlb_engineer ON cat FROM ROLE sales_rw;
```

21. Hubungkan ke database sebagai `salesanalyst` pengguna.
22. Untuk mencoba melihat 5 entri pertama dalam `cat` tabel, gunakan contoh berikut. Perhatikan bahwa tidak ada entri yang muncul karena kebijakan tolak semua default diterapkan.

```
SELECT *
```

```
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
|     1 | Sports   | MLB     | Major League Baseball           |
|     2 | Sports   | NHL     | National Hockey League          |
|     3 | Sports   | NFL     | National Football League        |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer             |
+-------+----------+---------+---------------------------------+
```

23. Hubungkan ke database sebagai `dbadmin` pengguna.
24. Untuk menghapus kebijakan RLS, gunakan contoh berikut.

```
DROP RLS POLICY policy_mlb_engineer;
```

25. Untuk menghapus RLS, gunakan contoh berikut.

```
ALTER TABLE cat ROW LEVEL SECURITY OFF;
```

Topik terkait

Untuk informasi selengkapnya tentang RBAC, lihat dokumentasi berikut:

- Hirarki peran
- Penetapan peran
- Izin objek database
- UBAH HAK ISTIMEWA STANDAR untuk RBAC

Row-level keamanan

Menggunakan keamanan tingkat baris (RLS) di Amazon Redshift, Anda dapat memiliki kontrol akses terperinci atas data sensitif Anda. Anda dapat memutuskan pengguna atau peran mana yang dapat mengakses catatan data tertentu dalam skema atau tabel, berdasarkan kebijakan keamanan yang ditentukan pada tingkat objek database. Selain keamanan tingkat kolom, di mana Anda dapat

memberikan izin kepada pengguna ke subset kolom, gunakan kebijakan RLS untuk membatasi akses lebih lanjut ke baris tertentu dari kolom yang terlihat. Untuk informasi selengkapnya tentang keamanan tingkat kolom, lihat. Catatan penggunaan untuk kontrol akses tingkat kolom

Saat menerapkan kebijakan RLS pada tabel, Anda dapat membatasi kumpulan hasil yang dikembalikan saat pengguna menjalankan kueri.

Saat membuat kebijakan RLS, Anda dapat menentukan ekspresi yang menentukan apakah Amazon Redshift mengembalikan baris yang ada dalam tabel dalam kueri. Dengan membuat kebijakan RLS untuk membatasi akses, Anda tidak perlu menambahkan atau mengeksternalisasi kondisi tambahan dalam kueri Anda.

Saat membuat kebijakan RLS, sebaiknya buat kebijakan sederhana dan menghindari pernyataan rumit dalam kebijakan. Saat menentukan kebijakan RLS, jangan gunakan gabungan tabel berlebihan dalam definisi kebijakan yang didasarkan pada kebijakan.

Saat kebijakan mengacu pada tabel pencarian, Amazon Redshift memindai tabel tambahan, selain tabel di mana kebijakan tersebut ada. Akan ada perbedaan kinerja antara kueri yang sama untuk pengguna dengan kebijakan RLS terlampir, dan pengguna tanpa kebijakan yang dilampirkan.

Menggunakan kebijakan RLS dalam pernyataan SQL

Saat menggunakan kebijakan RLS dalam pernyataan SQL, Amazon Redshift menerapkan aturan berikut:

- Amazon Redshift menerapkan kebijakan RLS ke pernyataan SELECT, UPDATE, dan DELETE secara default.
- Untuk SELECT dan UNLOAD, Amazon Redshift memfilter baris sesuai dengan kebijakan yang Anda tentukan.
- Untuk UPDATE, Amazon Redshift hanya memperbarui baris yang terlihat oleh Anda. Jika kebijakan membatasi subset baris dalam tabel, Anda tidak dapat memperbaruinya.
- Untuk DELETE, Anda hanya dapat menghapus baris yang terlihat oleh Anda. Jika kebijakan membatasi subset baris dalam tabel, Anda tidak dapat menghapusnya. Untuk TRUNCATE, Anda masih dapat memotong tabel.
- Untuk CREATE TABLE LIKE, tabel yang dibuat dengan opsi LIKE tidak akan mewarisi pengaturan izin dari tabel sumber. Demikian pula, tabel target tidak akan mewarisi kebijakan RLS dari tabel sumber.

Menggabungkan beberapa kebijakan per pengguna

RLS di Amazon Redshift mendukung melampirkan beberapa kebijakan per pengguna dan objek. Jika ada beberapa kebijakan yang ditentukan untuk pengguna, Amazon Redshift menerapkan semua kebijakan dengan sintaks AND atau OR tergantung pada setelan RLS CONJUNCTION TYPE untuk tabel. Untuk informasi selengkapnya tentang jenis konjungsi, lihatALTER TABLE.

Beberapa kebijakan pada tabel dapat dikaitkan dengan Anda. Beberapa kebijakan dilampirkan langsung kepada Anda, atau Anda termasuk dalam beberapa peran, dan peran tersebut memiliki kebijakan berbeda yang melekat padanya.

Ketika beberapa kebijakan harus membatasi akses baris dalam relasi tertentu, Anda dapat mengatur RLS CONJUNCTION TYPE dari relasi ke AND. Pertimbangkan contoh berikut. Alice hanya dapat melihat acara Olahraga yang memiliki “catname” NBA sesuai kebijakan yang ditentukan.

```
-- Create an analyst role and grant it to a user named Alice.
CREATE ROLE analyst;
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
GRANT ROLE analyst TO alice;

-- Create an RLS policy that only lets the user see sports.
CREATE RLS POLICY policy_sports
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Sports');

-- Create an RLS policy that only lets the user see NBA.
CREATE RLS POLICY policy_nba
WITH (catname VARCHAR(10))
USING (catname = 'NBA');

-- Attach both to the analyst role.
ATTACH RLS POLICY policy_sports ON category TO ROLE analyst;
ATTACH RLS POLICY policy_nba ON category TO ROLE analyst;

-- Activate RLS on the category table with AND CONJUNCTION TYPE.
ALTER TABLE category ROW LEVEL SECURITY ON CONJUNCTION TYPE AND;

-- Change session to Alice.
SET SESSION AUTHORIZATION alice;

-- Select all from the category table.
SELECT catgroup, catname
```

```
FROM category;

 catgroup | catname
---------+---------
 Sports   | NBA
(1 row)
```

Ketika beberapa kebijakan harus mengizinkan pengguna untuk melihat lebih banyak baris dalam relasi tertentu, pengguna dapat mengatur RLS CONJUNCTION TYPE dari relasi ke OR. Pertimbangkan contoh berikut. Alice hanya dapat melihat “Konser” dan “Olahraga” sesuai kebijakan yang ditentukan.

```
-- Create an analyst role and grant it to a user named Alice.
CREATE ROLE analyst;
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
GRANT ROLE analyst TO alice;

-- Create an RLS policy that only lets the user see concerts.
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');

-- Create an RLS policy that only lets the user see sports.
CREATE RLS POLICY policy_sports
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Sports');

-- Attach both to the analyst role.
ATTACH RLS POLICY policy_concerts ON category TO ROLE analyst;
ATTACH RLS POLICY policy_sports ON category TO ROLE analyst;

-- Activate RLS on the category table with OR CONJUNCTION TYPE.
ALTER TABLE category ROW LEVEL SECURITY ON CONJUNCTION TYPE OR;

-- Change session to Alice.
SET SESSION AUTHORIZATION alice;

-- Select all from the category table.
SELECT catgroup, count(*)
FROM category
GROUP BY catgroup ORDER BY catgroup;

 catgroup | count
```

```
---------+-------
 Concerts |  3
 Sports   |  5
(2 rows)
```

Kepemilikan dan manajemen kebijakan RLS

Sebagai superuser, administrator keamanan, atau pengguna yang memiliki peran sys:secadmin, Anda dapat membuat, memodifikasi, melampirkan, dan melepaskan kebijakan RLS. Kebijakan RLS dapat dilampirkan ke tabel, tampilan, tampilan pengikatan terlambat (LBV), dan tampilan terwujud (MV). Pada tingkat objek, Anda dapat mengaktifkan atau menonaktifkan keamanan tingkat baris tanpa mengubah definisi skema untuk tabel.

Untuk memulai dengan keamanan tingkat baris, berikut adalah pernyataan SQL yang dapat Anda gunakan:

- Gunakan pernyataan ALTER TABLE untuk mengaktifkan atau menonaktifkan RLS pada tabel, tampilan, atau tampilan pengikatan terlambat. Untuk informasi selengkapnya, lihat ALTER TABLE.
- Gunakan pernyataan ALTER MATERIALIZED VIEW to statement untuk mengaktifkan atau menonaktifkan RLS pada tampilan terwujud (MV). Untuk informasi selengkapnya, lihat MENGUBAH TAMPILAN YANG TERWUJUD.
- Gunakan pernyataan CREATE RLS POLICY untuk membuat kebijakan keamanan untuk satu atau lebih tabel, dan tentukan satu atau lebih pengguna atau peran dalam kebijakan.

  Untuk informasi selengkapnya, lihat BUAT KEBIJAKAN RLS.
- Gunakan pernyataan ALTER RLS POLICY untuk mengubah kebijakan, seperti mengubah definisi kebijakan. Anda dapat menggunakan kebijakan yang sama untuk beberapa tabel atau tampilan.

  Untuk informasi selengkapnya, lihat MENGUBAH KEBIJAKAN RLS.
- Gunakan pernyataan ATTACH RLS POLICY untuk melampirkan kebijakan ke satu atau lebih relasi, ke satu atau lebih pengguna, atau ke peran.

  Untuk informasi selengkapnya, lihat LAMPIRKAN KEBIJAKAN RLS.
- Gunakan pernyataan DETACH RLS POLICY untuk melepaskan kebijakan dari satu atau lebih relasi, dari satu atau lebih pengguna, atau dari peran.

  Untuk informasi selengkapnya, lihat LEPASKAN KEBIJAKAN RLS.
- Gunakan pernyataan DROP RLS POLICY untuk menghapus kebijakan.

Untuk informasi selengkapnya, lihat KEBIJAKAN DROP RLS.

- Gunakan pernyataan GRANT dan REVOKE untuk secara eksplisit memberikan dan mencabut izin SELECT ke kebijakan RLS yang mereferensikan tabel pencarian. Untuk informasi selengkapnya, lihat HIBAH dan MENCABUT.

Untuk memantau kebijakan yang dibuat, sys:secadmin dapat melihat dan. SVV_RLS_POLICY SVV_RLS_TERLAMPIR_KEBIJAKAN

Untuk daftar RLS-protected relasi, sys:secadmin dapat melihat. SVV_RLS_RELASI

Untuk melacak penerapan kebijakan RLS pada kueri yang merefer RLS-protected ensikan hubungan, superuser, sys: operator, atau pengguna dengan izin sistem TABEL SISTEM AKSES dapat melihat. SVV_RLS_APPLICED_POLICY Perhatikan bahwa sys:secadmin tidak diberikan izin ini secara default.

Untuk mengizinkan pengguna akses penuh ke RLS-protected relasi, Anda dapat memberikan izin IGNORE RLS. Superuser atau sys:secadmin secara otomatis diberikan IGNORE RLS. Untuk informasi selengkapnya, lihat HIBAH.

Untuk menjelaskan filter kebijakan RLS dari kueri dalam rencana EXPLAIN untuk memecahkan masalah RLS-related kueri, Anda dapat memberikan izin EXPLAIN RLS kepada pengguna mana pun. Untuk informasi selengkapnya, lihat HIBAH dan EXPLAIN.

Policy-dependent objek dan prinsip

Untuk memberikan keamanan bagi aplikasi dan mencegah objek kebijakan menjadi basi atau tidak valid, Amazon Redshift tidak mengizinkan penghapusan atau perubahan objek yang dirujuk oleh kebijakan RLS.

Berikut ini mencantumkan dependensi objek skema yang dilacak Amazon Redshift untuk kebijakan RLS.

- Saat melacak ketergantungan objek skema untuk tabel target, Amazon Redshift mengikuti aturan berikut:
  - Amazon Redshift melepaskan kebijakan dari relasi, pengguna, peran, atau publik saat Anda melepaskan tabel target.
  - Saat Anda mengganti nama tabel target, kebijakan yang dilampirkan tidak akan berdampak.

  - Anda hanya dapat melepaskan kolom tabel target yang direferensikan di dalam definisi kebijakan jika Anda menghapus atau melepaskan kebijakan terlebih dahulu. Ini juga berlaku ketika opsi CASCADE ditentukan. Anda dapat melepaskan kolom lain di tabel target.
  - Anda tidak dapat mengganti nama kolom yang dirujuk dari tabel target. Untuk mengganti nama kolom yang dirujuk, lepaskan kebijakan terlebih dahulu. Ini juga berlaku ketika opsi CASCADE ditentukan.
  - Anda tidak dapat mengubah jenis kolom yang dirujuk, bahkan ketika Anda menentukan opsi CASCADE.
- Saat melacak ketergantungan objek skema untuk tabel pencarian, Amazon Redshift mengikuti aturan berikut:
  - Anda tidak dapat menjatuhkan tabel pencarian. Untuk melepaskan tabel pencarian, pertama-tama lepaskan kebijakan di mana tabel pencarian dirujuk.
  - Anda tidak dapat mengganti nama tabel pencarian. Untuk mengganti nama tabel pencarian, pertama-tama lepaskan kebijakan di mana tabel pencarian dirujuk. Ini juga berlaku ketika opsi CASCADE ditentukan.
  - Anda tidak dapat menghapus kolom tabel pencarian yang digunakan dalam definisi kebijakan. Untuk menghapus kolom tabel pencarian yang digunakan dalam definisi kebijakan, pertama-tama lepas kebijakan di mana tabel pencarian dirujuk. Ini juga berlaku ketika opsi CASCADE ditentukan dalam pernyataan ALTER TABLE DROP COLUMN. Anda dapat melepaskan kolom lain di tabel pencarian.
  - Anda tidak dapat mengganti nama kolom yang dirujuk dari tabel pencarian. Untuk mengganti nama kolom yang dirujuk, pertama-tama lepaskan kebijakan di mana tabel pencarian dirujuk. Ini juga berlaku ketika opsi CASCADE ditentukan.
  - Anda tidak dapat mengubah jenis kolom yang dirujuk.
- Saat pengguna atau peran dihapus, Amazon Redshift melepaskan semua kebijakan yang dilampirkan ke pengguna atau peran secara otomatis.
- Saat Anda menggunakan opsi CASCADE dalam pernyataan DROP SCHEMA, Amazon Redshift juga menghapus relasi dalam skema. Ini juga menjatuhkan hubungan dalam skema lain yang bergantung pada hubungan dalam skema yang dijatuhkan. Untuk relasi yang merupakan tabel pencarian dalam kebijakan, Amazon Redshift gagal dalam DROP SCHEMA DDL. Untuk hubungan apa pun yang dibatalkan oleh pernyataan DROP SCHEMA, Amazon Redshift melepaskan semua kebijakan yang melekat pada relasi tersebut.
- Anda hanya dapat menghapus fungsi pencarian (fungsi yang dirujuk di dalam definisi kebijakan) saat Anda juga menghapus kebijakan. Ini juga berlaku ketika opsi CASCADE ditentukan.

- Saat kebijakan dilampirkan ke tabel, Amazon Redshift memeriksa apakah tabel ini adalah tabel pencarian dalam kebijakan yang berbeda. Jika demikian, Amazon Redshift tidak akan mengizinkan melampirkan kebijakan ke tabel ini.
- Saat membuat kebijakan RLS, Amazon Redshift memeriksa apakah tabel ini merupakan tabel target untuk kebijakan RLS lainnya. Jika ini masalahnya, Amazon Redshift tidak akan mengizinkan pembuatan kebijakan di tabel ini.

Contoh

Contoh berikut menggambarkan bagaimana ketergantungan skema dilacak.

```
-- The CREATE and ATTACH policy statements for `policy_events` references some
-- target and lookup tables.
-- Target tables are tickit_event_redshift and target_schema.target_event_table.
-- Lookup table is tickit_sales_redshift.
-- Policy `policy_events` has following dependencies:
--   table tickit_sales_redshift column eventid, qtysold
--   table tickit_event_redshift column eventid
--   table target_event_table column eventid
--   schema public and target_schema
CREATE RLS POLICY policy_events
WITH (eventid INTEGER)
USING (
    eventid IN (SELECT eventid FROM tickit_sales_redshift WHERE qtysold <3)
);

ATTACH RLS POLICY policy_events ON tickit_event_redshift TO ROLE analyst;

ATTACH RLS POLICY policy_events ON target_schema.target_event_table TO ROLE consumer;
```

Pertimbangan dan batasan menggunakan kebijakan RLS

Pertimbangan-pertimbangan

Berikut ini adalah pertimbangan untuk bekerja dengan kebijakan RLS:

- Amazon Redshift menerapkan kebijakan RLS untuk pernyataan SELECT, UPDATE, atau DELETE.
- Amazon Redshift tidak menerapkan kebijakan RLS untuk pernyataan INSERT, COPY, ALTER TABLE APPEND.

- Kebijakan RLS dapat dilampirkan ke tabel, tampilan, tampilan pengikatan terlambat (LBV), dan tampilan terwujud (MV).
- Row-level keamanan bekerja dengan keamanan tingkat kolom untuk melindungi data Anda.
- Saat RLS diaktifkan untuk relasi sumber, Amazon Redshift mendukung pernyataan ALTER TABLE APPEND untuk pengguna super, pengguna yang telah secara eksplisit diberikan izin sistem IGNORE RLS, atau peran sys:secadmin. Dalam hal ini, Anda dapat menjalankan pernyataan ALTER TABLE APPEND untuk menambahkan baris ke tabel target dengan memindahkan data dari tabel sumber yang ada. Amazon Redshift memindahkan semua tupel dari relasi sumber ke relasi target. Status RLS dari relasi target tidak mempengaruhi pernyataan ALTER TABLE APPEND.
- Untuk memfasilitasi migrasi dari sistem gudang data lainnya, Anda dapat mengatur dan mengambil variabel konteks sesi yang disesuaikan untuk koneksi dengan menentukan nama dan nilai variabel.

  Contoh berikut menetapkan variabel konteks sesi untuk kebijakan keamanan tingkat baris (RLS).

```
-- Set a customized context variable.
SELECT set_config('app.category', 'Concerts', FALSE);

-- Create a RLS policy using current_setting() to get the value of a customized
 context variable.
CREATE RLS POLICY policy_categories
WITH (catgroup VARCHAR(10))
USING (catgroup = current_setting('app.category', FALSE));

-- Set correct roles and attach the policy on the target table to one or more roles.
ATTACH RLS POLICY policy_categories ON tickit_category_redshift TO ROLE analyst, ROLE
 dbadmin;
```

  Untuk detail tentang cara mengatur dan mengambil variabel konteks sesi yang disesuaikan, bukaSET,SETEL_KONFIGURASI,MENUNJUKKAN,SETELINGAN_SAAT INI, danMENGATUR ULANG. Untuk informasi lebih lanjut tentang memodifikasi konfigurasi server secara umum, bukaMemodifikasi konfigurasi server.

  > **Important**
  >
  > Saat menggunakan variabel konteks sesi dalam kebijakan RLS, kebijakan keamanan bergantung pada pengguna atau peran yang memanggil kebijakan. Berhati-hatilah untuk

menghindari kerentanan keamanan saat menggunakan variabel konteks sesi dalam kebijakan RLS.

- Mengubah pengguna sesi menggunakan SET SESSION AUTHORIZATION antara DECLARE dan FETCH atau antara pernyataan FETCH berikutnya tidak akan menyegarkan paket yang sudah disiapkan berdasarkan kebijakan pengguna pada waktu DECLARE. Hindari mengubah pengguna sesi saat kursor digunakan dengan RLS-protected tabel.
- Ketika objek dasar di dalam objek tampilan ada RLS-protected, kebijakan yang dilampirkan ke pengguna yang menjalankan kueri diterapkan pada objek dasar masing-masing. Ini berbeda dari pemeriksaan izin tingkat objek, di mana izin pemilik tampilan diperiksa terhadap objek dasar tampilan. Anda dapat melihat RLS-protected hubungan kueri dalam output rencana EXPLAIN.
- Ketika fungsi yang ditentukan pengguna (UDF) direferensikan dalam kebijakan RLS dari relasi yang dilampirkan ke pengguna, pengguna harus memiliki izin EXECUTE atas UDF untuk menanyakan relasi.
- Row-level keamanan mungkin membatasi optimasi kueri. Sebaiknya evaluasi kinerja kueri dengan cermat sebelum menerapkan RLS-protected tampilan pada kumpulan data besar.
- Row-level kebijakan keamanan yang diterapkan pada tampilan yang mengikat terlambat mungkin didorong ke tabel federasi. Kebijakan RLS ini mungkin terlihat di log mesin pemrosesan eksternal.

Batasan

Berikut ini adalah batasan saat bekerja dengan kebijakan RLS:

- Kebijakan RLS tidak dapat dilampirkan ke tabel eksternal dan beberapa jenis relasi lainnya. Untuk informasi selengkapnya, lihat LAMPIRKAN KEBIJAKAN RLS.
- Amazon Redshift mendukung pernyataan SELECT untuk kebijakan RLS tertentu dengan pencarian yang memiliki gabungan kompleks, tetapi tidak mendukung pernyataan UPDATE atau DELETE. Dalam kasus dengan pernyataan UPDATE atau DELETE, Amazon Redshift mengembalikan kesalahan berikut:

```
ERROR: One of the RLS policies on target relation is not supported in UPDATE/DELETE.
```

- Setiap kali fungsi yang ditentukan pengguna (UDF) direferensikan dalam kebijakan RLS dari relasi yang dilampirkan ke pengguna, pengguna harus memiliki izin EXECUTE atas UDF untuk menanyakan relasi.
- Subquery yang berkorelasi tidak didukung. Amazon Redshift mengembalikan kesalahan berikut:

```
ERROR: RLS policy could not be rewritten.
```

- Amazon Redshift tidak mendukung berbagi data dengan RLS. Jika relasi tidak menonaktifkan RLS untuk berbagi data, kueri gagal pada cluster konsumen dengan kesalahan berikut:

  ```
  RLS-protected relation "rls_protected_table" cannot be accessed via datasharing
   query.
  ```

  Anda dapat mematikan RLS untuk berbagi data menggunakan perintah ALTER TABLE dengan parameter ROW LEVEL SECURITY OFF FOR DATASHARES. Untuk informasi lebih lanjut tentang menggunakan ALTER TABLE untuk mengaktifkan atau menonaktifkan RLS, buka. ALTER TABLE

- Dalam kueri lintas database, Amazon Redshift memblokir pembacaan ke RLS-protected relasi. Pengguna dengan izin IGNORE RLS dapat mengakses relasi yang dilindungi menggunakan kueri lintas database. Ketika pengguna tanpa izin IGNORE RLS mengakses RLS-protected relasi melalui kueri lintas database, kesalahan berikut muncul:

  ```
  RLS-protected relation "rls_protected_table" cannot be accessed via cross-database
   query.
  ```

- ALTER RLS POLICY hanya mendukung modifikasi kebijakan RLS menggunakan klausa USING (using_predicate_exp). Anda tidak dapat mengubah kebijakan RLS dengan klausa WITH saat menjalankan ALTER RLS POLICY.
- Anda tidak dapat menanyakan relasi yang mengaktifkan keamanan tingkat baris jika nilai untuk salah satu opsi konfigurasi berikut tidak cocok dengan nilai default sesi:
  - `enable_case_sensitive_super_attribute`
  - `enable_case_sensitive_identifier`
  - `downcase_delimited_identifier`

  Pertimbangkan untuk mengatur ulang opsi konfigurasi sesi Anda jika Anda mencoba menanyakan relasi dengan keamanan tingkat baris aktif dan melihat pesan “Relasi yang dilindungi RLS tidak mendukung konfigurasi tingkat sesi pada sensitivitas huruf besar yang berbeda dari nilai defaultnya.”
- Jika cluster yang disediakan atau namespace tanpa server memiliki kebijakan keamanan tingkat baris, perintah berikut diblokir untuk pengguna biasa:

```
ALTER <current_user> SET enable_case_sensitive_super_attribute/
enable_case_sensitive_identifier/downcase_delimited_identifier
```

Saat membuat kebijakan RLS, sebaiknya ubah setelan opsi konfigurasi default untuk pengguna biasa agar sesuai dengan setelan opsi konfigurasi sesi pada saat kebijakan dibuat. Pengguna super dan pengguna dengan hak istimewa ALTER USER dapat melakukan ini dengan menggunakan pengaturan grup parameter atau perintah ALTER USER. Untuk informasi tentang grup parameter, lihat grup parameter Amazon Redshift di Panduan Manajemen Amazon Redshift. Untuk informasi tentang perintah ALTER USER, lihatALTER USER.

- Tampilan dan tampilan yang mengikat terlambat dengan kebijakan keamanan tingkat baris tidak dapat diganti oleh pengguna biasa menggunakan perintah. CREATE VIEW Untuk mengganti tampilan atau LBV dengan kebijakan RLS, pertama-tama lepaskan kebijakan RLS yang dilampirkan padanya, ganti tampilan atau LBV, dan lampirkan kembali kebijakan. Pengguna super dan pengguna dengan `sys:secadmin permission` dapat menggunakan CREATE VIEW pada tampilan atau LBV dengan kebijakan RLS tanpa melepaskan kebijakan.
- Tampilan dengan kebijakan keamanan tingkat baris tidak dapat mereferensikan tabel sistem dan tampilan sistem.
- Tampilan pengikatan terlambat yang direferensikan oleh tampilan biasa tidak dapat dilindungi RLS.
- RLS-protected hubungan dan data bersarang dari danau data tidak dapat diakses dalam kueri yang sama.

Praktik terbaik untuk kinerja RLS

Berikut ini adalah praktik terbaik untuk kinerja yang lebih baik dari Amazon Redshift pada tabel yang dilindungi oleh RLS.

Keamanan operator dan fungsi

Saat menanyakan RLS-protected tabel, penggunaan operator atau fungsi tertentu dapat menyebabkan penurunan kinerja. Amazon Redshift mengklasifikasikan operator dan fungsi sebagai aman atau tidak aman untuk menanyakan tabel. RLS-protected Fungsi atau operator diklasifikasikan sebagai RLS-safe ketika tidak memiliki efek samping yang dapat diamati tergantung pada input. Secara khusus, RLS-safe fungsi atau operator tidak dapat menjadi salah satu dari berikut ini:

- Menghasilkan nilai input, atau nilai apa pun yang bergantung pada nilai input, dengan atau tanpa pesan kesalahan.

- Gagal atau mengembalikan kesalahan yang bergantung pada nilai input.

RLS-unsafe operator meliputi:

- Operator aritmatika — +, -,/, *,%.
- Operator teks - LIKE dan MIRIP DENGAN.
- Beberapa operator pemeran. Perhatikan bahwa cast tertentu diklasifikasikan sebagai aman, termasuk konversi tanggal-ke-timestamp dan timestamp-to-date, pemindakan pelebaran integer (seperti INT2 ke INT8 atau INT4 ke INT8), dan pemindahan bilangan bulat ke teks.
- UDF.

Gunakan pernyataan SELECT berikut untuk memeriksa keamanan operator dan fungsi.

```
SELECT proname, proc_is_rls_safe(oid) FROM pg_proc;
```

Amazon Redshift memberlakukan pembatasan pada urutan evaluasi predikat pengguna yang berisi RLS-unsafe operator dan fungsi saat merencanakan kueri pada tabel. RLS-protected Kueri yang meru RLS-unsafe juk operator atau fungsi dapat menyebabkan penurunan kinerja saat menanyakan tabel. RLS-protected Kinerja dapat menurun secara signifikan ketika Amazon Redshift tidak dapat mendorong RLS-unsafe predikat ke pemindaian tabel dasar untuk memanfaatkan kunci pengurutan. Untuk kinerja yang lebih baik, hindari kueri RLS-unsafe menggunakan predikat yang memanfaatkan kunci pengurutan. Untuk memverifikasi bahwa Amazon Redshift dapat menekan operator dan fungsi ke bawah, Anda dapat menggunakan pernyataan EXPLAIN dalam kombinasi dengan izin sistem EXPLAIN RLS.

Fungsi aman secara kondisional

Beberapa fungsi diklasifikasikan sebagai tidak aman secara umum tetapi menjadi aman ketika argumen tertentu adalah nilai konstan (nilai literal, bukan referensi kolom). Ketika argumen yang relevan adalah konstanta, Amazon Redshift dapat mendorong predikat ini ke pemindaian tabel dasar, meningkatkan kinerja kueri.

Misalnya, aman `DATE_TRUNC('day', timestamp_col)` secara kondisional karena `'day'` merupakan literal konstan. Namun, `DATE_TRUNC(datepart_col, timestamp_col)` tidak aman secara kondisional karena `datepart_col` merupakan referensi kolom.

Tabel berikut mencantumkan kategori fungsi aman kondisional dan argumen mana yang harus konstan agar fungsi dianggap aman.

Kategori fungsi	Fungsi	Argumen yang harus konstan
TANGGAL_TRUNC	DATE_TRUNC (bagian tanggal, stempel waktu), DATE_TRUNC (tanggalan tanggal, stempel waktu)	datepart (argumen pertama)
EKSTRAK/DATE_PART	EKSTRAK (bidang DARI sumber), DATE_PART (bidang, sumber) untuk stempel waktu, timestamptz, tanggal, waktu, timez, interval, dan tipe datetime	bidang (argumen pertama)
TANGGAL DIAKHIRI	DATEDIFF (datepart, start, end) untuk stempel waktu dan tipe waktu	datepart (argumen pertama)
TO_CHAR	TO_CHAR (stempel waktu, format), TO_CHAR (stempel waktu, format)	format (argumen kedua)
CONVERT_TIMEZONE	CONVERT_TIMEZONE (source_tz, target_tz, stempel waktu), CONVERT_TIMEZONE (target_tz, stempel waktu)	argumen zona waktu
KIRI/KANAN	KIRI (string, panjang), KANAN (string, panjang)	panjang (argumen kedua)
SUBSTRING	SUBSTRING (string, start, length) untuk jenis teks, bytea, dan varbyte	panjang (argumen ketiga)
BAGIAN_BAGIAN	SPLIT_PART (string, pembatas, bagian)	bagian (argumen ketiga)

Gunakan pernyataan SELECT berikut untuk memeriksa fungsi mana yang aman secara kondisional dan posisi argumen mana yang harus konstan.

```
SELECT proname, proc_is_rls_conditionally_safe(oid) FROM pg_proc
WHERE proc_is_rls_conditionally_safe(oid) IS NOT NULL;
```

Fungsi mengembalikan array posisi argumen terindeks 0 yang harus konstan, atau NULL jika fungsi tidak aman secara kondisional.

Caching hasil

Untuk mengurangi runtime kueri dan meningkatkan kinerja sistem, Amazon Redshift menyimpan hasil jenis kueri tertentu dalam memori pada node pemimpin.

Amazon Redshift menggunakan hasil cache untuk RLS-protected tabel pemindaian kueri baru ketika semua kondisi untuk tabel yang tidak dilindungi benar dan ketika semua hal berikut benar:

- Tabel atau tampilan dalam kebijakan belum dimodifikasi.
- Kebijakan tidak menggunakan fungsi yang harus dievaluasi setiap kali dijalankan, seperti GETDATE atau CURRENT_USER.

Untuk kinerja yang lebih baik, hindari menggunakan predikat kebijakan yang tidak memenuhi kondisi sebelumnya.

Untuk informasi selengkapnya tentang cache hasil di Amazon Redshift, lihat. Caching hasil

Kebijakan yang kompleks

Untuk kinerja yang lebih baik, hindari menggunakan kebijakan kompleks dengan subkueri yang menggabungkan beberapa tabel.

Menggunakan argumen konstan untuk pushdown yang lebih baik

Saat menggunakan fungsi seperti DATE_TRUNC, EXTRACT, DATEDIFF, TO_CHAR, CONVERT_TIMEZONE, LEFT, RIGHT, SUBSTRING, atau SPLIT_PART dalam kueri pada RLS-protected tabel, gunakan argumen literal konstan alih-alih referensi kolom untuk argumen yang mengontrol perilaku kesalahan. Hal ini memungkinkan Amazon Redshift untuk mengklasifikasikan fungsi-fungsi ini sebagai aman dan mendorong predikat turun ke pemindaian tabel dasar, yang secara signifikan dapat meningkatkan kinerja.

Misalnya, kueri berikut memungkinkan pushdown predikat karena argumen datepart adalah konstanta:

```
SELECT * FROM rls_protected_table
WHERE DATE_TRUNC('month', event_date) = '2024-01-01';
```

Kueri berikut mencegah pushdown predikat karena argumen datepart adalah referensi kolom:

```
SELECT * FROM rls_protected_table
WHERE DATE_TRUNC(datepart_col, event_date) = '2024-01-01';
```

Untuk daftar lengkap fungsi aman kondisional dan argumen mana yang harus konstan, lihatFungsi aman secara kondisional.

Row-level contoh keamanan ujung ke ujung

Berikut ini adalah contoh ujung ke ujung untuk menggambarkan bagaimana superuser menciptakan beberapa pengguna dan peran. Kemudian, pengguna dengan peran secadmin membuat, melampirkan, melepaskan, dan menghapus kebijakan RLS. Contoh ini menggunakan database sampel tickit. Untuk informasi selengkapnya, lihat Memuat data dari Amazon S3 ke Amazon Redshift di Panduan Memulai Amazon Redshift.

```
-- Create users and roles referenced in the policy statements.
CREATE ROLE analyst;
CREATE ROLE consumer;
CREATE ROLE dbadmin;
CREATE ROLE auditor;
CREATE USER bob WITH PASSWORD 'Name_is_bob_1';
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
CREATE USER joe WITH PASSWORD 'Name_is_joe_1';
CREATE USER molly WITH PASSWORD 'Name_is_molly_1';
CREATE USER bruce WITH PASSWORD 'Name_is_bruce_1';
GRANT ROLE sys:secadmin TO bob;
GRANT ROLE analyst TO alice;
GRANT ROLE consumer TO joe;
GRANT ROLE dbadmin TO molly;
GRANT ROLE auditor TO bruce;
GRANT ALL ON TABLE tickit_category_redshift TO PUBLIC;
GRANT ALL ON TABLE tickit_sales_redshift TO PUBLIC;
GRANT ALL ON TABLE tickit_event_redshift TO PUBLIC;

-- Create table and schema referenced in the policy statements.
CREATE SCHEMA target_schema;
GRANT ALL ON SCHEMA target_schema TO PUBLIC;
CREATE TABLE target_schema.target_event_table (LIKE tickit_event_redshift);
GRANT ALL ON TABLE target_schema.target_event_table TO PUBLIC;

-- Change session to analyst alice.
```

```
SET SESSION AUTHORIZATION alice;

-- Check the tuples visible to analyst alice.
-- Should contain all 3 categories.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');

SELECT poldb, polname, polalias, polatts, polqual, polenabled, polmodifiedby FROM
 svv_rls_policy WHERE poldb = CURRENT_DATABASE();

ATTACH RLS POLICY policy_concerts ON tickit_category_redshift TO ROLE analyst, ROLE
 dbadmin;

ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;

SELECT * FROM svv_rls_attached_policy;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check that tuples with only `Concert` category will be visible to analyst alice.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to consumer joe.
SET SESSION AUTHORIZATION joe;

-- Although the policy is attached to a different role, no tuples will be
-- visible to consumer joe because the default deny all policy is applied.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to dbadmin molly.
SET SESSION AUTHORIZATION molly;
```

```
-- Check that tuples with only `Concert` category will be visible to dbadmin molly.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Check that EXPLAIN output contains RLS SecureScan to prevent disclosure of
-- sensitive information such as RLS filters.
EXPLAIN SELECT catgroup, count(*) FROM tickit_category_redshift GROUP BY catgroup ORDER
 BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

-- Grant IGNORE RLS permission so that RLS policies do not get applicable to role
 dbadmin.
GRANT IGNORE RLS TO ROLE dbadmin;

-- Grant EXPLAIN RLS permission so that anyone in role auditor can view complete
 EXPLAIN output.
GRANT EXPLAIN RLS TO ROLE auditor;

-- Change session to dbadmin molly.
SET SESSION AUTHORIZATION molly;

-- Check that all tuples are visible to dbadmin molly because `IGNORE RLS` is granted
 to role dbadmin.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to auditor bruce.
SET SESSION AUTHORIZATION bruce;

-- Check explain plan is visible to auditor bruce because `EXPLAIN RLS` is granted to
 role auditor.
EXPLAIN SELECT catgroup, count(*) FROM tickit_category_redshift GROUP BY catgroup ORDER
 BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

DETACH RLS POLICY policy_concerts ON tickit_category_redshift FROM ROLE analyst, ROLE
 dbadmin;
```

```
-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check that no tuples are visible to analyst alice.
-- Although the policy is detached, no tuples will be visible to analyst alice
-- because of default deny all policy is applied if the table has RLS on.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

CREATE RLS POLICY policy_events
WITH (eventid INTEGER) AS ev
USING (
    ev.eventid IN (SELECT eventid FROM tickit_sales_redshift WHERE qtysold <3)
);

ATTACH RLS POLICY policy_events ON tickit_event_redshift TO ROLE analyst;
ATTACH RLS POLICY policy_events ON target_schema.target_event_table TO ROLE consumer;

RESET SESSION AUTHORIZATION;

-- Can not cannot alter type of dependent column.
ALTER TABLE target_schema.target_event_table ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_event_redshift ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_sales_redshift ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_sales_redshift ALTER COLUMN qtysold TYPE float;

-- Can not cannot rename dependent column.
ALTER TABLE target_schema.target_event_table RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_event_redshift RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_sales_redshift RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_sales_redshift RENAME COLUMN qtysold TO renamed_qtysold;

-- Can not drop dependent column.
ALTER TABLE target_schema.target_event_table DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_event_redshift DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_sales_redshift DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_sales_redshift DROP COLUMN qtysold CASCADE;

-- Can not drop lookup table.
```

```
DROP TABLE tickit_sales_redshift CASCADE;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

DROP RLS POLICY policy_concerts;
DROP RLS POLICY IF EXISTS policy_events;

ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY OFF;

RESET SESSION AUTHORIZATION;

-- Drop users and roles.
DROP USER bob;
DROP USER alice;
DROP USER joe;
DROP USER molly;
DROP USER bruce;
DROP ROLE analyst;
DROP ROLE consumer;
DROP ROLE auditor FORCE;
DROP ROLE dbadmin FORCE;
```

Keamanan metadata

Seperti keamanan tingkat baris Amazon Redshift, keamanan metadata memberi Anda kontrol yang lebih terperinci atas metadata Anda. Jika keamanan metadata diaktifkan untuk cluster yang disediakan atau grup kerja tanpa server, pengguna dapat melihat metadata untuk objek yang memiliki akses tampilannya. Dengan keamanan metadata, Anda dapat memisahkan visibilitas berdasarkan kebutuhan Anda. Misalnya, Anda dapat menggunakan gudang data tunggal untuk memusatkan semua penyimpanan data Anda. Namun, jika Anda menyimpan data untuk beberapa sektor, mengelola keamanan bisa menjadi merepotkan. Dengan keamanan metadata diaktifkan, Anda dapat mengonfigurasi visibilitas Anda. Pengguna dari satu sektor dapat memiliki lebih banyak visibilitas atas objek mereka, sementara Anda membatasi akses tampilan ke pengguna sektor lain. Keamanan metadata mendukung semua jenis objek, seperti skema, tabel, tampilan, tampilan terwujud, prosedur tersimpan, fungsi yang ditentukan pengguna, dan model pembelajaran mesin.

Pengguna dapat melihat metadata objek dalam keadaan berikut:

- Jika akses objek diberikan kepada pengguna.

- Jika akses objek diberikan ke grup atau peran yang menjadi bagiannya pengguna.
- Objeknya bersifat publik.
- Pengguna adalah pemilik objek database.

Untuk mengaktifkan keamanan metadata, gunakan perintah ALTER SYSTEM. Berikut ini adalah sintaks cara menggunakan perintah ALTER SYSTEM dengan keamanan metadata.

```
ALTER SYSTEM SET metadata_security=[true|t|on|false|f|off];
```

Saat Anda mengaktifkan keamanan metadata, semua pengguna yang memiliki izin yang diperlukan dapat melihat metadata objek yang relevan yang dapat mereka akses. Jika Anda ingin hanya pengguna tertentu yang dapat melihat keamanan metadata, berikan `ACCESS CATALOG` izin ke peran, lalu tetapkan peran tersebut kepada pengguna. Untuk informasi selengkapnya tentang menggunakan peran untuk mengontrol keamanan dengan lebih baik, lihat kontrol Role-based akses.

Contoh berikut menunjukkan cara memberikan `ACCESS CATALOG` izin ke peran, dan kemudian menetapkan peran tersebut kepada pengguna. Untuk informasi selengkapnya tentang pemberian izin, lihat perintah GRANT.

```
CREATE ROLE sample_metadata_viewer;

GRANT ACCESS CATALOG TO ROLE sample_metadata_viewer;

GRANT ROLE sample_metadata_viewer to salesadmin;
```

Jika Anda lebih suka menggunakan peran yang sudah ditentukan, peran yang ditentukan sistem `operator`,, `secadmindba`, dan `superuser` semuanya memiliki izin yang diperlukan untuk melihat metadata objek. Secara default, pengguna super dapat melihat katalog lengkap.

```
GRANT ROLE operator to sample_user;
```

Jika Anda menggunakan peran untuk mengontrol keamanan metadata, Anda memiliki akses ke semua tampilan sistem dan fungsi yang disertakan dengan kontrol akses berbasis peran. Misalnya, Anda dapat menanyakan tampilan SVV_ROLES untuk melihat semua peran. Untuk melihat apakah pengguna adalah anggota peran atau grup, gunakan fungsi USER_IS_MEMBER_OF. Untuk daftar lengkap tampilan SVV, lihat tampilan metadata SVV. Untuk daftar fungsi informasi sistem, lihat Fungsi informasi sistem.

Penyembunyian data dinamis

> **Note**
>
> Amazon Redshift secara otomatis menutupi kolom tabel sistem tertentu saat mencatat informasi tentang kueri yang dibuat ke tampilan Katalog Data untuk mencegah paparan metadata sensitif. Untuk informasi selengkapnya, lihat Log in aman di Panduan Manajemen Amazon Redshift.

Menggunakan dynamic data masking (DDM) di Amazon Redshift, Anda dapat melindungi data sensitif di gudang data Anda. Anda dapat memanipulasi cara Amazon Redshift menampilkan data sensitif kepada pengguna pada saat kueri, tanpa mengubahnya dalam database. Anda mengontrol akses ke data melalui kebijakan masking yang menerapkan aturan pengaburan khusus ke pengguna atau peran tertentu. Dengan cara itu, Anda dapat menanggapi perubahan persyaratan privasi tanpa mengubah data yang mendasarinya atau mengedit kueri SQL.

Kebijakan penyembunyian data dinamis menyembunyikan, mengaburkan, atau membuat nama samaran data yang cocok dengan format tertentu. Saat dilampirkan ke tabel, ekspresi masking diterapkan ke satu atau lebih kolomnya. Anda dapat memodifikasi kebijakan masking lebih lanjut untuk hanya menerapkannya pada pengguna tertentu, atau ke peran yang ditentukan pengguna yang dapat Anda buat dengannyaRole-based kontrol akses (RBAC). Selain itu, Anda dapat menerapkan DDM pada tingkat sel dengan menggunakan kolom bersyarat saat membuat kebijakan masking. Untuk informasi selengkapnya tentang masking bersyarat, lihatPenyembunyian data dinamis bersyarat.

Anda dapat menerapkan beberapa kebijakan masking dengan berbagai tingkat pengaburan ke kolom yang sama dalam tabel dan menetapkannya ke peran yang berbeda. Untuk menghindari konflik saat Anda memiliki peran berbeda dengan kebijakan berbeda yang berlaku untuk satu kolom, Anda dapat menetapkan prioritas untuk setiap aplikasi. Dengan cara itu, Anda dapat mengontrol data apa yang dapat diakses oleh pengguna atau peran tertentu. Kebijakan DDM dapat menyunting data sebagian atau seluruhnya, atau menghasahnya dengan menggunakan fungsi yang ditentukan pengguna yang ditulis dalam SQL, Python, atau dengan. AWS Lambda Dengan menutupi data menggunakan hash, Anda dapat menerapkan gabungan pada data ini tanpa akses ke informasi yang berpotensi sensitif.

Perintah SQL untuk mengelola kebijakan penyembunyian data dinamis

Anda dapat melakukan tindakan berikut untuk membuat, melampirkan, melepaskan, dan menghapus kebijakan penyembunyian data dinamis:

- Untuk membuat kebijakan DDM, gunakan BUAT KEBIJAKAN MASKING perintah.

  Berikut ini adalah contoh pembuatan kebijakan masking menggunakan fungsi SHA-2 hash.

```
CREATE MASKING POLICY hash_credit
WITH (credit_card varchar(256))
USING (sha2(credit_card + 'testSalt', 256));
```

- Untuk mengubah kebijakan DDM yang ada, gunakan MENGUBAH KEBIJAKAN MASKING perintah.

  Berikut ini adalah contoh mengubah kebijakan masking yang ada.

```
ALTER MASKING POLICY hash_credit
USING (sha2(credit_card + 'otherTestSalt', 256));
```

- Untuk melampirkan kebijakan DDM pada tabel ke satu atau lebih pengguna atau peran, gunakan LAMPIRKAN KEBIJAKAN MASKING perintah.

  Berikut ini adalah contoh melampirkan kebijakan masking ke column/role pasangan.

```
 ATTACH MASKING POLICY hash_credit
ON credit_cards (credit_card)
TO ROLE science_role
PRIORITY 30;
```

  Klausa PRIORITY menentukan kebijakan masking mana yang berlaku untuk sesi pengguna saat beberapa kebijakan dilampirkan ke kolom yang sama. Misalnya, jika pengguna dalam contoh sebelumnya memiliki kebijakan masking lain yang dilampirkan ke kolom kartu kredit yang sama dengan prioritas 20, kebijakan science_role adalah kebijakan yang berlaku, karena memiliki prioritas yang lebih tinggi yaitu 30.

- Untuk melepaskan kebijakan DDM pada tabel dari satu atau lebih pengguna atau peran, gunakan perintahLEPASKAN KEBIJAKAN MASKING.

  Berikut ini adalah contoh melepaskan kebijakan masking dari column/role pasangan.

```
DETACH MASKING POLICY hash_credit
ON credit_cards(credit_card)
FROM ROLE science_role;
```

- Untuk menghapus kebijakan DDM dari semua database, gunakan KEBIJAKAN DROP MASKING perintah.

  Berikut ini adalah contoh menghapus kebijakan masking dari semua database.

  ```
  DROP MASKING POLICY hash_credit;
  ```

Hirarki kebijakan penyembunyian data dinamis

Saat melampirkan beberapa kebijakan masking, pertimbangkan hal berikut:

- Anda dapat melampirkan beberapa kebijakan masking ke satu kolom.
- Ketika beberapa kebijakan masking berlaku untuk kueri, kebijakan prioritas tertinggi yang dilampirkan ke setiap kolom akan berlaku. Pertimbangkan contoh berikut.

  ```
  ATTACH MASKING POLICY partial_hash
  ON credit_cards(address, credit_card)
  TO ROLE analytics_role
  PRIORITY 20;

  ATTACH MASKING POLICY full_hash
  ON credit_cards(credit_card, ssn)
  TO ROLE auditor_role
  PRIORITY 30;

  SELECT address, credit_card, ssn
  FROM credit_cards;
  ```

  Saat menjalankan pernyataan SELECT, pengguna dengan peran analitik dan auditor melihat kolom alamat dengan kebijakan `partial_hash` masking yang diterapkan. Mereka melihat kolom kartu kredit dan SSN dengan kebijakan `full_hash` masking diterapkan karena `full_hash` kebijakan tersebut memiliki prioritas yang lebih tinggi pada kolom kartu kredit.
- Jika Anda tidak menentukan prioritas saat melampirkan kebijakan masking, prioritas defaultnya adalah 0.

- Anda tidak dapat melampirkan dua kebijakan ke kolom yang sama dengan prioritas yang sama.
- Anda tidak dapat melampirkan dua kebijakan ke kombinasi pengguna dan kolom atau peran dan kolom yang sama.
- Ketika beberapa kebijakan masking diterapkan di sepanjang jalur SUPER yang sama saat dilampirkan ke pengguna atau peran yang sama, hanya lampiran prioritas tertinggi yang berlaku. Pertimbangkan contoh berikut.

  Contoh pertama menunjukkan dua kebijakan masking yang dilampirkan pada jalur yang sama, dengan kebijakan prioritas yang lebih tinggi mulai berlaku.

```
ATTACH MASKING POLICY hide_name
ON employees(col_person.name)
TO PUBLIC
PRIORITY 20;

ATTACH MASKING POLICY hide_last_name
ON employees(col_person.name.last)
TO PUBLIC
PRIORITY 30;

--Only the hide_last_name policy takes effect.
SELECT employees.col_person.name FROM employees;
```

  Contoh kedua menunjukkan dua kebijakan masking yang dilampirkan ke jalur yang berbeda dalam objek SUPER yang sama, tanpa konflik antara kebijakan. Kedua lampiran akan berlaku secara bersamaan.

```
ATTACH MASKING POLICY hide_first_name
ON employees(col_person.name.first)
TO PUBLIC
PRIORITY 20;

ATTACH MASKING POLICY hide_last_name
ON employees(col_person.name.last)
TO PUBLIC
PRIORITY 20;

--Both col_person.name.first and col_person.name.last are masked.
SELECT employees.col_person.name FROM employees;
```

Untuk mengonfirmasi kebijakan masking mana yang berlaku untuk pengguna dan kolom atau kombinasi peran dan kolom tertentu, pengguna dengan `sys:secadmin` peran tersebut dapat mencari column/role atau column/user berpasangan dalam KEBIJAKAN PENYAMARAN SVV_TERLAMPIR tampilan sistem. Untuk informasi selengkapnya, lihat Tampilan sistem penyembunyian data dinamis.

Menggunakan penyembunyian data dinamis dengan jalur tipe data SUPER

Amazon Redshift mendukung melampirkan kebijakan penyembunyian data dinamis ke jalur kolom tipe SUPER. Untuk informasi selengkapnya tentang tipe data SUPER, lihatSemi-structured data dalam Amazon Redshift.

Saat melampirkan kebijakan masking ke jalur kolom tipe SUPER, pertimbangkan hal berikut.

- Saat melampirkan kebijakan masking ke jalur pada kolom, kolom tersebut harus didefinisikan sebagai tipe data SUPER. Anda hanya dapat menerapkan kebijakan masking ke nilai skalar di jalur SUPER. Anda tidak dapat menerapkan kebijakan masking ke struktur atau array yang kompleks.
- Anda dapat menerapkan kebijakan masking yang berbeda ke beberapa nilai skalar pada satu kolom SUPER, selama jalur SUPER tidak bertentangan. Misalnya, jalur SUPER `a.b` dan `a.b.c` konflik karena mereka berada di jalur yang sama, dengan `a.b` menjadi induk daria`.b.c`. Jalur SUPER `a.b.c` dan `a.b.d` tidak bertentangan.
- Amazon Redshift tidak dapat memeriksa apakah jalur yang dilampirkan kebijakan masking ada di data dan merupakan jenis yang diharapkan hingga kebijakan diterapkan pada runtime kueri pengguna. Misalnya, saat Anda melampirkan kebijakan masking yang menutupi nilai TEXT ke jalur SUPER yang berisi nilai INT, Amazon Redshift akan mencoba untuk menampilkan jenis nilai di jalur.

  Dalam situasi seperti itu, perilaku Amazon Redshift saat runtime bergantung pada pengaturan konfigurasi Anda untuk menanyakan objek SUPER. Secara default, Amazon Redshift dalam mode lax, dan akan menyelesaikan jalur yang hilang dan cast yang tidak valid seperti `NULL` untuk jalur SUPER yang diberikan. Untuk informasi selengkapnya tentang pengaturan SUPER-related konfigurasi, lihatKonfigurasi SUPER.
- SUPER adalah tipe tanpa skema, yang berarti Amazon Redshift tidak dapat mengkonfirmasi keberadaan nilai pada jalur SUPER tertentu. Jika Anda melampirkan kebijakan masking ke jalur SUPER yang tidak ada dan Amazon Redshift dalam mode lemah, Amazon Redshift akan menyelesaikan jalur ke nilai. `NULL` Sebaiknya pertimbangkan format objek SUPER yang diharapkan dan kemungkinan mereka memiliki atribut tak terduga saat melampirkan kebijakan

masking ke jalur kolom SUPER. Jika menurut Anda mungkin ada skema tak terduga di kolom SUPER Anda, pertimbangkan untuk melampirkan kebijakan masking langsung ke kolom SUPER. Anda dapat menggunakan fungsi informasi tipe SUPER untuk memeriksa atribut dan tipe, dan menggunakan `OBJECT_TRANSFORM` untuk menutupi nilai. Untuk informasi selengkapnya tentang fungsi informasi tipe SUPER, lihatFungsi informasi tipe SUPER.

Contoh

Melampirkan kebijakan masking ke jalur SUPER

Contoh berikut melampirkan beberapa kebijakan masking ke beberapa jalur tipe SUPER dalam satu kolom.

```
CREATE TABLE employees (
    col_person SUPER
);

INSERT INTO employees
VALUES
    (
        json_parse('
            {
                "name": {
                    "first": "John",
                    "last": "Doe"
                },
                "age": 25,
                "ssn": "111-22-3333",
                "company": "Company Inc."
            }
        ')
    ),
    (
        json_parse('
            {
                "name": {
                    "first": "Jane",
                    "last": "Appleseed"
                },
                "age": 34,
                "ssn": "444-55-7777",
                "company": "Organization Org."
```

```
            }
        ')
    )
;
GRANT ALL ON ALL TABLES IN SCHEMA "public" TO PUBLIC;

-- Create the masking policies.

-- This policy converts the given name to all uppercase letters.
CREATE MASKING POLICY mask_first_name
WITH(first_name TEXT)
USING ( UPPER(first_name) );

-- This policy replaces the given name with the fixed string 'XXXX'.
CREATE MASKING POLICY mask_last_name
WITH(last_name TEXT)
USING ( 'XXXX'::TEXT );

-- This policy rounds down the given age to the nearest 10.
CREATE MASKING POLICY mask_age
WITH(age INT)
USING ( (FLOOR(age::FLOAT / 10) * 10)::INT );

-- This policy converts the first five digits of the given SSN to 'XXX-XX'.
CREATE MASKING POLICY mask_ssn
WITH(ssn TEXT)
USING ( 'XXX-XX-'::TEXT || SUBSTRING(ssn::TEXT FROM 8 FOR 4) );

-- Attach the masking policies to the employees table.
ATTACH MASKING POLICY mask_first_name
ON employees(col_person.name.first)
TO PUBLIC;

ATTACH MASKING POLICY mask_last_name
ON employees(col_person.name.last)
TO PUBLIC;

ATTACH MASKING POLICY mask_age
ON employees(col_person.age)
TO PUBLIC;

ATTACH MASKING POLICY mask_ssn
ON employees(col_person.ssn)
TO PUBLIC;
```

```
-- Verify that your masking policies are attached.
SELECT
    policy_name,
    TABLE_NAME,
    priority,
    input_columns,
    output_columns
FROM
    svv_attached_masking_policy;

   policy_name   | table_name | priority |           input_columns           |
 output_columns
-----------------+------------+----------+-----------------------------------
+-----------------------------------
 mask_age        | employees  |        0 | ["col_person.\"age\""]            |
 ["col_person.\"age\""]
 mask_first_name | employees  |        0 | ["col_person.\"name\".\"first\""] |
 ["col_person.\"name\".\"first\""]
 mask_last_name  | employees  |        0 | ["col_person.\"name\".\"last\""]  |
 ["col_person.\"name\".\"last\""]
 mask_ssn        | employees  |        0 | ["col_person.\"ssn\""]            |
 ["col_person.\"ssn\""]
(4 rows)

-- Observe the masking policies taking effect.
SELECT col_person FROM employees ORDER BY col_person.age;

-- This result is formatted for ease of reading.
         col_person
--------------------------------
{
    "name": {
        "first": "JOHN",
        "last": "XXXX"
    },
    "age": 20,
    "ssn": "XXX-XX-3333",
    "company": "Company Inc."
}
{
    "name": {
        "first": "JANE",
        "last": "XXXX"
```

```
    },
    "age": 30,
    "ssn": "XXX-XX-7777",
    "company": "Organization Org."
}
```

Berikut ini adalah beberapa contoh lampiran kebijakan masking yang tidak valid ke jalur SUPER.

```
-- This attachment fails because there is already a policy
-- with equal priority attached to employees.name.last, which is
-- on the same SUPER path as employees.name.
ATTACH MASKING POLICY mask_ssn
ON employees(col_person.name)
TO PUBLIC;
ERROR:  DDM policy "mask_last_name" is already attached on relation "employees" column
 "col_person."name"."last"" with same priority

-- Create a masking policy that masks DATETIME objects.
CREATE MASKING POLICY mask_date
WITH(INPUT DATETIME)
USING ( INPUT );

-- This attachment fails because SUPER type columns can't contain DATETIME objects.
ATTACH MASKING POLICY mask_date
ON employees(col_person.company)
TO PUBLIC;
ERROR:  cannot attach masking policy for output of type "timestamp without time zone"
 to column "col_person."company"" of type "super
```

Berikut ini adalah contoh melampirkan kebijakan masking ke jalur SUPER yang tidak ada. Secara default, Amazon Redshift akan menyelesaikan jalur ke`NULL`.

```
ATTACH MASKING POLICY mask_first_name
ON employees(col_person.not_exists)
TO PUBLIC;

SELECT col_person FROM employees LIMIT 1;

-- This result is formatted for ease of reading.
         col_person
-----------------------------------
{
    "name": {
```

```
        "first": "JOHN",
        "last": "XXXX"
    },
    "age": 20,
    "ssn": "XXX-XX-3333",
    "company": "Company Inc.",
    "not_exists": null
}
```

Penyembunyian data dinamis bersyarat

Anda dapat menutupi data di tingkat sel dengan membuat kebijakan masking dengan ekspresi kondisional dalam ekspresi masking. Misalnya, Anda dapat membuat kebijakan masking yang menerapkan masker berbeda ke nilai, tergantung pada nilai kolom lain di baris tersebut.

Berikut ini adalah contoh penggunaan penyembunyian data bersyarat untuk membuat dan melampirkan kebijakan masking yang sebagian menyunting nomor kartu kredit yang terlibat dalam penipuan, sementara sepenuhnya menyembunyikan semua nomor kartu kredit lainnya. Anda harus menjadi superuser atau memiliki `sys:secadmin` peran untuk menjalankan contoh ini.

```
--Create an analyst role.
CREATE ROLE analyst;

--Create a credit card table. The table contains an is_fraud boolean column,
--which is TRUE if the credit card number in that row was involved in a fraudulent
 transaction.
CREATE TABLE credit_cards (id INT, is_fraud BOOLEAN, credit_card_number VARCHAR(16));

--Create a function that partially redacts credit card numbers.
CREATE FUNCTION REDACT_CREDIT_CARD (credit_card VARCHAR(16))
RETURNS VARCHAR(16) IMMUTABLE
AS $$
    import re
    regexp = re.compile("^([0-9]{6})[0-9]{5,6}([0-9]{4})")

    match = regexp.search(credit_card)
    if match != None:
        first = match.group(1)
        last = match.group(2)
    else:
        first = "000000"
        last = "0000"
```

```
    return "{}XXXXX{}".format(first, last)
$$ LANGUAGE plpythonu;

--Create a masking policy that partially redacts credit card numbers if the is_fraud
 value for that row is TRUE,
--and otherwise blanks out the credit card number completely.
CREATE MASKING POLICY card_number_conditional_mask
    WITH (fraudulent BOOLEAN, pan varchar(16))
    USING (CASE WHEN fraudulent THEN REDACT_CREDIT_CARD(pan)
                ELSE Null
           END);

--Attach the masking policy to the credit_cards/analyst table/role pair.
ATTACH MASKING POLICY card_number_conditional_mask ON credit_cards (credit_card_number)
 USING (is_fraud, credit_card_number)
 TO ROLE analyst PRIORITY 100;
```

Tampilan sistem penyembunyian data dinamis

Pengguna super, pengguna dengan `sys:operator` peran, dan pengguna dengan izin TABEL SISTEM AKSES dapat mengakses tampilan DDM-related sistem berikut.

- SVV_MASKING_POLICY

  Gunakan SVV_MASKING_POLICY untuk melihat semua kebijakan masking yang dibuat pada cluster atau workgroup.
- KEBIJAKAN PENYAMARAN SVV_TERLAMPIR

  Gunakan SVV_ATTACHED_MASKING_POLICY untuk melihat semua hubungan dan pengguna atau peran dengan kebijakan yang dilampirkan pada database yang saat ini terhubung.
- LOG KEBIJAKAN SYS_APLIKASI_MASKING_MASKING_

  Gunakan SYS_APPLIED_MASKING_POLICY_LOG untuk melacak penerapan kebijakan masking pada kueri yang mereferensikan hubungan. DDM-protected

Berikut ini adalah beberapa contoh informasi yang dapat Anda temukan menggunakan tampilan sistem.

```
--Select all policies associated with specific users, as opposed to roles
```

```
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE grantee_type = 'user';

--Select all policies attached to a specific user
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE grantee = 'target_grantee_name'

--Select all policies attached to a given table
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE table_name = 'target_table_name'
      AND schema_name = 'target_schema_name';

--Select the highest priority policy attachment for a given role
SELECT samp.policy_name,
       samp.priority,
       samp.grantee,
       smp.policy_expression
FROM svv_masking_policy AS smp
JOIN svv_attached_masking_policy AS samp
    ON samp.policy_name = smp.policy_name
WHERE
    samp.grantee_type = 'role' AND
    samp.policy_name = mask_get_policy_for_role_on_column(
        'target_schema_name',
        'target_table_name',
        'target_column_name',
        'target_role_name')
ORDER BY samp.priority desc
LIMIT 1;

--See which policy a specific user will see on a specific column in a given relation
SELECT samp.policy_name,
```

```
       samp.priority,
       samp.grantee,
       smp.policy_expression
FROM svv_masking_policy AS smp
JOIN svv_attached_masking_policy AS samp
    ON samp.policy_name = smp.policy_name
WHERE
    samp.grantee_type = 'role' AND
    samp.policy_name = mask_get_policy_for_user_on_column(
        'target_schema_name',
        'target_table_name',
        'target_column_name',
        'target_user_name')
ORDER BY samp.priority desc;

 --Select all policies attached to a given relation.
SELECT policy_name,
schema_name,
relation_name,
database_name
FROM sys_applied_masking_policy_log
WHERE relation_name = 'relation_name'
AND schema_name = 'schema_name';
```

Pertimbangan saat menggunakan penyembunyian data dinamis

Saat menggunakan penyembunyian data dinamis, pertimbangkan hal berikut:

- Saat menanyakan objek yang dibuat dari tabel, seperti tampilan, pengguna akan melihat hasil berdasarkan kebijakan masking mereka sendiri, bukan kebijakan pengguna yang membuat objek. Misalnya, pengguna dengan peran analis yang menanyakan tampilan yang dibuat oleh secadmin akan melihat hasil dengan kebijakan masking yang dilampirkan ke peran analis.
- Untuk mencegah perintah EXPLAIN berpotensi mengekspos filter kebijakan masking sensitif, hanya pengguna dengan izin SYS_EXPLAIN_DDM yang dapat melihat kebijakan masking yang diterapkan dalam output EXPLAIN. Pengguna tidak memiliki izin SYS_EXPLAIN_DDM secara default.

  Berikut ini adalah sintaks untuk memberikan izin ke peran.

  ```
  GRANT EXPLAIN MASKING TO ROLE rolename
  ```

Untuk informasi selengkapnya tentang perintah EXPLAIN, lihatEXPLAIN.

- Pengguna dengan peran yang berbeda dapat melihat hasil yang berbeda berdasarkan kondisi filter atau kondisi gabungan yang digunakan. Misalnya, menjalankan perintah SELECT pada tabel menggunakan nilai kolom tertentu akan gagal jika pengguna yang menjalankan perintah memiliki kebijakan masking yang diterapkan yang mengaburkan kolom tersebut.
- Kebijakan DDM harus diterapkan sebelum operasi predikat, atau proyeksi. Kebijakan masking dapat mencakup hal-hal berikut:
  - Operasi konstan biaya rendah seperti mengubah nilai menjadi nol
  - Operasi biaya moderat seperti hashing HMAC
  - Operasi berbiaya tinggi seperti panggilan ke fungsi Lambda eksternal yang ditentukan pengguna

  Karena itu, sebaiknya gunakan ekspresi masking sederhana bila memungkinkan.
- Anda dapat menggunakan kebijakan DDM untuk peran dengan kebijakan keamanan tingkat baris, tetapi perhatikan bahwa kebijakan RLS diterapkan sebelum DDM. Ekspresi penyembunyian data dinamis tidak akan dapat membaca baris yang dilindungi oleh RLS. Untuk informasi lebih lanjut tentang RLS, lihatRow-level keamanan.
- Saat menggunakan MENYALIN perintah untuk menyalin dari parket ke tabel target yang dilindungi, Anda harus secara eksplisit menentukan kolom dalam pernyataan COPY. Untuk informasi selengkapnya tentang memetakan kolom dengan COPY, lihatOpsi pemetaan kolom.
- Kebijakan DDM tidak dapat dilampirkan ke relasi berikut:
  - Tabel sistem dan katalog
  - Tabel eksternal
  - Tabel berbagi data
  - Cross-DB hubungan
  - Tabel sementara
  - Kueri yang berkorelasi
- Kebijakan DDM dapat berisi tabel pencarian. Tabel pencarian dapat hadir dalam klausa USING. Jenis relasi berikut tidak dapat digunakan sebagai tabel pencarian:
  - Tabel sistem dan katalog
  - Tabel eksternal
  - Tabel berbagi data
  - Tampilan, tampilan yang terwujud, dan tampilan yang mengikat terlambat

- Cross-DB hubungan
- Tabel sementara
- Kueri yang berkorelasi

Berikut ini adalah contoh melampirkan kebijakan masking ke tabel pencarian.

```
--Create a masking policy referencing a lookup table
CREATE MASKING POLICY lookup_mask_credit_card WITH (credit_card TEXT) USING (
  CASE
    WHEN
      credit_card IN (SELECT credit_card_lookup FROM credit_cards_lookup)
    THEN '000000XXXX0000'
    ELSE REDACT_CREDIT_CARD(credit_card)
    END
  );

--Provides access to the lookup table via a policy attached to a role
GRANT SELECT ON TABLE credit_cards_lookup TO MASKING POLICY lookup_mask_credit_card;
```

- Anda tidak dapat melampirkan kebijakan masking yang akan menghasilkan output yang tidak kompatibel dengan jenis dan ukuran kolom target. Misalnya, Anda tidak dapat melampirkan kebijakan masking yang mengeluarkan string panjang 12 karakter ke kolom VARCHAR (10). Amazon Redshift mendukung pengecualian berikut:
  - Kebijakan masking dengan tipe input INTN dapat dilampirkan ke kebijakan dengan ukuran INTM selama M < N. Misalnya, kebijakan input BIGINT (INT8) dapat dilampirkan ke kolom smallint (INT4).
  - Kebijakan masking dengan tipe input NUMERIC atau DECIMAL selalu dapat dilampirkan ke kolom FLOAT.
- Anda tidak dapat menggunakan kebijakan DDM dengan berbagi data. Jika pembuat data datashare melampirkan kebijakan DDM ke tabel di datashare, tabel menjadi tidak dapat diakses oleh pengguna dari konsumen data yang mencoba menanyakan tabel. Mencoba menambahkan relasi ke datashare gagal pada cluster sisi produsen atau namespace dengan kesalahan berikut:

```
<ddm_protected_relation> or a relation dependent on it is protected by a masking
 policy and cannot be added to a datashare
```

Jika Anda melampirkan kebijakan masking ke relasi di sisi produsen dan relasi tersebut sudah disertakan dalam datashare, mencoba menanyakan relasi di sisi konsumen gagal dengan kesalahan berikut:

```
cross-cluster query of the masked relation <ddm_protected_relation> is not supported.
```

Anda dapat mematikan DDM untuk berbagi data menggunakan perintah ALTER TABLE dengan parameter MASKING OFF FOR DATASHARES. Untuk informasi selengkapnya, lihat ALTER TABLE.

- Anda tidak dapat menanyakan relasi yang telah melampirkan kebijakan DDM jika nilai Anda untuk salah satu opsi konfigurasi berikut tidak cocok dengan nilai default sesi:
  - `enable_case_sensitive_super_attribute`
  - `enable_case_sensitive_identifier`
  - `downcase_delimited_identifier`

  Pertimbangkan untuk mengatur ulang opsi konfigurasi sesi Anda jika Anda mencoba menanyakan relasi dengan kebijakan DDM terlampir dan melihat pesan “Relasi yang dilindungi DDM tidak mendukung konfigurasi tingkat sesi pada sensitivitas huruf besar-kecil yang berbeda dari nilai defaultnya.”
- Jika cluster yang disediakan atau namespace tanpa server memiliki kebijakan penyembunyian data dinamis, perintah berikut diblokir untuk pengguna biasa:

  ```
  ALTER <current_user> SET enable_case_sensitive_super_attribute/
  enable_case_sensitive_identifier/downcase_delimited_identifier
  ```

  Saat membuat kebijakan DDM, sebaiknya ubah setelan opsi konfigurasi default untuk pengguna biasa agar sesuai dengan setelan opsi konfigurasi sesi pada saat kebijakan dibuat. Pengguna super dan pengguna dengan hak istimewa ALTER USER dapat melakukan ini dengan menggunakan pengaturan grup parameter atau perintah ALTER USER. Untuk informasi tentang grup parameter, lihat grup parameter Amazon Redshift di Panduan Manajemen Amazon Redshift. Untuk informasi tentang perintah ALTER USER, lihatALTER USER.
- Tampilan dan tampilan yang terlambat mengikat dengan kebijakan DDM terlampir tidak dapat diganti oleh pengguna biasa menggunakan perintah. CREATE VIEW Untuk mengganti tampilan atau LBV dengan kebijakan DDM, pertama-tama lepaskan kebijakan DDM yang dilampirkan padanya, ganti tampilan atau LBV, dan lampirkan kembali kebijakan. Pengguna super dan

pengguna dengan `sys:secadmin` izin dapat menggunakan CREATE VIEW pada tampilan atau LBV dengan kebijakan DDM tanpa melepaskan kebijakan tersebut.

- Tampilan dengan kebijakan DDM terlampir tidak dapat mereferensikan tabel dan tampilan sistem. Late-bindingtampilan dapat mereferensikan tabel sistem dan tampilan.
- Late-binding tampilan dengan kebijakan DDM terlampir tidak dapat mereferensikan data bersarang di danau data, seperti dokumen JSON.
- Late-binding tampilan tidak dapat memiliki kebijakan DDM yang dilampirkan jika tampilan pengikatan terlambat tersebut direferensikan oleh tampilan apa pun.
- Kebijakan DDM yang dilampirkan pada tampilan pengikatan terlambat dilampirkan dengan nama kolom. Pada saat kueri, Amazon Redshift memvalidasi bahwa semua kebijakan masking yang dilampirkan ke tampilan pengikatan terlambat telah berhasil diterapkan, dan bahwa jenis kolom keluaran tampilan pengikatan terlambat cocok dengan tipe dalam kebijakan masking terlampir. Jika validasi gagal, Amazon Redshift mengembalikan kesalahan untuk kueri.
- Anda dapat menggunakan variabel konteks sesi yang disesuaikan saat membuat kebijakan DDM. Contoh berikut menetapkan variabel konteks sesi untuk kebijakan DDM.

```
-- Set a customized context variable.
SELECT set_config('app.city', 'XXXX', FALSE);

-- Create a MASKING policy using current_setting() to get the value of a customized
 context variable.
CREATE MASKING POLICY city_mask
WITH (city VARCHAR(30))
USING (current_setting('app.city')::VARCHAR(30));

-- Attach the policy on the target table to one or more roles.
ATTACH MASKING POLICY city_mask
ON tickit_users_redshift(city)
TO ROLE analyst, ROLE dbadmin;
```

Untuk detail tentang cara mengatur dan mengambil variabel konteks sesi yang disesuaikan, bukaSET,SETEL_KONFIGURASI,MENUNJUKKAN,SETELINGAN_SAAT INI, danMENGATUR ULANG. Untuk informasi lebih lanjut tentang memodifikasi konfigurasi server secara umum, bukaMemodifikasi konfigurasi server.

> **Important**
>
> Saat menggunakan variabel konteks sesi dalam kebijakan DDM, kebijakan keamanan bergantung pada pengguna atau peran yang memanggil kebijakan. Berhati-hatilah untuk menghindari kerentanan keamanan saat menggunakan variabel konteks sesi dalam kebijakan DDM.

Contoh penyembunyian data dinamis ujung ke ujung

Berikut ini adalah contoh ujung ke ujung yang menunjukkan bagaimana Anda dapat membuat dan melampirkan kebijakan masking ke kolom. Kebijakan ini memungkinkan pengguna mengakses kolom dan melihat nilai yang berbeda, tergantung pada tingkat kebingungan dalam kebijakan yang dilampirkan pada peran mereka. Anda harus menjadi superuser atau memiliki `sys:secadmin` peran untuk menjalankan contoh ini.

Membuat kebijakan masking

Pertama, buat tabel dan isi dengan nilai kartu kredit.

```
--create the table
CREATE TABLE credit_cards (
  customer_id INT,
  credit_card TEXT
);

--populate the table with sample values
INSERT INTO credit_cards
VALUES
  (100, '4532993817514842'),
  (100, '4716002041425888'),
  (102, '5243112427642649'),
  (102, '6011720771834675'),
  (102, '6011378662059710'),
  (103, '373611968625635')
;

--run GRANT to grant permission to use the SELECT statement on the table
GRANT SELECT ON credit_cards TO PUBLIC;
```

```
--create two users
CREATE USER regular_user WITH PASSWORD '1234Test!';

CREATE USER analytics_user WITH PASSWORD '1234Test!';

--create the analytics_role role and grant it to analytics_user
--regular_user does not have a role
CREATE ROLE analytics_role;

GRANT ROLE analytics_role TO analytics_user;
```

Selanjutnya, buat kebijakan masking untuk diterapkan ke peran analitik.

```
--create a masking policy that fully masks the credit card number
CREATE MASKING POLICY mask_credit_card_full
WITH (credit_card VARCHAR(256))
USING ('000000XXXX0000'::TEXT);

--create a user-defined function that partially obfuscates credit card data
CREATE FUNCTION REDACT_CREDIT_CARD (credit_card TEXT)
RETURNS TEXT IMMUTABLE
AS $$
    import re
    regexp = re.compile("^([0-9]{6})[0-9]{5,6}([0-9]{4})")

    match = regexp.search(credit_card)
    if match != None:
        first = match.group(1)
        last = match.group(2)
    else:
        first = "000000"
        last = "0000"

    return "{}XXXXX{}".format(first, last)
$$ LANGUAGE plpythonu;

--create a masking policy that applies the REDACT_CREDIT_CARD function
CREATE MASKING POLICY mask_credit_card_partial
WITH (credit_card VARCHAR(256))
USING (REDACT_CREDIT_CARD(credit_card));

--confirm the masking policies using the associated system views
SELECT * FROM svv_masking_policy;
```

```
SELECT * FROM svv_attached_masking_policy;
```

Melampirkan kebijakan masking

Lampirkan kebijakan masking ke tabel kartu kredit.

```
--attach mask_credit_card_full to the credit card table as the default policy
--all users will see this masking policy unless a higher priority masking policy is
 attached to them or their role
ATTACH MASKING POLICY mask_credit_card_full
ON credit_cards(credit_card)
TO PUBLIC;

--attach mask_credit_card_partial to the analytics role
--users with the analytics role can see partial credit card information
ATTACH MASKING POLICY mask_credit_card_partial
ON credit_cards(credit_card)
TO ROLE analytics_role
PRIORITY 10;

--confirm the masking policies are applied to the table and role in the associated
 system view
SELECT * FROM svv_attached_masking_policy;

--confirm the full masking policy is in place for normal users by selecting from the
 credit card table as regular_user
SET SESSION AUTHORIZATION regular_user;

SELECT * FROM credit_cards;

--confirm the partial masking policy is in place for users with the analytics role by
 selecting from the credit card table as analytics_user
SET SESSION AUTHORIZATION analytics_user;

SELECT * FROM credit_cards;
```

Mengubah kebijakan masking

Bagian berikut menunjukkan cara mengubah kebijakan penyembunyian data dinamis.

```
--reset session authorization to the default
RESET SESSION AUTHORIZATION;
```

```
--alter the mask_credit_card_full policy
ALTER MASKING POLICY mask_credit_card_full
USING ('00000000000000'::TEXT);

--confirm the full masking policy is in place after altering the policy, and that
 results are altered from '000000XXXX0000' to '00000000000000'
SELECT * FROM credit_cards;
```

Melepaskan dan menghapus kebijakan masking

Bagian berikut menunjukkan cara melepaskan dan menghapus kebijakan masking dengan menghapus semua kebijakan penyembunyian data dinamis dari tabel.

```
--reset session authorization to the default
RESET SESSION AUTHORIZATION;

--detach both masking policies from the credit_cards table
DETACH MASKING POLICY mask_credit_card_full
ON credit_cards(credit_card)
FROM PUBLIC;

DETACH MASKING POLICY mask_credit_card_partial
ON credit_cards(credit_card)
FROM ROLE analytics_role;

--drop both masking policies
DROP MASKING POLICY mask_credit_card_full;

DROP MASKING POLICY mask_credit_card_partial;
```

Izin terlingkup

Izin cakupan memungkinkan Anda memberikan izin kepada pengguna atau peran pada semua objek dari jenis dalam database atau skema. Pengguna dan peran dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema.

Anda dapat melihat cakupan izin cakupan tingkat database di. SVV_DATABASE_HAK ISTIMEWA Anda dapat melihat cakupan izin cakupan tingkat skema di. SVV_SCHEMA_HAK ISTIMEWA

Untuk informasi selengkapnya tentang menerapkan izin bercakupan, lihat HIBAH danMENCABUT.

Pertimbangan untuk menggunakan izin terlingkup

Saat menggunakan izin bercakupan, pertimbangkan hal berikut:

- Anda dapat menggunakan izin cakupan untuk memberikan atau mencabut izin pada database atau cakupan skema ke atau dari pengguna atau peran tertentu.
- Anda tidak dapat memberikan izin cakupan ke grup pengguna.
- Memberikan atau mencabut izin cakupan mengubah izin untuk semua objek saat ini dan yang akan datang dalam cakupan.
- Izin cakupan dan izin tingkat objek beroperasi secara independen satu sama lain. Misalnya, pengguna akan mempertahankan izin pada tabel dalam kedua kasus berikut.
  - Pengguna diberikan izin select pada tabel schema1.table1 dan SELECT bercakupan pada schema1. Pengguna kemudian mencabut SELECT untuk semua tabel dalam skema schema1. Pengguna mempertahankan SELECT pada schema1.table1.
  - Pengguna diberikan izin select pada tabel schema1.table1 dan SELECT bercakupan pada schema1. Pengguna kemudian mencabut SELECT untuk schema1.table1. Pengguna mempertahankan SELECT pada schema1.table1.
- Untuk memberikan atau mencabut izin bercakupan, Anda harus memenuhi salah satu kriteria berikut:
  - Pengguna super.
  - Pengguna dengan opsi hibah untuk izin itu. Untuk informasi lebih lanjut tentang opsi hibah, buka parameter WITH GRANT OPTION diHIBAH.
- Izin cakupan hanya dapat diberikan atau dicabut dari objek untuk database yang terhubung, atau dari database yang diimpor dari datashare.
- Anda dapat menggunakan izin cakupan untuk mengatur izin default pada database yang dibuat dari datashare. Pengguna datashare sisi konsumen yang diberikan izin cakupan pada database bersama akan secara otomatis mendapatkan izin tersebut untuk objek baru apa pun yang ditambahkan ke datashare di sisi produsen.
- Produsen dapat memberikan izin cakupan pada objek dalam skema ke datashare. (pratinjau)

Referensi SQL

Dengan Amazon Redshift, Anda dapat memanfaatkan SQL untuk secara efisien menanyakan dan menganalisis sejumlah besar data yang disimpan di gudang data Anda. Referensi SQL mencakup sintaks dan penggunaan perintah SQL, tipe data, fungsi, operator, dan lainnya, memberdayakan Anda untuk mengekstrak wawasan dan membuat keputusan berbasis data. Anda dapat berkonsultasi dengan referensi ini untuk menulis kueri yang dioptimalkan, membuat dan mengelola objek database, dan melakukan transformasi data yang kompleks. Referensi berikut memberikan rincian komprehensif tentang penggunaan SQL dalam lingkungan Amazon Redshift untuk memenuhi persyaratan analisis data Anda.

Topik

- Amazon Redshift SQL
- Menggunakan SQL
- Perintah SQL
- Referensi fungsi SQL
- Kata yang dicadangkan

Amazon Redshift SQL

Topik

- Fungsi SQL didukung pada node pemimpin
- Amazon Redshift dan PostgreSQL

Amazon Redshift dibangun di sekitar SQL standar industri, dengan fungsionalitas tambahan untuk mengelola kumpulan data yang sangat besar dan mendukung analisis dan pelaporan data berkinerja tinggi.

> **Note**
>
> Ukuran maksimum untuk satu pernyataan Amazon Redshift SQL adalah 16 MB.

Fungsi SQL didukung pada node pemimpin

Beberapa kueri Amazon Redshift didistribusikan dan dijalankan pada node komputasi, dan kueri lainnya dijalankan secara eksklusif pada node pemimpin.

Node pemimpin mendistribusikan SQL ke node komputasi setiap kali kueri mereferensikan tabel atau tabel sistem yang dibuat pengguna (tabel dengan awalan STL atau STV dan tampilan sistem dengan awalan SVL atau SVV). Kueri yang hanya mereferensikan tabel katalog (tabel dengan awalan PG, seperti PG_TABLE_DEF, yang berada di node pemimpin) atau yang tidak mereferensikan tabel apa pun, berjalan secara eksklusif pada node pemimpin.

Beberapa fungsi Amazon Redshift SQL hanya didukung pada node pemimpin dan tidak didukung pada node komputasi. Kueri yang menggunakan fungsi leader-node harus berjalan secara eksklusif pada node pemimpin, bukan pada node komputasi, atau akan mengembalikan kesalahan.

Dokumentasi untuk setiap fungsi yang harus dijalankan secara eksklusif pada node pemimpin menyertakan catatan yang menyatakan bahwa fungsi akan mengembalikan kesalahan jika mereferensikan tabel yang ditentukan pengguna atau tabel sistem Amazon Redshift. Lihat Node—hanya fungsi utama untuk daftar fungsi yang berjalan secara eksklusif pada node pemimpin.

Contoh

Contoh berikut menggunakan contoh database TICKIT. Untuk informasi lebih lanjut tentang database sampel, bukaDatabase sampel.

SKEMA SAAT INI

Fungsi CURRENT_SCHEMA adalah fungsi utama leader-node. Dalam contoh ini, query tidak mereferensikan tabel, sehingga berjalan secara eksklusif pada node pemimpin.

```
select current_schema();

current_schema
---------------
public
```

Dalam contoh berikutnya, query mereferensikan tabel katalog sistem, sehingga berjalan secara eksklusif pada node pemimpin.

```
select * from pg_table_def
```

```
where schemaname = current_schema() limit 1;

 schemaname | tablename | column |   type   | encoding | distkey | sortkey | notnull
------------+-----------+--------+----------+----------+---------+---------+---------
 public     | category  | catid  | smallint | none     | t       |       1 | t
```

Dalam contoh berikutnya, kueri mereferensikan tabel sistem Amazon Redshift yang berada di node komputasi, sehingga mengembalikan kesalahan.

```
select current_schema(), userid from users;

INFO:  Function "current_schema()" not supported.
ERROR:  Specified types or functions (one per INFO message) not supported on Amazon
 Redshift tables.
```

SUBSTIR

SUBSTR juga merupakan fungsi pemimpin-node saja. Dalam contoh berikut, query berjalan eksklusif pada node pemimpin karena tidak mereferensikan tabel.

```
SELECT SUBSTR('amazon', 5);

+--------+
| substr |
+--------+
| on     |
+--------+
```

Dalam contoh berikut, query mereferensikan tabel yang berada di node komputasi. Ini menghasilkan kesalahan.

```
SELECT SUBSTR(catdesc, 1) FROM category LIMIT 1;

ERROR: SUBSTR() function is not supported (Hint: use SUBSTRING instead)
```

Untuk berhasil menjalankan kueri sebelumnya, gunakan SUBSTRING.

```
SELECT SUBSTRING(catdesc, 1) FROM category LIMIT 1;

+--------------------------------+
|           substring            |
```

```
+---------------------------------+
| National Basketball Association |
+---------------------------------+
```

FAKTORIAL ()

FACTORIAL () adalah fungsi pemimpin-node saja. Dalam contoh berikut, query berjalan eksklusif pada node pemimpin karena tidak mereferensikan tabel.

```
SELECT FACTORIAL(5);

 factorial
-------------
 120
```

Dalam contoh berikut, query mereferensikan tabel yang berada di node komputasi. Ini menghasilkan kesalahan saat dijalankan menggunakan editor kueri v2.

```
create table t(a int);
insert into t values (5);
select factorial(a) from t;

ERROR: Specified types or functions (one per INFO message) not supported on Redshift
 tables.
Info: Function "factorial(bigint)" not supported.
```

Amazon Redshift dan PostgreSQL

Topik

- Amazon Redshift dan PostgreSQL JDBC dan ODBC
- Fitur yang diimplementasikan secara berbeda
- Fitur PostgreSQL yang tidak didukung
- Jenis data PostgreSQL tidak didukung
- Fungsi PostgreSQL yang tidak didukung

Amazon Redshift didasarkan pada PostgreSQL. Amazon Redshift dan PostgreSQL memiliki sejumlah perbedaan yang sangat penting yang harus Anda ketahui saat merancang dan mengembangkan aplikasi gudang data Anda.

Amazon Redshift dirancang khusus untuk aplikasi pemrosesan analitik online (OLAP) dan intelijen bisnis (BI), yang memerlukan kueri kompleks terhadap kumpulan data besar. Karena memenuhi persyaratan yang sangat berbeda, skema penyimpanan data khusus dan mesin eksekusi kueri yang digunakan Amazon Redshift benar-benar berbeda dari implementasi PostgreSQL. Misalnya, di mana aplikasi pemrosesan transaksi online (OLTP) biasanya menyimpan data dalam baris, Amazon Redshift menyimpan data dalam kolom, menggunakan pengkodean kompresi data khusus untuk penggunaan memori dan disk yang optimal. I/O Beberapa fitur PostgreSQL yang cocok untuk pemrosesan OLTP skala kecil, seperti indeks sekunder dan operasi manipulasi data baris tunggal yang efisien, telah dihilangkan untuk meningkatkan kinerja.

Lihat Arsitektur Amazon Redshift untuk penjelasan rinci tentang arsitektur sistem gudang data Amazon Redshift.

PostgreSQL 9.x menyertakan beberapa fitur yang tidak didukung di Amazon Redshift. Selain itu, ada perbedaan penting antara Amazon Redshift SQL dan PostgreSQL yang harus Anda ketahui. Bagian ini menyoroti perbedaan antara Amazon Redshift dan PostgreSQL dan memberikan panduan untuk mengembangkan gudang data yang memanfaatkan sepenuhnya implementasi Amazon Redshift SQL.

Amazon Redshift dan PostgreSQL JDBC dan ODBC

Karena Amazon Redshift didasarkan pada PostgreSQL, kami sebelumnya merekomendasikan penggunaan driver JDBC4 Postgresql versi 8.4.703 dan driver PSQloDBC versi 9.x. Jika saat ini Anda menggunakan driver tersebut, kami sarankan untuk pindah ke driver khusus Amazon Redshift baru di masa mendatang. Untuk informasi selengkapnya tentang driver dan mengonfigurasi koneksi, lihat Driver JDBC dan ODBC untuk Amazon Redshift di Panduan Manajemen Amazon Redshift.

Untuk menghindari kesalahan kehabisan memori sisi klien saat mengambil kumpulan data besar menggunakan JDBC, Anda dapat mengaktifkan klien untuk mengambil data dalam batch dengan menyetel parameter ukuran pengambilan JDBC. Untuk informasi selengkapnya, lihat Mengatur parameter ukuran pengambilan JDBC.

Amazon Redshift tidak mengenali parameter JDBC MaxRows. Sebagai gantinya, tentukan LIMIT klausa untuk membatasi kumpulan hasil. Anda juga dapat menggunakan OFFSET klausa untuk melompat ke titik awal tertentu dalam kumpulan hasil.

Fitur yang diimplementasikan secara berbeda

Banyak elemen bahasa Amazon Redshift SQL memiliki karakteristik kinerja yang berbeda dan menggunakan sintaks dan semantik dan yang sangat berbeda dari implementasi PostgreSQL yang setara.

> **Important**
>
> Jangan berasumsi bahwa semantik elemen yang dimiliki Amazon Redshift dan PostgreSQL sama. Pastikan untuk berkonsultasi dengan Panduan Pengembang Amazon Redshift Perintah SQL untuk memahami perbedaan yang seringkali halus.

Salah satu contoh khususnya adalah VAKUM perintah, yang digunakan untuk membersihkan dan mengatur ulang tabel. VACUUM berfungsi secara berbeda dan menggunakan kumpulan parameter yang berbeda dari versi PostgreSQL. Lihat Meja penyedot debu untuk informasi selengkapnya tentang menggunakan VACUUM di Amazon Redshift.

Seringkali, fitur dan alat manajemen basis data dan administrasi berbeda juga. Misalnya, Amazon Redshift memelihara satu set tabel sistem dan tampilan yang memberikan informasi tentang bagaimana sistem berfungsi. Untuk informasi selengkapnya, lihat Tampilan pemantauan SYS.

Daftar berikut mencakup beberapa contoh fitur SQL yang diimplementasikan secara berbeda di Amazon Redshift.

- CREATE TABLE

  Amazon Redshift tidak mendukung tabel, partisi tabel, pewarisan, dan batasan tertentu. Dengan implementasi Amazon Redshift dari CREATE TABLE, Anda dapat menentukan algoritma pengurutan dan distribusi untuk tabel untuk mengoptimalkan pemrosesan paralel.

  Amazon Redshift Spectrum mendukung partisi tabel menggunakan perintahCREATE EXTERNAL TABLE.
- ALTER TABLE

  Hanya sebagian dari tindakan ALTER COLUMN yang didukung.

  ADD COLUMN mendukung penambahan hanya satu kolom di setiap pernyataan ALTER TABLE.
- MENYALIN

Perintah Amazon Redshift COPY sangat khusus untuk memungkinkan pemuatan data dari bucket Amazon S3 dan tabel Amazon DynamoDB dan untuk memfasilitasi kompresi otomatis. Lihat Memuat data di Amazon Redshift bagian dan referensi perintah COPY untuk detailnya.

- VAKUM

  Parameter untuk VACUUM sama sekali berbeda. Misalnya, operasi VACUUM default di PostgreSQL hanya mengambil kembali ruang dan membuatnya tersedia untuk digunakan kembali; namun, operasi VACUUM default di Amazon Redshift adalah VACUUM FULL, yang mengambil kembali ruang disk dan mengambil semua baris.
- Spasi akhir dalam nilai VARCHAR diabaikan ketika nilai string dibandingkan. Untuk informasi selengkapnya, lihat Signifikansi blanko yang tertinggal.

Fitur PostgreSQL yang tidak didukung

Fitur PostgreSQL ini tidak didukung di Amazon Redshift.

> **Important**
>
> Jangan berasumsi bahwa semantik elemen yang dimiliki Amazon Redshift dan PostgreSQL sama. Pastikan untuk berkonsultasi dengan Panduan Pengembang Amazon Redshift Perintah SQL untuk memahami perbedaan yang seringkali halus.

- Alat kueri psql tidak didukung. Kli en Amazon Redshift RSQL didukung.
- Partisi tabel (partisi rentang dan daftar)
- Ruang tabel
- Batasan
  - Unik
  - Kunci asing
  - Kunci primer
  - Batasan pemeriksaan
  - Kendala pengecualian

  Kendala unik, kunci utama, dan kunci asing diizinkan, tetapi hanya bersifat informasi. Mereka tidak ditegakkan oleh sistem, tetapi mereka digunakan oleh perencana kueri.

- Warisan
- Kolom sistem PostgreSQL

  Amazon Redshift SQL tidak secara implisit mendefinisikan kolom sistem. Namun, nama kolom sistem PostgreSQL berikut tidak dapat digunakan sebagai nama kolom yang ditentukan pengguna:`oid`,,,`tableoid`, `xmin` `cminxmax`, `cmax` dan. `ctid` Untuk informasi selengkapnya, lihat https://www.postgresql.org/docs/8.0/static/ddl-system-columns.html.
- Indeks
- Klausa NULLS dalam fungsi Window
- Kolasi

  Amazon Redshift tidak mendukung urutan kolasi khusus lokal atau yang ditentukan pengguna. Lihat Urutan kolasi.
- Ekspresi nilai
  - Ekspresi berlangganan
  - Konstruktor array
  - Konstruktor baris
- Pemicu
- Manajemen Data Eksternal (SQL/MED)
- Fungsi tabel
- Daftar NILAI digunakan sebagai tabel konstan
- Urutan
- Pencarian teks lengkap
- Perizinan ATURAN dan TRIGGER.

  Amazon Redshift memberikan atau mencabut izin ini saat Anda menjalankan GRANT ALL atau CABUT ALL, tetapi ada atau tidaknya izin RULE dan TRIGGER tidak memengaruhi izin akses penerima hibah dengan cara apa pun.

Jenis data PostgreSQL tidak didukung

Umumnya, jika kueri mencoba menggunakan tipe data yang tidak didukung, termasuk cast eksplisit atau implisit, itu akan mengembalikan kesalahan. Namun, beberapa kueri yang menggunakan tipe

data yang tidak didukung akan berjalan pada node pemimpin tetapi tidak pada node komputasi. Lihat Fungsi SQL didukung pada node pemimpin.

Untuk daftar tipe data yang didukung, lihatJenis Data.

Jenis data PostgreSQL ini tidak didukung di Amazon Redshift.

- Array
- SEDIKIT, SEDIKIT BERVARIASI
- BYTEA
- Jenis Komposit
- Jenis yang Dicantumkan
- Jenis Geometris (Implementasi tipe geometris Amazon Redshift berbeda dari PostgreSQL)
- HSTORE
- JSON
- Jenis Alamat Jaringan
- Jenis Numerik
  - SERIAL, SERIAL BESAR, SERIAL KECIL
  - MONEY
- Jenis Pengidentifikasi Objek
- Pseudo-Types
- Jenis Rentang
- Jenis Karakter Khusus
  - “char” — Tipe internal byte tunggal (di mana tipe data bernama char tertutup dalam tanda kutip).
  - name — Jenis internal untuk nama objek.

  Untuk informasi selengkapnya tentang jenis ini, lihat Jenis Karakter Khusus dalam dokumentasi PostgreSQL.
- Jenis Pencarian Teks
- TXID_SNAPSHOT
- UUID
- XML

Fungsi PostgreSQL yang tidak didukung

Banyak fungsi yang tidak dikecualikan memiliki semantik atau penggunaan yang berbeda. Misalnya, beberapa fungsi yang didukung hanya akan berjalan pada node pemimpin. Juga, beberapa fungsi yang tidak didukung tidak akan mengembalikan kesalahan saat dijalankan pada node pemimpin. Fakta bahwa fungsi-fungsi ini tidak mengembalikan kesalahan dalam beberapa kasus tidak boleh dianggap untuk menunjukkan bahwa fungsi tersebut didukung oleh Amazon Redshift.

> **Important**
>
> Jangan berasumsi bahwa semantik elemen yang dimiliki Amazon Redshift dan PostgreSQL sama. Pastikan untuk membaca Panduan Pengembang Database Amazon Redshift Perintah SQL untuk memahami perbedaan yang seringkali tidak jelas.

Untuk informasi selengkapnya, lihat Fungsi SQL didukung pada node pemimpin.

Fungsi PostgreSQL ini tidak didukung di Amazon Redshift.

- Akses fungsi penyelidikan hak istimewa
- Fungsi kunci penasihat
- Fungsi agregat
  - STRING_AGG ()
  - ARRAY_AGG ()
  - SETIAP ()
  - XML_AGG ()
  - RUN ()
  - COVAR_POP ()
  - COVAR_SAMP ()
  - REGR_AVGX (), REGR_AVGY ()
  - REGR_COUNT ()
  - REGR_INTERCEPT ()
  - REGR_R2 ()
  - REGR_LERENG ()
  - REGR_SXX (), REGR_SXY (), REGR_SYY ()

- Fungsi dan operator array
- Fungsi kontrol cadangan
- Fungsi informasi komentar
- Fungsi lokasi objek database
- Fungsi ukuran objek database
- Date/Time fungsi dan operator
  - TIMESTAMP JAM ()
  - JUSTIFY_DAYS (), JUSTIFY_JAM (), JUSTIFY_INTERVAL ()
  - PG_TIDUR ()
  - STEMPEL WAKTU TRANSAKSI_()
- Fungsi dukungan ENUM
- Fungsi geometris dan operator
- Fungsi akses file generik
- IS DISTINCT FROM
- Fungsi alamat jaringan dan operator
- Fungsi matematika
  - DIVISI ()
  - SETSEED ()
  - LEBAR_BUCKET ()
- Atur fungsi pengembalian
  - GENERATE_SERIES ()
  - GENERATE_SUBSCRIPTS ()
- Fungsi rentang dan operator
- Fungsi kontrol pemulihan
- Fungsi informasi pemulihan
- ROLLBACK TO SAVEPOINT fungsi
- Fungsi penyelidikan visibilitas skema
- Fungsi pensinyalan server
- Fungsi sinkronisasi snapshot
- Fungsi manipulasi urutan

- Fungsi string
  - BIT_PANJANG ()
  - HAMPARAN ()
  - MENGUBAH (), CONVERT_FROM (), CONVERT_TO ()
  - MENYANDIKAN ()
  - FORMAT ()
  - QUOTE_NULLABLE ()
  - REGEXP_MATCH ()
  - REGEXP_SPLIT_KE_ARRAY ()
  - REGEXP_SPLIT_TO_TABLE ()
- Fungsi informasi katalog sistem
- Fungsi informasi sistem
  - KATALOG_SAAT INI_KUERI SAAT INI ()
  - INET_CLIENT_ADDR ()
  - INET_PORT_KLIEN ()
  - INET_SERVER_ADDR () INET_SERVER_PORT_()
  - PG_CONF_LOAD_TIME ()
  - PG_IS_SCHEMA_TEMP_LAIN_()
  - PG_LISTENING_CHANNEL ()
  - PG_TIME_SCHEMA_SAYA ()
  - PG_POSTMASTER_START_TIME ()
  - PG_TRIGGER_KEDALAMAN ()
  - TAMPILKAN VERSI ()
- Fungsi dan operator pencarian teks
- Fungsi ID transaksi dan snapshot
- Fungsi pemicu
- Fungsi XML

Menggunakan SQL

Topik

- Konvensi referensi SQL
- Elemen dasar
- Ekspresi
- Ketentuan

Bahasa SQL terdiri dari perintah dan fungsi yang Anda gunakan untuk bekerja dengan database dan objek database. Bahasa ini juga memberlakukan aturan mengenai penggunaan tipe data, ekspresi, dan literal.

Konvensi referensi SQL

Bagian ini menjelaskan konvensi yang digunakan untuk menulis sintaks untuk ekspresi SQL, perintah, dan fungsi yang dijelaskan di bagian referensi SQL.

Karakter	Deskripsi
TOPI	Kata-kata dalam huruf kapital adalah kata kunci.
[ ]	Kurung menunjukkan argumen opsional. Beberapa argumen dalam tanda kurung menunjukkan bahwa Anda dapat memilih sejumlah argumen. Selain itu, argumen dalam tanda kurung pada baris terpisah menunjukkan bahwa parser Amazon Redshift mengharapkan argumen berada dalam urutan yang dicantumkan dalam sintaks. Sebagai contoh, lihat SELECT.
{ }	Tanda kurung menunjukkan bahwa Anda diminta untuk memilih salah satu penjelasan di dalam kurung kurawal.
|	Pipa menunjukkan bahwa Anda dapat memilih di antara argumen.
huruf miring	Kata-kata yang dicetak miring menunjukkan placeholder. Anda harus memasukkan nilai yang sesuai di tempat kata yang dicetak miring.
. . .	Elipsis menunjukkan bahwa Anda dapat mengulangi elemen sebelumnya.
'	Kata-kata dalam tanda kutip tunggal menunjukkan bahwa Anda harus mengetik tanda kutip.

Elemen dasar

Topik

- Nama dan pengidentifikasi
- Literal
- Nulls
- Jenis Data
- Urutan kolasi

Bagian ini mencakup aturan untuk bekerja dengan nama objek database, literal, nol, dan tipe data.

Nama dan pengidentifikasi

Nama mengidentifikasi objek database, termasuk tabel dan kolom, serta pengguna dan kata sandi. Istilah nama dan pengenal dapat digunakan secara bergantian. Ada dua jenis pengidentifikasi, pengidentifikasi standar dan pengidentifikasi dikutip atau dibatasi. Pengidentifikasi hanya harus terdiri dari karakter UTF-8 yang dapat dicetak. Huruf ASCII dalam pengidentifikasi standar dan dibatasi tidak peka huruf besar dan dilipat menjadi huruf kecil dalam database. Dalam hasil kueri, nama kolom dikembalikan sebagai huruf kecil secara default. Untuk mengembalikan nama kolom dalam huruf besar, atur parameter deskripsi_field_name_in_uppercase konfigurasi ke**`true`**.

Pengidentifikasi standar

Pengidentifikasi SQL standar mematuhi seperangkat aturan dan harus:

- Mulailah dengan karakter alfabet byte tunggal ASCII atau karakter garis bawah, atau karakter UTF-8 multibyte sepanjang dua hingga empat byte.
- Karakter selanjutnya dapat berupa karakter alfanumerik byte tunggal ASCII, garis bawah, atau tanda dolar, atau karakter UTF-8 multibyte sepanjang dua hingga empat byte.
- Panjangnya antara 1 dan 127 byte, tidak termasuk tanda kutip untuk pengidentifikasi yang dibatasi.
- Tidak mengandung tanda kutip dan tidak ada spasi.
- Jangan menjadi kata kunci SQL yang dicadangkan.

Pengidentifikasi yang dibatasi

Pengidentifikasi yang dibatasi (juga dikenal sebagai pengenal yang dikutip) dimulai dan diakhiri dengan tanda kutip ganda (“). Jika Anda menggunakan pengenal yang dibatasi, Anda harus menggunakan tanda kutip ganda untuk setiap referensi ke objek itu. Pengidentifikasi dapat berisi karakter standar UTF-8 yang dapat dicetak selain tanda kutip ganda itu sendiri. Oleh karena itu, Anda dapat membuat nama kolom atau tabel yang menyertakan karakter ilegal, seperti spasi atau simbol persen.

Huruf ASCII dalam pengidentifikasi yang dibatasi tidak peka huruf besar/kecil dan dilipat menjadi huruf kecil. Untuk menggunakan tanda kutip ganda dalam string, Anda harus mendahuluinya dengan karakter tanda kutip ganda lainnya.

Case-sensitive pengidentifikasi

Case-sensitive pengidentifikasi (juga dikenal sebagai pengidentifikasi huruf campuran) dapat berisi huruf besar dan huruf kecil. Untuk menggunakan pengidentifikasi peka huruf besar/kecil, Anda dapat mengatur konfigurasi `enable_case_sensitive_identifier` ke`true`. Anda dapat mengatur konfigurasi ini untuk cluster atau untuk sesi. Untuk informasi selengkapnya, lihat Nilai parameter default di Panduan Manajemen Amazon Redshift dan`enable_case_sensitive_identifier`.

Nama kolom sistem

Nama kolom sistem PostgreSQL berikut tidak dapat digunakan sebagai nama kolom di kolom yang ditentukan pengguna. Untuk informasi selengkapnya, lihat https://www.postgresql.org/docs/8.0/static/ddl-system-columns.html.

- `oid`
- `tableoid`
- `xmin`
- `cmin`
- `xmax`
- `cmax`
- `ctid`

Contoh

Tabel ini menunjukkan contoh pengidentifikasi yang dibatasi, output yang dihasilkan, dan diskusi:

Sintaksis	Hasil	Diskusi
“kelompok”	grup	GROUP adalah kata yang dicadangkan, jadi penggunaannya dalam pengenal memerlukan tanda kutip ganda.
“"DI MANA" "”	“dimana”	WHERE juga merupakan kata yang dicadangkan. Untuk memasukkan tanda kutip dalam string, hindari setiap karakter tanda kutip ganda dengan karakter tanda kutip ganda tambahan.
“Nama ini”	nama ini	Tanda kutip ganda diperlukan untuk melestarikan ruang.
“Ini “" APAKAH ITULA" "”	ini “apakah itu”	Tanda kutip yang mengelilingi IS IT masing-masing harus didahului dengan tanda kutip tambahan untuk menjadi bagian dari nama.

Untuk membuat tabel bernama grup dengan kolom bernama ini “apakah itu”:

```
create table "group" (
"This ""IS IT""" char(10));
```

Kueri berikut mengembalikan hasil yang sama:

```
select "This ""IS IT"""
from "group";

this "is it"
--------------
(0 rows)
```

```
select "this ""is it"""
from "group";

this "is it"
--------------
(0 rows)
```

`table.column`Sintaks yang memenuhi syarat berikut ini juga mengembalikan hasil yang sama:

```
select "group"."this ""is it"""
from "group";

this "is it"
--------------
(0 rows)
```

Perintah CREATE TABLE berikut membuat tabel dengan garis miring dalam nama kolom:

```
create table if not exists city_slash_id(
                  "city/id" integer not null,
                  state char(2) not null);
```

Literal

Literal atau konstan adalah nilai data tetap, terdiri dari urutan karakter atau konstanta numerik. Amazon Redshift mendukung beberapa jenis literal, termasuk:

- Literal numerik untuk bilangan integer, desimal, dan floating-point. Untuk informasi selengkapnya, lihat Literal integer dan floating-point.
- Literal karakter, juga disebut sebagai string, string karakter, atau konstanta karakter
- Literal tanggal-waktu dan interval, digunakan dengan tipe data datetime. Untuk informasi selengkapnya, lihat Literal tanggal, waktu, dan stempel waktu dan Jenis data interval dan literal.

Nulls

Jika kolom dalam satu baris hilang, tidak diketahui, atau tidak berlaku, itu adalah nilai nol atau dikatakan mengandung null. Null dapat muncul di bidang tipe data apa pun yang tidak dibatasi oleh kunci primer atau batasan NOT NULL. Null tidak setara dengan nilai nol atau string kosong.

Setiap ekspresi aritmatika yang mengandung nol selalu dievaluasi menjadi null. Semua operator mengembalikan null ketika diberi argumen null atau operan.

Untuk menguji nol, gunakan kondisi perbandingan IS NULL dan IS NOT NULL. Karena null mewakili kekurangan data, null tidak sama atau tidak sama dengan nilai apa pun atau nol lainnya.

Jenis Data

Topik

- Karakter multibyte
- Jenis numerik
- Jenis karakter
- Jenis tanggal-waktu
- Tipe Boolean
- Jenis HLLSKETCH
- Jenis SUPER
- Jenis VARBYTE
- Kompatibilitas dan konversi tipe

Setiap nilai yang disimpan atau diambil Amazon Redshift memiliki tipe data dengan serangkaian properti terkait tetap. Tipe data dideklarasikan saat tabel dibuat. Tipe data membatasi kumpulan nilai yang dapat berisi kolom atau argumen.

Tabel berikut mencantumkan tipe data yang dapat Anda gunakan di tabel Amazon Redshift.

Jenis data	Alias	Deskripsi
SMALLINT	INT2	Bilangan bulat dua byte bertanda
INTEGER	INT, INT4	Bilangan bulat empat byte bertanda
BIGINT	INT8	Bilangan bulat delapan byte bertanda
DECIMAL	NUMERIC	Numerik persis dari presisi yang dapat dipilih
REAL	FLOAT4	Angka floating-point presisi tunggal
DOUBLE PRECISION	FLOAT8, MENGAPUNG	Angka floating-point presisi ganda

Jenis data	Alias	Deskripsi
CHAR	KARAKTER, NCHAR, BPCHAR	Fixed-length string karakter
VARCHAR	KARAKTER BERVARIASI, NVARCHAR, TEKS	Variable-length string karakter dengan batas yang ditentukan pengguna
DATE		Tanggal kalender (tahun, bulan, hari)
TIME	WAKTU TANPA ZONA WAKTU	Waktu hari
TIMEZ	TIME WITH TIME ZONE	Waktu hari dengan zona waktu
TIMESTAMP	STEMPEL WAKTU TANPA ZONA WAKTU	Tanggal dan waktu (tanpa zona waktu)
TIMESTAMPTZ	TIMESTAMP WITH TIME ZONE	Tanggal dan waktu (dengan zona waktu)
INTERVAL TAHUN KE BULAN		Durasi waktu dalam urutan tahun ke bulan
INTERVAL HARI KE DETIK		Durasi waktu dalam hari ke urutan kedua
BOOLEAN	BOOL	Logika Boolean () true/false
HLLSKETSA		Jenis yang digunakan dengan HyperLogLog sketsa.

Jenis data	Alias	Deskripsi
SUPER		Tipe data superset yang mencakup semua jenis skalar Amazon Redshift termasuk tipe kompleks seperti ARRAY dan STRUCTS.
WARBYTE	VARBINARY, BINER BERVARIASI	Variable-length nilai biner
GEOMETRY		Data spasial
GEOGRAPHY		Data spasial

> **Note**
>
> Untuk informasi tentang tipe data yang tidak didukung, seperti “char” (perhatikan bahwa char tertutup dalam tanda kutip), lihatJenis data PostgreSQL tidak didukung.

Karakter multibyte

Tipe data VARCHAR mendukung karakter UTF-8 multibyte hingga maksimal empat byte. Five-byte atau karakter yang lebih panjang tidak didukung. Untuk menghitung ukuran kolom VARCHAR yang berisi karakter multibyte, kalikan jumlah karakter dengan jumlah byte per karakter. Misalnya, jika string memiliki empat karakter Mandarin, dan setiap karakter memiliki panjang tiga byte, maka Anda akan memerlukan kolom VARCHAR (12) untuk menyimpan string.

Tipe data VARCHAR tidak mendukung titik kode yang tidak valid berikut UTF-8:

`0xD800 – 0xDFFF`(Urutan Byte: `ED A0 80` —`ED BF BF`)

Tipe data CHAR tidak mendukung karakter multibyte.

Jenis numerik

Topik

- Jenis bilangan bulat

- Tipe DESIMAL atau NUMERIK
- Catatan tentang menggunakan kolom DECIMAL atau NUMERIC 128-bit
- Floating-Point jenis
- Perhitungan dengan nilai numerik
- Literal integer dan floating-point
- Contoh dengan tipe numerik

Tipe data numerik termasuk bilangan bulat, desimal, dan angka floating-point.

Jenis bilangan bulat

Gunakan tipe data SMALLINT, INTEGER, dan BIGINT untuk menyimpan bilangan bulat dari berbagai rentang. Anda tidak dapat menyimpan nilai di luar rentang yang diizinkan untuk setiap jenis.

Nama	Penyimpanan	Kisaran
SMALLINT atau INT2	2 byte	-32768 sampai +32767
INTEGER, INT, atau INT4	4 byte	-2147483648 sampai +2147483647
BIGINT atau INT8	8 byte	-9223372036854775808 sampai 9223372036854775807

Tipe DESIMAL atau NUMERIK

Gunakan tipe data DECIMAL atau NUMERIC untuk menyimpan nilai dengan presisi yang ditentukan pengguna. Kata kunci DECIMAL dan NUMERIC dapat dipertukarkan. Dalam dokumen ini, desimal adalah istilah yang disukai untuk tipe data ini. Istilah numerik digunakan secara umum untuk merujuk pada tipe data integer, desimal, dan floating-point.

Penyimpanan	Kisaran
Variabel, hingga 128 bit untuk tipe DECIMAL yang tidak terkompresi.	Bilangan bulat bertanda 128-bit dengan presisi hingga 38 digit.

Tentukan kolom DECIMAL dalam tabel dengan menentukan presisi dan skala:

```
decimal(precision, scale)
```

presisi

Jumlah total digit signifikan dalam keseluruhan nilai: jumlah digit di kedua sisi titik desimal. Misalnya, angka tersebut `48.2891` memiliki presisi 6 dan skala 4. Presisi default, jika tidak ditentukan, adalah 18. Presisi maksimum adalah 38.

Jika jumlah digit di sebelah kiri titik desimal dalam nilai input melebihi ketepatan kolom dikurangi skalanya, nilainya tidak dapat disalin ke kolom (atau dimasukkan atau diperbarui). Aturan ini berlaku untuk nilai apa pun yang berada di luar rentang definisi kolom. Misalnya, rentang nilai yang diizinkan untuk `numeric(5,2)` kolom adalah `-999.99` to`999.99`.

skala

Jumlah digit desimal di bagian pecahan nilai, di sebelah kanan titik desimal. Bilangan bulat memiliki skala nol. Dalam spesifikasi kolom, nilai skala harus kurang dari atau sama dengan nilai presisi. Skala default, jika tidak ditentukan, adalah 0. Skala maksimum adalah 37.

Jika skala nilai input yang dimuat ke dalam tabel lebih besar dari skala kolom, nilainya dibulatkan ke skala yang ditentukan. Misalnya, kolom PRICEPAID dalam tabel PENJUALAN adalah kolom DECIMAL (8,2). Jika nilai DECIMAL (8,4) dimasukkan ke dalam kolom PRICEPAID, nilainya dibulatkan ke skala 2.

```
insert into sales
values (0, 8, 1, 1, 2000, 14, 5, 4323.8951, 11.00, null);

select pricepaid, salesid from sales where salesid=0;

pricepaid | salesid
-----------+---------
4323.90 |       0
```

```
(1 row)
```

Namun, hasil pemertian eksplisit nilai yang dipilih dari tabel tidak dibulatkan.

> **Note**
>
> Nilai positif maksimum yang dapat Anda masukkan ke dalam kolom DECIMAL (19,0) adalah `9223372036854775807` (2^{63} -1). Nilai negatif maksimum adalah`-9223372036854775808`. Misalnya, upaya untuk memasukkan nilai `999999999999999999` (19 sembilan) akan menyebabkan kesalahan luapan. Terlepas dari penempatan titik desimal, string terbesar yang dapat diwakili oleh Amazon Redshift sebagai angka DESIMAL adalah`9223372036854775807`. Misalnya, nilai terbesar yang dapat Anda muat ke dalam kolom DECIMAL (19,18) adalah. `9.223372036854775807`
> Aturan ini karena nilai DECIMAL dengan 19 digit presisi signifikan atau lebih sedikit disimpan secara internal sebagai bilangan bulat 8-byte, sedangkan nilai DECIMAL dengan presisi 20 hingga 38 digit signifikan disimpan sebagai bilangan bulat 16-byte.

Catatan tentang menggunakan kolom DECIMAL atau NUMERIC 128-bit

Jangan sewenang-wenang menetapkan presisi maksimum ke kolom DECIMAL kecuali Anda yakin bahwa aplikasi Anda memerlukan presisi itu. Nilai 128-bit menggunakan ruang disk dua kali lebih banyak daripada nilai 64-bit dan dapat memperlambat waktu eksekusi kueri.

Floating-Point jenis

Gunakan tipe data REAL dan DOUBLE PRECISION untuk menyimpan nilai numerik dengan presisi variabel. Jenis ini adalah tipe yang tidak tepat, yang berarti bahwa beberapa nilai disimpan sebagai perkiraan, sehingga menyimpan dan mengembalikan nilai tertentu dapat mengakibatkan sedikit perbedaan. Jika Anda memerlukan penyimpanan dan perhitungan yang tepat (seperti untuk jumlah moneter), gunakan tipe data DECIMAL.

REAL mewakili format titik mengambang presisi tunggal, menurut Standar IEEE 754 untuk Aritmatika B Floating-Point iner. Ini memiliki presisi sekitar 6 digit, dan kisaran sekitar 1 E-37 hingga 1E+37. Anda juga dapat menentukan tipe data ini sebagai FLOAT4.

DOUBLE PRECISION mewakili format titik mengambang presisi ganda, menurut Standar IEEE 754 untuk Aritmatika B Floating-Point iner. Ini memiliki presisi sekitar 15 digit, dan kisaran sekitar 1 E-307 hingga 1E+308. Anda juga dapat menentukan tipe data ini sebagai FLOAT atau FLOAT8.

Selain nilai numerik biasa, tipe floating-point memiliki beberapa nilai khusus. Gunakan tanda kutip tunggal di sekitar nilai-nilai ini saat menggunakannya di SQL:

- `NaN`— bukan angka
- `Infinity`— tak terbatas
- `-Infinity`— tak terbatas negatif

Misalnya, untuk menyisipkan not-a-number di kolom `day_charge` tabel `customer_activity` jalankan SQL berikut:

```
insert into customer_activity(day_charge) values('NaN');
```

Perhitungan dengan nilai numerik

Dalam konteks ini, komputasi mengacu pada operasi matematika biner: penambahan, pengurangan, perkalian, dan pembagian. Bagian ini menjelaskan jenis pengembalian yang diharapkan untuk operasi ini, serta rumus spesifik yang diterapkan untuk menentukan presisi dan skala ketika tipe data DECIMAL terlibat.

Ketika nilai numerik dihitung selama pemrosesan kueri, Anda mungkin menemukan kasus di mana perhitungan tidak mungkin dan kueri mengembalikan kesalahan luapan numerik. Anda mungkin juga mengalami kasus di mana skala nilai yang dihitung bervariasi atau tidak terduga. Untuk beberapa operasi, Anda dapat menggunakan parameter konfigurasi casting eksplisit (promosi jenis) atau Amazon Redshift untuk mengatasi masalah ini.

Untuk informasi tentang hasil perhitungan serupa dengan fungsi SQL, lihatFungsi agregat.

Jenis pengembalian untuk perhitungan

Mengingat kumpulan tipe data numerik yang didukung di Amazon Redshift, tabel berikut menunjukkan tipe pengembalian yang diharapkan untuk operasi penambahan, pengurangan, perkalian, dan pembagian. Kolom pertama di sisi kiri tabel mewakili operan pertama dalam perhitungan, dan baris atas mewakili operan kedua.

Operan 1	Operan 2	Jenis pengembalian
INT2	INT2	INT2
INT2	INT4	INT4

Operan 1	Operan 2	Jenis pengembalian
INT2	INT8	INT8
INT2	DECIMAL	DECIMAL
INT2	FLOAT4	FLOAT8
INT2	FLOAT8	FLOAT8
INT4	INT4	INT4
INT4	INT8	INT8
INT4	DECIMAL	DECIMAL
INT4	FLOAT4	FLOAT8
INT4	FLOAT8	FLOAT8
INT8	INT8	INT8
INT8	DECIMAL	DECIMAL
INT8	FLOAT4	FLOAT8
INT8	FLOAT8	FLOAT8
DECIMAL	DECIMAL	DECIMAL
DECIMAL	FLOAT4	FLOAT8
DECIMAL	FLOAT8	FLOAT8
FLOAT4	FLOAT8	FLOAT8
FLOAT8	FLOAT8	FLOAT8

Presisi dan skala hasil DESIMAL yang dihitung

Tabel berikut merangkum aturan untuk menghitung presisi dan skala yang dihasilkan ketika operasi matematika mengembalikan hasil DESIMAL. Dalam tabel ini, `p1` dan `s1` mewakili presisi dan skala operan pertama dalam perhitungan dan `p2` dan `s2` mewakili presisi dan skala operan kedua. (Terlepas dari perhitungan ini, presisi hasil maksimum adalah 38, dan skala hasil maksimum adalah 38.)

Operasi	Presisi dan skala hasil
+ atau -	Skala = `max(s1,s2)` presisi = `max(p1-s1,p2-s2)+1+scale`
*	Skala = `s1+s2` presisi = `p1+p2+1`
/	Skala = `max(4,s1+p2-s2+1)` presisi = `p1-s1+ s2+scale`

Misalnya, kolom PRICEPAID dan COMMISSION dalam tabel PENJUALAN keduanya kolom DESIMAL (8,2). Jika Anda membagi PRICEPAID dengan KOMISI (atau sebaliknya), rumus diterapkan sebagai berikut:

```
Precision = 8-2 + 2 + max(4,2+8-2+1)
= 6 + 2 + 9 = 17

Scale = max(4,2+8-2+1) = 9

Result = DECIMAL(17,9)
```

Perhitungan berikut adalah aturan umum untuk menghitung presisi dan skala yang dihasilkan untuk operasi yang dilakukan pada nilai DECIMAL dengan operator set seperti UNION, INTERSECT, dan EXCEPT atau fungsi seperti COALESCE dan DECODE:

```
Scale = max(s1,s2)
Precision = min(max(p1-s1,p2-s2)+scale,19)
```

Misalnya, tabel DEC1 dengan satu kolom DECIMAL (7,2) digabungkan dengan tabel DEC2 dengan satu kolom DECIMAL (15,3) untuk membuat tabel DEC3. Skema DEC3 menunjukkan bahwa kolom menjadi kolom NUMERIC (15,3).

```
create table dec3 as select * from dec1 union select * from dec2;
```

Hasil

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'dec3';

column |     type      | encoding | distkey | sortkey
-------+---------------+----------+---------+---------
c1     | numeric(15,3) | none     | f       | 0
```

Dalam contoh di atas, rumus diterapkan sebagai berikut:

```
Precision = min(max(7-2,15-3) + max(2,3), 19)
= 12 + 3 = 15

Scale = max(2,3) = 3

Result = DECIMAL(15,3)
```

Catatan tentang operasi divisi

Untuk operasi pembagian, kondisi dibagi dengan nol mengembalikan kesalahan.

Batas skala 100 diterapkan setelah presisi dan skala dihitung. Jika skala hasil yang dihitung lebih besar dari 100, hasil pembagian diskalakan sebagai berikut:

- presisi = `precision - (scale - max_scale)`
- Skala = `max_scale`

Jika presisi yang dihitung lebih besar dari presisi maksimum (38), presisi dikurangi menjadi 38, dan skala menjadi hasil dari: `max((38 + scale - precision), min(4, 100))`

Kondisi luapan

Overflow diperiksa untuk semua perhitungan numerik. Data DESIMAL dengan presisi 19 atau kurang disimpan sebagai bilangan bulat 64-bit. Data DESIMAL dengan presisi yang lebih besar dari 19 disimpan sebagai bilangan bulat 128-bit. Presisi maksimum untuk semua nilai DECIMAL adalah 38, dan skala maksimum adalah 37. Kesalahan luapan terjadi ketika nilai melebihi batas ini, yang berlaku untuk kumpulan hasil menengah dan akhir:

- Pengecoran eksplisit menghasilkan kesalahan overflow runtime ketika nilai data tertentu tidak sesuai dengan presisi atau skala yang diminta yang ditentukan oleh fungsi cast. Misalnya, Anda tidak dapat menampilkan semua nilai dari kolom PRICEPAID di tabel SALES (kolom DECIMAL (8,2)) dan mengembalikan hasil DECIMAL (7,3):

  ```
  select pricepaid::decimal(7,3) from sales;
  ERROR:  Numeric data overflow (result precision)
  ```

  Kesalahan ini terjadi karena beberapa nilai yang lebih besar di kolom PRICEPAID tidak dapat diberikan.

- Operasi perkalian menghasilkan hasil di mana skala hasil adalah jumlah skala masing-masing operan. Jika kedua operan memiliki skala 4, misalnya, skala hasilnya adalah 8, hanya menyisakan 10 digit untuk sisi kiri titik desimal. Oleh karena itu, relatif mudah untuk mengalami kondisi luapan ketika mengalikan dua bilangan besar yang keduanya memiliki skala signifikan.

  Contoh berikut menghasilkan kesalahan overflow.

  ```
  SELECT CAST(1 AS DECIMAL(38, 20)) * CAST(10 AS DECIMAL(38, 20));
  ERROR: 128 bit numeric data overflow (multiplication)
  ```

  Anda dapat mengatasi kesalahan luapan dengan menggunakan pembagian alih-alih perkalian. Gunakan contoh berikut untuk membagi dengan 1 dibagi dengan pembagi asli.

  ```
  SELECT CAST(1 AS DECIMAL(38, 20)) / (1 / CAST(10 AS DECIMAL(38, 20)));
  +----------+
  | ?column? |
  +----------+
  | 10       |
  +----------+
  ```

Perhitungan numerik dengan tipe INTEGER dan DECIMAL

Ketika salah satu operan dalam perhitungan memiliki tipe data INTEGER dan operan lainnya adalah DECIMAL, operan INTEGER secara implisit diberikan sebagai DECIMAL:

- INT2 (SMALLINT) diberikan sebagai DECIMAL (5,0)
- INT4 (INTEGER) diberikan sebagai DECIMAL (10,0)
- INT8 (BIGINT) diberikan sebagai DECIMAL (19,0)

Misalnya, jika Anda mengalikan SALES.COMMISSION, kolom DECIMAL (8,2) SALES.QTYSOLD, dan kolom SMALINT, perhitungan ini diberikan sebagai:

```
DECIMAL(8,2) * DECIMAL(5,0)
```

Literal integer dan floating-point

Literal atau konstanta yang mewakili angka dapat berupa integer atau floating-point.

Literal bilangan bulat

Konstanta integer adalah urutan digit 0-9, dengan tanda opsional positif (+) atau negatif (-) sebelum digit.

Sintaksis

```
[ + | - ] digit ...
```

Contoh

Bilangan bulat yang valid meliputi:

```
23
-555
+17
```

Floating-point literal

Floating-point literal (juga disebut sebagai literal desimal, numerik, atau pecahan) adalah urutan digit yang dapat mencakup titik desimal, dan opsional penanda eksponen (e).

Sintaksis

```
[ + | - ] digit ... [ . ] [ digit ...]
[ e | E [ + | - ] digit ... ]
```

Argumen

e | E

e atau E menunjukkan bahwa angka tersebut ditentukan dalam notasi ilmiah.

Contoh

Literal floating-point yang valid meliputi yang berikut:

```
3.14159
-37.
2.0e19
-2E-19
```

Contoh dengan tipe numerik

PERNYATAAN CREATE TABLE

Pernyataan CREATE TABLE berikut menunjukkan deklarasi tipe data numerik yang berbeda:

```
create table film (
film_id integer,
language_id smallint,
original_language_id smallint,
rental_duration smallint default 3,
rental_rate numeric(4,2) default 4.99,
length smallint,
replacement_cost real default 25.00);
```

Cobalah untuk menyisipkan bilangan bulat yang berada di luar jangkauan

Contoh berikut mencoba untuk memasukkan nilai 33000 ke dalam kolom SMALLINT.

```
insert into film(language_id) values(33000);
```

Rentang untuk SMALLINT adalah -32768 hingga +32767, jadi Amazon Redshift mengembalikan kesalahan.

```
An error occurred when executing the SQL command:
insert into film(language_id) values(33000)

ERROR: smallint out of range [SQL State=22003]
```

Masukkan nilai desimal ke dalam kolom integer

Contoh berikut menyisipkan nilai desimal ke kolom INT.

```
insert into film(language_id) values(1.5);
```

Nilai ini dimasukkan tetapi dibulatkan ke nilai integer 2.

Masukkan desimal yang berhasil karena skalanya dibulatkan

Contoh berikut menyisipkan nilai desimal yang memiliki presisi lebih tinggi dari kolom.

```
insert into film(rental_rate) values(35.512);
```

Dalam hal ini, nil `35.51` ainya dimasukkan ke dalam kolom.

Cobalah untuk memasukkan nilai desimal yang berada di luar jangkauan

Dalam hal ini, nilainya `350.10` di luar jangkauan. Jumlah digit untuk nilai dalam kolom DECIMAL sama dengan presisi kolom dikurangi skalanya (4 dikurangi 2 untuk kolom RENTAL_RATE). Dengan kata lain, rentang yang diizinkan untuk `DECIMAL(4,2)` kolom sudah hab `-99.99` is`99.99`.

```
insert into film(rental_rate) values (350.10);
ERROR:  numeric field overflow
DETAIL:  The absolute value is greater than or equal to 10^2 for field with precision
 4, scale 2.
```

Masukkan nilai presisi variabel ke kolom REAL

Contoh berikut menyisipkan nilai presisi variabel ke kolom REAL.

```
insert into film(replacement_cost) values(1999999.99);
```

```
insert into film(replacement_cost) values(1999.99);

select replacement_cost from film;

+------------------+
| replacement_cost |
+------------------+
| 2000000          |
| 1999.99          |
+------------------+
```

Nilai `1999999.99` dikonversi `2000000` untuk memenuhi persyaratan presisi untuk `REAL` kolom. Nilai `1999.99` dimuat apa adanya.

Jenis karakter

Topik

- Penyimpanan dan rentang
- CHAR atau KARAKTER
- VARCHAR atau KARAKTER VARIASI
- Jenis NCHAR dan NVARCHAR
- Jenis TEKS dan BPCHAR
- Signifikansi blanko yang tertinggal
- Contoh dengan tipe karakter

Tipe data karakter termasuk CHAR (karakter) dan VARCHAR (karakter bervariasi).

Penyimpanan dan rentang

Tipe data CHAR dan VARCHAR didefinisikan dalam istilah byte, bukan karakter. Kolom CHAR hanya dapat berisi karakter byte tunggal, sehingga kolom CHAR (10) dapat berisi string dengan panjang maksimum 10 byte. Sebuah VARCHAR dapat berisi karakter multibyte, hingga maksimal empat byte per karakter. Misalnya, kolom VARCHAR (12) dapat berisi 12 karakter byte tunggal, 6 karakter dua byte, 4 karakter tiga byte, atau 3 karakter empat byte.

Nama	Penyimpanan	Rentang (lebar kolom)
CHAR, KARAKTER atau NCHAR	Panjang string, termasuk sisa kosong (jika ada)	4096 bita
VARCHAR, VARIASI KARAKTER, atau NVARCHAR	4 byte + total byte untuk karakter, di mana setiap karakter bisa 1 hingga 4 byte.	65535 byte (64K -1)
BPCHAR	Dikonversi ke CHAR dengan panjang tetap (256).	256 byte
TEXT	Dikonversi ke VARCHAR (256).	260 byte

Note

Sintaks CREATE TABLE mendukung kata kunci MAX untuk tipe data karakter. Contoh:

```
create table test(col1 varchar(max));
```

Untuk CHAR, MAX mendefinisikan lebar kolom sebagai 4096 byte.
Untuk VARCHAR, MAX mendefinisikan lebar kolom sebagai 65.535 byte dalam pernyataan CREATE TABLE. Untuk operasi dalam memori, VARCHAR (MAX) mendukung hingga 16.000.000 byte.

CHAR atau KARAKTER

Gunakan kolom CHAR atau CHARACTER untuk menyimpan string dengan panjang tetap. String ini dilapisi dengan blanko, sehingga kolom CHAR (10) selalu menempati 10 byte penyimpanan.

```
char(10)
```

Kolom CHAR tanpa spesifikasi panjang menghasilkan kolom CHAR (1).

VARCHAR atau KARAKTER VARIASI

Gunakan kolom VARCHAR atau CHARACTER VARIING untuk menyimpan string panjang variabel dengan batas tetap. String ini tidak dilapisi dengan blanko, jadi kolom VARCHAR (120) terdiri dari maksimum 120 karakter byte tunggal, 60 karakter dua byte, 40 karakter tiga byte, atau 30 karakter empat byte.

```
varchar(120)
```

Jika Anda menggunakan tipe data VARCHAR tanpa penentu panjang dalam pernyataan CREATE TABLE, panjang defaultnya adalah 256.

Fungsi stringsekarang mendukung hingga 16.000.000 byte. Misalnya, output fungsi CONCAT sebelumnya terbatas pada 65535 byte, tetapi sekarang mendukung hingga 16.000.000 byte.

```
SELECT LEN(CONCAT(REPEAT('A', 5000000), REPEAT('B', 5000000))) AS total_length;

 total_length
--------------
     10000000
```

Jenis NCHAR dan NVARCHAR

Anda dapat membuat kolom dengan tipe NCHAR dan NVARCHAR (juga dikenal sebagai tipe NATIONAL CHARACTER dan NATIONAL CHARACTER VARIING). Tipe-tipe ini dikonversi ke tipe CHAR dan VARCHAR, masing-masing, dan disimpan dalam jumlah byte yang ditentukan.

Kolom NCHAR tanpa spesifikasi panjang diubah menjadi kolom CHAR (1).

Kolom NVARCHAR tanpa spesifikasi panjang diubah menjadi kolom VARCHAR (256).

Jenis TEKS dan BPCHAR

Anda dapat membuat tabel Amazon Redshift dengan kolom TEXT, tetapi dikonversi ke kolom VARCHAR (256) yang menerima nilai panjang variabel dengan maksimum 256 karakter.

Anda dapat membuat kolom Amazon Redshift dengan tipe BPCHAR (karakter kosong), yang dikonversi Amazon Redshift menjadi kolom CHAR (256) dengan panjang tetap.

Signifikansi blanko yang tertinggal

Tipe data CHAR dan VARCHAR menyimpan string hingga panjang n byte. Upaya untuk menyimpan string yang lebih panjang ke dalam kolom jenis ini menghasilkan kesalahan, kecuali karakter tambahan semuanya spasi (kosong), dalam hal ini string dipotong ke panjang maksimum. Jika string lebih pendek dari panjang maksimum, nilai CHAR dilapisi dengan blanko, tetapi nilai VARCHAR menyimpan string tanpa blanko.

Terakhir kosong dalam nilai CHAR selalu tidak signifikan secara semantik. Mereka diabaikan ketika Anda membandingkan dua nilai CHAR, tidak termasuk dalam perhitungan LENGTH, dan dihapus ketika Anda mengonversi nilai CHAR ke jenis string lain.

Spasi akhir dalam nilai VARCHAR dan CHAR diperlakukan sebagai tidak signifikan secara semantik ketika nilai dibandingkan.

Perhitungan panjang mengembalikan panjang string karakter VARCHAR dengan spasi akhir termasuk dalam panjang. Bagian akhir kosong tidak dihitung dalam panjang untuk string karakter dengan panjang tetap.

Contoh dengan tipe karakter

PERNYATAAN CREATE TABLE

Pernyataan CREATE TABLE berikut menunjukkan penggunaan tipe data VARCHAR dan CHAR:

```
create table address(
address_id integer,
address1 varchar(100),
address2 varchar(50),
district varchar(20),
city_name char(20),
state char(2),
postal_code char(5)
```

```
);
```

Contoh berikut menggunakan tabel ini.

Mengakhiri bagian kosong dalam string karakter dengan panjang variabel

Karena ADDRESS1 adalah kolom VARCHAR, bagian akhir yang kosong di alamat yang dimasukkan kedua secara semantik tidak signifikan. Dengan kata lain, dua alamat yang dimasukkan ini cocok.

```
insert into address(address1) values('9516 Magnolia Boulevard');

insert into address(address1) values('9516 Magnolia Boulevard  ');
```

```
select count(*) from address
where address1='9516 Magnolia Boulevard';

count
-------
2
(1 row)
```

Jika kolom ADDRESS1 adalah kolom CHAR dan nilai yang sama dimasukkan, kueri COUNT (*) akan mengenali string karakter sebagai sama dan kembali2.

Hasil dari fungsi LENGTH

Fungsi LENGTH mengenali bagian kosong di kolom VARCHAR:

```
select length(address1) from address;

length
--------
23
25
(2 rows)
```

Nilai `Augusta` di kolom CITY_NAME, yang merupakan kolom CHAR, akan selalu mengembalikan panjang 7 karakter, terlepas dari kekosongan akhir dalam string input.

Nilai yang melebihi panjang kolom

String karakter tidak dipotong agar sesuai dengan lebar kolom yang dinyatakan:

```
insert into address(city_name) values('City of South San Francisco');
ERROR: value too long for type character(20)
```

Solusi untuk masalah ini adalah dengan memberikan nilai ke ukuran kolom:

```
insert into address(city_name)
values('City of South San Francisco'::char(20));
```

Dalam hal ini, 20 karakter pertama dari string (`City of South San Fr`) akan dimuat ke dalam kolom.

Jenis tanggal-waktu

Topik

- Penyimpanan dan rentang
- DATE
- TIME
- TIMETZ
- TIMESTAMP
- TIMESTAMPTZ
- Contoh dengan tipe datetime
- Literal tanggal, waktu, dan stempel waktu
- Jenis data interval dan literal

Jenis data datetime termasuk DATE, TIME, TIMETZ, TIMESTAMP, dan TIMESTAMPTZ.

Penyimpanan dan rentang

Nama	Penyimpanan	Kisaran	Resolusi
DATE	4 byte	4713 SM hingga 294276 M	1 hari
TIME	8 byte	00:00:00 hingga 24:00:00	1 mikrodetik

Nama	Penyimpanan	Kisaran	Resolusi
TIMETZ	8 byte	00:00:00 +1459 hingga 00:00:00 +1459	1 mikrodetik
TIMESTAMP	8 byte	4713 SM hingga 294276 M	1 mikrodetik
TIMESTAMPTZ	8 byte	4713 SM hingga 294276 M	1 mikrodetik

DATE

Gunakan tipe data DATE untuk menyimpan tanggal kalender sederhana tanpa stempel waktu.

TIME

Waktu adalah alias dari TIME WITHOUT TIME ZONE.

Gunakan tipe data TIME untuk menyimpan waktu dalam sehari.

Kolom TIME menyimpan nilai dengan presisi maksimal enam digit untuk detik pecahan.

Secara default, nilai TIME adalah Waktu Universal Terkoordinasi (UTC) di tabel pengguna dan tabel sistem Amazon Redshift.

TIMETZ

TIMETZ adalah alias dari TIME WITH TIME ZONE.

Gunakan tipe data TIMETZ untuk menyimpan waktu hari dengan zona waktu.

Kolom TIMETZ menyimpan nilai dengan presisi maksimal enam digit untuk detik pecahan.

Secara default, nilai TIMETZ adalah UTC di tabel pengguna dan tabel sistem Amazon Redshift.

TIMESTAMP

TIMESTAMP adalah alias dari TIMESTAMP TANPA ZONA WAKTU.

Gunakan tipe data TIMESTAMP untuk menyimpan nilai stempel waktu lengkap yang mencakup tanggal dan waktu hari.

Kolom TIMESTAMP menyimpan nilai dengan presisi maksimal enam digit untuk detik pecahan.

Jika Anda memasukkan tanggal ke kolom TIMESTAMP, atau tanggal dengan nilai stempel waktu sebagian, nilai tersebut secara implisit diubah menjadi nilai stempel waktu penuh. Nilai stempel waktu penuh ini memiliki nilai default (00) untuk jam, menit, dan detik yang hilang. Nilai zona waktu dalam string input diabaikan.

Secara default, nilai TIMESTAMP adalah UTC di tabel pengguna dan tabel sistem Amazon Redshift.

TIMESTAMPTZ

TIMESTAMPTZ adalah alias dari TIMESTAMP WITH TIME ZONE.

Gunakan tipe data TIMESTAMPTZ untuk memasukkan nilai stempel waktu lengkap yang mencakup tanggal, waktu hari, dan zona waktu. Ketika nilai input menyertakan zona waktu, Amazon Redshift menggunakan zona waktu untuk mengonversi nilai ke UTC dan menyimpan nilai UTC.

Untuk melihat daftar nama zona waktu yang didukung, jalankan perintah berikut.

```
select pg_timezone_names();
```

Untuk melihat daftar singkatan zona waktu yang didukung, jalankan perintah berikut.

```
select pg_timezone_abbrevs();
```

Anda juga dapat menemukan informasi terkini tentang zona waktu di IANA Time Zone Database.

Tabel berikut memiliki contoh format zona waktu.

format	Contoh
dd mon hh:mi:ss yyyy tz	17 Des 07:37:16 1997 PST
mm/dd/yyyy hh:mi:ss.ss tz	12/17/1997 07:37:16.00 PST
mm/dd/yyyy hh:mi:ss.ss tz	12/17/1997 07:37:16.00 US/Pacific
yyyy-mm-dd hh: mi: ss +/- tz	1997-12-17 07:37:16-08
dd.mm.yyyy hh:mi:ss tz	17.12.1997 07:37:16.00 PST

Kolom TIMESTAMPTZ menyimpan nilai dengan presisi maksimal enam digit untuk detik pecahan.

Jika Anda memasukkan tanggal ke kolom TIMESTAMPTZ, atau tanggal dengan stempel waktu sebagian, nilainya secara implisit diubah menjadi nilai stempel waktu penuh. Nilai stempel waktu penuh ini memiliki nilai default (00) untuk jam, menit, dan detik yang hilang.

Nilai TIMESTAMPTZ adalah UTC dalam tabel pengguna.

Contoh dengan tipe datetime

Berikut ini, Anda dapat menemukan contoh untuk bekerja dengan tipe datetime yang didukung oleh Amazon Redshift.

Contoh tanggal

Contoh berikut menyisipkan tanggal yang memiliki format berbeda dan menampilkan output.

```
create table datetable (start_date date, end_date date);
```

```
insert into datetable values ('2008-06-01','2008-12-31');

insert into datetable values ('Jun 1,2008','20081231');
```

```
select * from datetable order by 1;

start_date |  end_date
-----------------------
2008-06-01 | 2008-12-31
2008-06-01 | 2008-12-31
```

Jika Anda memasukkan nilai stempel waktu ke kolom DATE, bagian waktu diabaikan dan hanya tanggal yang dimuat.

Contoh waktu

Contoh berikut menyisipkan nilai TIME dan TIMETZ yang memiliki format berbeda dan menampilkan output.

```
create table timetable (start_time time, end_time timetz);
```

```
insert into timetable values ('19:11:19','20:41:19 UTC');
insert into timetable values ('191119', '204119 UTC');
```

```
select * from timetable order by 1;
start_time |  end_time
-----------------------
 19:11:19  | 20:41:19+00
 19:11:19  | 20:41:19+00
```

Contoh cap waktu

Jika Anda memasukkan tanggal ke kolom TIMESTAMP atau TIMESTAMPTZ, waktu default adalah tengah malam. Misalnya, jika Anda memasukkan literal`20081231`, nilai yang disimpan adalah`2008-12-31 00:00:00`.

Untuk mengubah zona waktu untuk sesi saat ini, gunakan SET perintah untuk mengatur parameter timezone konfigurasi.

Contoh berikut menyisipkan stempel waktu yang memiliki format berbeda dan menampilkan tabel yang dihasilkan.

```
create table tstamp(timeofday timestamp, timeofdaytz timestamptz);

insert into tstamp values('Jun 1,2008  09:59:59', 'Jun 1,2008 09:59:59 EST' );
insert into tstamp values('Dec 31,2008 18:20','Dec 31,2008 18:20');
insert into tstamp values('Jun 1,2008  09:59:59 EST', 'Jun 1,2008 09:59:59');

SELECT * FROM tstamp;

+---------------------+------------------------+
|      timeofday      |      timeofdaytz       |
+---------------------+------------------------+
| 2008-06-01 09:59:59 | 2008-06-01 14:59:59+00 |
| 2008-12-31 18:20:00 | 2008-12-31 18:20:00+00 |
| 2008-06-01 09:59:59 | 2008-06-01 09:59:59+00 |
+---------------------+------------------------+
```

Literal tanggal, waktu, dan stempel waktu

Berikut ini adalah aturan untuk bekerja dengan literal tanggal, waktu, dan stempel waktu yang didukung oleh Amazon Redshift.

Tanggal

Tanggal input berikut adalah semua contoh valid dari nilai tanggal literal untuk tipe data DATE yang dapat Anda muat ke tabel Amazon Redshift. `MDY DateStyle`Mode default diasumsikan berlaku. Mode ini berarti bahwa nilai bulan mendahului nilai hari dalam string seperti `1999-01-08` dan`01/02/00`.

> **Note**
>
> Literal tanggal atau stempel waktu harus dilampirkan dalam tanda kutip saat Anda memuatnya ke dalam tabel.

Tanggal masukan	Tanggal penuh
8 Januari 1999	8 Januari 1999
1999-01-08	8 Januari 1999
1/8/1999	8 Januari 1999
01/02/00	2 Januari 2000
2000- Jan-31	Januari 31, 2000
Jan-31-2000	Januari 31, 2000
31- Jan-2000	Januari 31, 2000
20080215	Februari 15, 2008
080215	Februari 15, 2008
2008.366	31 Desember 2008 (bagian tiga digit tanggal harus antara 001 dan 366)

Waktu

Waktu input berikut adalah semua contoh valid dari nilai waktu literal untuk tipe data TIME dan TIMETZ yang dapat Anda muat ke tabel Amazon Redshift.

Waktu masukan	Deskripsi (bagian waktu)
04:05:06.789	4:05 pagi dan 6.789 detik
04:05:06	4:05 AM dan 6 detik
04:05	4:05 pagi tepatnya
4040506	4:05 AM dan 6 detik
04:05 PAGI	4:05 pagi tepatnya; AM adalah opsional
04:05 SORE	4:05 PM tepatnya; nilai jam harus kurang dari 12
16:05	4:05 WIB tepatnya

Stempel waktu

Stempel waktu input berikut adalah contoh valid dari nilai waktu literal untuk tipe data TIMESTAMP dan TIMESTAMPTZ yang dapat Anda muat ke tabel Amazon Redshift. Semua literal tanggal yang valid dapat digabungkan dengan literal waktu berikut.

Masukkan stempel waktu (tanggal dan waktu yang digabungkan)	Deskripsi (bagian waktu)
20080215 04:05:06.789	4:05 pagi dan 6.789 detik
20080215 04:05:06	4:05 AM dan 6 detik
20080215 0405	4:05 pagi tepatnya
20080215 4040506	4:05 AM dan 6 detik
20080215 04:05 PAGI	4:05 pagi tepatnya; AM adalah opsional
20080215 04:05 SORE	4:05 PM tepatnya; nilai jam harus kurang dari 12

Masukkan stempel waktu (tanggal dan waktu yang digabungkan)	Deskripsi (bagian waktu)
20080215 1605	4:05 WIB tepatnya
20080215	Midnight (secara default)

Nilai datetime khusus

Nilai-nilai khusus berikut dapat digunakan sebagai literal datetime dan sebagai argumen untuk fungsi tanggal. Mereka memerlukan tanda kutip tunggal dan dikonversi ke nilai stempel waktu reguler selama pemrosesan kueri.

Nilai khusus	Deskripsi
now	Mengevaluasi waktu mulai transaksi saat ini dan mengembalikan stempel waktu dengan presisi mikrodetik.
today	Mengevaluasi ke tanggal yang sesuai dan mengembalikan stempel waktu dengan nol untuk bagian waktu.
tomorrow	Mengevaluasi ke tanggal yang sesuai dan mengembalikan stempel waktu dengan nol untuk bagian waktu.
yesterday	Mengevaluasi ke tanggal yang sesuai dan mengembalikan stempel waktu dengan nol untuk bagian waktu.

Contoh berikut menunjukkan bagaimana now dan today bekerja dengan fungsi DATEADD.

```
select dateadd(day,1,'today');

date_add
---------------------
2009-11-17 00:00:00
```

```
(1 row)

select dateadd(day,1,'now');

date_add
----------------------------
2009-11-17 10:45:32.021394
(1 row)
```

Jenis data interval dan literal

Anda dapat menggunakan tipe data interval untuk menyimpan durasi waktu dalam satuan seperti,,`seconds`,`minutes`,`hours`, `daysmonths`, dan`years`. Jenis data interval dan literal dapat digunakan dalam perhitungan datetime, seperti, menambahkan interval ke tanggal dan stempel waktu, menjumlahkan interval, dan mengurangi interval dari tanggal atau stempel waktu. Literal interval dapat digunakan sebagai nilai input ke kolom tipe data interval dalam tabel.

Sintaks tipe data interval

Untuk menentukan tipe data interval untuk menyimpan durasi waktu dalam tahun dan bulan:

```
INTERVAL year_to_month_qualifier
```

Untuk menentukan tipe data interval untuk menyimpan durasi dalam hari, jam, menit, dan detik:

```
INTERVAL day_to_second_qualifier [ (fractional_precision) ]
```

Sintaks interval literal

Untuk menentukan interval literal untuk menentukan durasi waktu dalam tahun dan bulan:

```
INTERVAL quoted-string year_to_month_qualifier
```

Untuk menentukan interval literal untuk menentukan durasi dalam hari, jam, menit, dan detik:

```
INTERVAL quoted-string day_to_second_qualifier [ (fractional_precision) ]
```

Argumen

string kutipan

Menentukan nilai numerik positif atau negatif yang menentukan kuantitas dan unit datetime sebagai string input. Jika string yang dikutip hanya berisi angka, maka Amazon Redshift menentukan unit dari year _to_month_qualifier atau day_to_second_qualifier. Misalnya, `'23' MONTH` mewakili`1 year 11 months`, `'-2' DAY` `'1-2' MONTH` mewakili `-2 days 0 hours 0 minutes 0.0 seconds`1 `year 2 months`, dan `'13 day 1 hour 1 minute 1.123 seconds' SECOND` mewakili`13 days 1 hour 1 minute 1.123 seconds`. Untuk informasi selengkapnya tentang format keluaran interval, lihatGaya interval.

kualifikasi tahun_to_bulan_

Menentukan rentang interval. Jika Anda menggunakan kualifikasi dan membuat interval dengan satuan waktu yang lebih kecil dari kualifikasi, Amazon Redshift memotong dan membuang bagian interval yang lebih kecil. Nilai yang valid untuk year_to_month_qualifier adalah:

- `YEAR`
- `MONTH`
- `YEAR TO MONTH`

kualifikasi day_to_second

Menentukan rentang interval. Jika Anda menggunakan kualifikasi dan membuat interval dengan satuan waktu yang lebih kecil dari kualifikasi, Amazon Redshift memotong dan membuang bagian interval yang lebih kecil. Nilai yang valid untuk day_to_second_qualifier adalah:

- `DAY`
- `HOUR`
- `MINUTE`
- `SECOND`
- `DAY TO HOUR`
- `DAY TO MINUTE`
- `DAY TO SECOND`
- `HOUR TO MINUTE`
- `HOUR TO SECOND`
- `MINUTE TO SECOND`

Output dari literal INTERVAL dipotong ke komponen INTERVAL terkecil yang ditentukan. Misalnya, saat menggunakan kualifikasi MINUTE, Amazon Redshift membuang satuan waktu yang lebih kecil dari MINUTE.

```
select INTERVAL '1 day 1 hour 1 minute 1.123 seconds' MINUTE
```

Nilai yang dihasilkan dipotong menjadi. `'1 day 01:01:00'`

presisi pecahan

Parameter opsional yang menentukan jumlah digit pecahan yang diizinkan dalam interval. Argumen fractional_precision hanya boleh ditentukan jika interval Anda berisi SECOND. Misalnya, membuat `SECOND(3)` interval yang memungkinkan hanya tiga digit pecahan, seperti 1,234 detik. Jumlah maksimum digit pecahan adalah enam.

Konfigurasi sesi `interval_forbid_composite_literals` menentukan apakah kesalahan dikembalikan ketika interval ditentukan dengan bagian YEAR TO MONTH dan DAY TO SECOND. Untuk informasi selengkapnya, lihat interval_terban_composite_literal.

Interval aritmatika

Anda dapat menggunakan nilai interval dengan nilai datetime lainnya untuk melakukan operasi aritmatika. Tabel berikut menjelaskan operasi yang tersedia dan jenis data apa yang dihasilkan dari setiap operasi.

> **Note**
>
> Operasi yang dapat menghasilkan keduanya `date` dan `timestamp` hasilnya melakukannya berdasarkan satuan waktu terkecil yang terlibat dalam persamaan. Misalnya, ketika Anda menambahkan a `interval` ke a `date` hasilnya adalah `date` jika itu adalah interval TAHUN KE BULAN, dan stempel waktu jika itu adalah interval HARI KE KEDUA.

Operasi di mana operan pertama adalah `interval` menghasilkan hasil berikut untuk operan kedua yang diberikan:

Operator	Date	Stempel waktu	Interval	Numerik
-	N/A	N/A	Interval	N/A

Operator	Date	Stempel waktu	Interval	Numerik
+	Date	Date/Timestamp	Interval	N/A
*	N/A	N/A	N/A	Interval
/	N/A	N/A	N/A	Interval

Operasi di mana operan pertama adalah a `date` menghasilkan hasil berikut untuk operan kedua yang diberikan:

Operator	Date	Stempel waktu	Interval	Numerik
-	Numerik	Interval	Date/Timestamp	Date
+	N/A	N/A	N/A	N/A

Operasi di mana operan pertama adalah a `timestamp` menghasilkan hasil berikut untuk operan kedua yang diberikan:

Operator	Date	Stempel waktu	Interval	Numerik
-	Numerik	Interval	Stempel waktu	Stempel waktu
+	N/A	N/A	N/A	N/A

Gaya interval

Anda dapat menggunakan the section called “SET” perintah SQL untuk mengubah format tampilan output dari nilai interval Anda. Saat Anda menggunakan tipe data interval di SQL, berikan ke teks untuk melihat gaya interval yang diharapkan, misalnya,`YEAR TO MONTH::text`. Nilai yang tersedia untuk SET nil `IntervalStyle` ainya adalah:

- `postgres`— mengikuti gaya PostgreSQL. Ini adalah opsi default.
- `postgres_verbose`— mengikuti gaya verbose PostgreSQL.
- `sql_standard`— mengikuti gaya literal interval standar SQL.

Perintah berikut mengatur gaya interval kesql_standard.

```
SET IntervalStyle to 'sql_standard';
```

format output postgres

Berikut ini adalah format output untuk gaya postgres interval. Setiap nilai numerik bisa negatif.

```
'<numeric> <unit> [, <numeric> <unit> ...]'
```

```
select INTERVAL '1-2' YEAR TO MONTH::text

varchar
---------------
1 year 2 mons
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text

varchar
------------------
1 day 02:03:04.5678
```

format keluaran postgres_verbose

sintaks postgres_verbose mirip dengan postgres, tetapi output postgres_verbose juga berisi satuan waktu.

```
'[@] <numeric> <unit> [, <numeric> <unit> ...] [direction]'
```

```
select INTERVAL '1-2' YEAR TO MONTH::text

varchar
-----------------
@ 1 year 2 mons
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text

varchar
---------------------------
```

```
@ 1 day 2 hours 3 mins 4.56 secs
```

format keluaran sql_standar

Nilai interval tahun ke bulan diformat sebagai berikut. Menentukan tanda negatif sebelum interval menunjukkan interval adalah nilai negatif dan berlaku untuk seluruh interval.

```
'[-]yy-mm'
```

Nilai interval hari ke detik diformat sebagai berikut.

```
'[-]dd hh:mm:ss.ffffff'
```

```
SELECT INTERVAL '1-2' YEAR TO MONTH::text

varchar
-------
1-2
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text

varchar
---------------
1 2:03:04.5678
```

Contoh tipe data interval

Contoh berikut menunjukkan cara menggunakan tipe data INTERVAL dengan tabel.

```
create table sample_intervals (y2m interval month, h2m interval hour to minute);
insert into sample_intervals values (interval '20' month, interval '2 days
 1:1:1.123456' day to second);
select y2m::text, h2m::text from sample_intervals;


      y2m      |       h2m
---------------+-----------------
 1 year 8 mons | 2 days 01:01:00
```

```
update sample_intervals set y2m = interval '2' year where y2m = interval '1-8' year to
 month;
```

```
select * from sample_intervals;

   y2m   |       h2m
---------+-----------------
 2 years | 2 days 01:01:00
```

```
delete from sample_intervals where h2m = interval '2 1:1:0' day to second;
select * from sample_intervals;

 y2m | h2m
-----+-----
```

Contoh literal interval

Contoh berikut dijalankan dengan gaya interval disetel ke`postgres`.

Contoh berikut menunjukkan cara membuat INTERVAL literal 1 tahun.

```
select INTERVAL '1' YEAR

intervaly2m
---------------
1 years 0 mons
```

Jika Anda menentukan string kutipan yang melebihi kualifikasi, satuan waktu yang tersisa dipotong dari interval. Dalam contoh berikut, interval 13 bulan menjadi 1 tahun dan 1 bulan, tetapi 1 bulan sisanya ditinggalkan karena kualifikasi TAHUN.

```
select INTERVAL '13 months' YEAR

intervaly2m
---------------
1 years 0 mons
```

Jika Anda menggunakan kualifikasi yang lebih rendah dari string interval Anda, unit sisa disertakan.

```
select INTERVAL '13 months' MONTH

intervaly2m
---------------
```

```
1 years 1 mons
```

Menentukan presisi dalam interval Anda memotong jumlah digit pecahan ke presisi yang ditentukan.

```
select INTERVAL '1.234567' SECOND (3)

intervald2s
--------------------------------
0 days 0 hours 0 mins 1.235 secs
```

Jika Anda tidak menentukan presisi, Amazon Redshift menggunakan presisi maksimum 6.

```
select INTERVAL '1.23456789' SECOND

intervald2s
-----------------------------------
0 days 0 hours 0 mins 1.234567 secs
```

Contoh berikut menunjukkan cara membuat interval jarak.

```
select INTERVAL '2:2' MINUTE TO SECOND

intervald2s
-----------------------------
0 days 0 hours 2 mins 2.0 secs
```

Kualifikasi menentukan unit yang Anda tentukan. Misalnya, meskipun contoh berikut menggunakan string kutipan '2:2' yang sama seperti contoh sebelumnya, Amazon Redshift mengakui bahwa ia menggunakan satuan waktu yang berbeda karena kualifikasi.

```
select INTERVAL '2:2' HOUR TO MINUTE

intervald2s
-----------------------------
0 days 2 hours 2 mins 0.0 secs
```

Singkatan dan jamak setiap unit juga didukung. Misalnya,, `5s` `5 second`, dan `5 seconds` merupakan interval yang setara. Unit yang didukung adalah tahun, bulan, jam, menit, dan detik.

```
select INTERVAL '5s' SECOND
```

```
intervald2s
----------------------------
0 days 0 hours 0 mins 5.0 secs
```

```
select INTERVAL '5 HOURS' HOUR

intervald2s
----------------------------
0 days 5 hours 0 mins 0.0 secs
```

```
select INTERVAL '5 h' HOUR

intervald2s
----------------------------
0 days 5 hours 0 mins 0.0 secs
```

Contoh literal interval tanpa sintaks kualifier

> **Note**
>
> Contoh berikut menunjukkan penggunaan interval literal tanpa ku `DAY TO SECOND` alifikasi `YEAR TO MONTH` atau. Untuk informasi tentang menggunakan interval literal yang disarankan dengan kualifikasi, lihatJenis data interval dan literal.

Gunakan interval literal untuk mengidentifikasi periode waktu tertentu, seperti `12 hours` atau6 `months`. Anda dapat menggunakan literal interval ini dalam kondisi dan perhitungan yang melibatkan ekspresi datetime.

Interval literal dinyatakan sebagai kombinasi dari kata kunci INTERVAL dengan kuantitas numerik dan bagian tanggal yang didukung, misalnya `INTERVAL '7 days'` atau`INTERVAL '59 minutes'`. Anda dapat menghubungkan beberapa jumlah dan unit untuk membentuk interval yang lebih tepat, misalnya:`INTERVAL '7 days, 3 hours, 59 minutes'`. Singkatan dan jamak dari setiap unit juga didukung; misalnya:`5 s`,`5 second`, dan merupakan interval `5 seconds` yang setara.

Jika Anda tidak menentukan bagian tanggal, nilai interval mewakili detik. Anda dapat menentukan nilai kuantitas sebagai pecahan (misalnya:`0.5 days`).

Contoh berikut menunjukkan serangkaian perhitungan dengan nilai interval yang berbeda.

Berikut ini menambahkan 1 detik ke tanggal yang ditentukan.

```
select caldate + interval '1 second' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 00:00:01
(1 row)
```

Berikut ini menambahkan 1 menit ke tanggal yang ditentukan.

```
select caldate + interval '1 minute' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 00:01:00
(1 row)
```

Berikut ini menambahkan 3 jam dan 35 menit ke tanggal yang ditentukan.

```
select caldate + interval '3 hours, 35 minutes' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 03:35:00
(1 row)
```

Berikut ini menambahkan 52 minggu ke tanggal yang ditentukan.

```
select caldate + interval '52 weeks' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2009-12-30 00:00:00
(1 row)
```

Berikut ini menambahkan 1 minggu, 1 jam, 1 menit, dan 1 detik ke tanggal yang ditentukan.

```
select caldate + interval '1w, 1h, 1m, 1s' as dateplus from date
```

```
where caldate='12-31-2008';
dateplus
---------------------
2009-01-07 01:01:01
(1 row)
```

Berikut ini menambahkan 12 jam (setengah hari) ke tanggal yang ditentukan.

```
select caldate + interval '0.5 days' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 12:00:00
(1 row)
```

Berikut ini mengurangi 4 bulan dari 15 Februari 2023 dan hasilnya adalah 15 Oktober 2022.

```
select date '2023-02-15' - interval '4 months';

?column?
---------------------
2022-10-15 00:00:00
```

Berikut ini mengurangi 4 bulan dari 31 Maret 2023 dan hasilnya adalah 30 November 2022. Perhitungan mempertimbangkan jumlah hari dalam sebulan.

```
select date '2023-03-31' - interval '4 months';

?column?
---------------------
2022-11-30 00:00:00
```

Tipe Boolean

Gunakan tipe data BOOLEAN untuk menyimpan nilai true dan false dalam kolom byte tunggal. Tabel berikut menjelaskan tiga kemungkinan status untuk nilai Boolean dan nilai literal yang menghasilkan keadaan itu. Terlepas dari string input, kolom Boolean menyimpan dan mengeluarkan “t” untuk true dan “f” untuk false.

Status	Nilai literal yang valid	Penyimpanan
True	`TRUE 't' 'true' 'y' 'yes' '1'`	1 byte
False	`FALSE 'f' 'false' 'n' 'no' '0'`	1 byte
Tidak Diketahui	`NULL`	1 byte

Anda dapat menggunakan perbandingan IS untuk memeriksa nilai Boolean hanya sebagai predikat dalam klausa WHERE. Anda tidak dapat menggunakan perbandingan IS dengan nilai Boolean dalam daftar SELECT.

Contoh

Anda dapat menggunakan kolom BOOLEAN untuk menyimpan status "Active/Inactive" untuk setiap pelanggan dalam tabel CUSTOMER.

```
create table customer(
custid int,
active_flag boolean default true);
```

```
insert into customer values(100, default);
```

```
select * from customer;
custid | active_flag
-------+-------------
   100 | t
```

Jika tidak ada nilai default (`true`atau`false`) yang ditentukan dalam pernyataan CREATE TABLE, memasukkan nilai default berarti memasukkan null.

Dalam contoh ini, kueri memilih pengguna dari tabel USERS yang menyukai olahraga tetapi tidak suka teater:

```
select firstname, lastname, likesports, liketheatre
from users
where likesports is true and liketheatre is false
order by userid limit 10;

firstname |  lastname  | likesports | liketheatre
----------+------------+------------+-------------
Lars      | Ratliff    | t          | f
Mufutau   | Watkins    | t          | f
Scarlett  | Mayer      | t          | f
Shafira   | Glenn      | t          | f
Winifred  | Cherry     | t          | f
Chase     | Lamb       | t          | f
Liberty   | Ellison    | t          | f
Aladdin   | Haney      | t          | f
Tashya    | Michael    | t          | f
Lucian    | Montgomery | t          | f
(10 rows)
```

Contoh berikut memilih pengguna dari tabel USERS untuk siapa tidak diketahui apakah mereka menyukai musik rock.

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;

firstname | lastname | likerock
----------+----------+----------
Rafael    | Taylor   |
Vladimir  | Humphrey |
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett  | Mayer    |
(10 rows)
```

Contoh berikut mengembalikan kesalahan karena menggunakan perbandingan IS dalam daftar SELECT.

```
select firstname, lastname, likerock is true as "check"
from users
order by userid limit 10;

[Amazon](500310) Invalid operation: Not implemented
```

Contoh berikut berhasil karena menggunakan perbandingan yang sama (=) dalam daftar SELECT alih-alih perbandingan IS.

```
select firstname, lastname, likerock = true as "check"
from users
order by userid limit 10;

firstname | lastname  | check
----------+-----------+------
Rafael    | Taylor    |
Vladimir  | Humphrey  |
Lars      | Ratliff   | true
Barry     | Roy       |
Reagan    | Hodge     | true
Victor    | Hernandez | true
Tamekah   | Juarez    |
Colton    | Roy       | false
Mufutau   | Watkins   |
Naida     | Calderon  |
```

Jenis HLLSKETCH

Gunakan tipe data HLLSKETCH untuk HyperLogLog sketsa. Amazon Redshift mendukung representasi HyperLogLog sketsa yang jarang atau padat. Sketsa mulai jarang dan beralih ke padat ketika format padat lebih efisien untuk meminimalkan jejak memori yang digunakan.

Amazon Redshift secara otomatis mentransisi HyperLogLog sketsa yang jarang saat mengimpor, mengekspor, atau mencetak sketsa dalam format JSON berikut.

```
{"logm":15,"sparse":{"indices":[4878,9559,14523],"values":[1,2,1]}}
```

Amazon Redshift menggunakan representasi string dalam format Base64 untuk mewakili sketsa padat HyperLogLog .

Amazon Redshift menggunakan representasi string berikut dalam format Base64 untuk mewakili sketsa padat HyperLogLog .

```
"ABAABA..."
```

Ukuran maksimum objek HLLSKETCH adalah 24.580 byte bila digunakan dalam kompresi mentah.

Jenis SUPER

Gunakan tipe data SUPER untuk menyimpan data atau dokumen semi-terstruktur sebagai nilai. Meskipun Amazon Redshift mampu menyimpan nilai-nilai tersebut menggunakan VARCHAR, sebaiknya gunakan tipe data SUPER sebagai gantinya.

Semi-structured data tidak sesuai dengan struktur kaku dan tabular dari model data relasional yang digunakan dalam database SQL. Ini berisi tag yang mereferensikan entitas berbeda dalam data. Mereka dapat berisi nilai-nilai kompleks seperti array, struktur bersarang, dan struktur kompleks lainnya yang terkait dengan format serialisasi, seperti JSON. Tipe data SUPER adalah sekumpulan nilai array dan struktur tanpa skema yang mencakup semua jenis skalar Amazon Redshift lainnya.

Tipe data SUPER mendukung hingga 16 MB data untuk objek SUPER individu. Untuk informasi selengkapnya tentang tipe data SUPER, termasuk contoh penerapannya dalam tabel, lihatSemi-structured data dalam Amazon Redshift.

Amazon Redshift menyediakan dukungan bawaan untuk menyerap format data semi-terstruktur berikut menggunakan perintah COPY:

- JSON
- ARRAY
- TEXT
- CSV

Objek SUPER yang lebih besar dari 1MB hanya dapat dicerna dari format file berikut:

- Parquet
- JSON

- TEXT
- CSV

Tipe data SUPER memiliki properti berikut:

- Nilai skalar Amazon Redshift:
  - Sebuah nol
  - Sebuah boolean
  - Angka, seperti smallint, integer, bigint, decimal, atau float point (seperti float4 atau float8)
  - Nilai string, seperti varchar atau char
- Nilai kompleks:
  - Array nilai, termasuk skalar atau kompleks
  - Struktur, juga dikenal sebagai tupel atau objek, yaitu peta nama atribut dan nilai (skalar atau kompleks)

Salah satu dari dua jenis nilai kompleks mengandung skalar atau nilai kompleks mereka sendiri tanpa memiliki batasan keteraturan.

Pengkodean kompresi default untuk tipe data SUPER adalah ZSTD. Untuk informasi selengkapnya tentang pengkodean kompresi, lihatPengkodean kompresi.

Tipe data SUPER mendukung persistensi data semi-terstruktur dalam bentuk tanpa skema. Meskipun model data hierarkis dapat berubah, versi lama data dapat hidup berdampingan di kolom SUPER yang sama.

Amazon Redshift menggunakan PartiQL untuk mengaktifkan navigasi ke array dan struktur. Amazon Redshift juga menggunakan sintaks PartiQL untuk mengulangi array SUPER. Untuk informasi selengkapnya, lihat PartiQL — bahasa SQL-compatible query untuk Amazon Redshift.

Amazon Redshift menggunakan pengetikan dinamis untuk memproses data SUPER tanpa skema tanpa perlu mendeklarasikan tipe data sebelum Anda menggunakannya dalam kueri. Untuk informasi selengkapnya, lihat Pengetikan dinamis.

Anda dapat menerapkan kebijakan penyembunyian data dinamis ke nilai skalar pada jalur kolom tipe SUPER. Untuk informasi selengkapnya tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis. Untuk informasi tentang menggunakan penyembunyian data dinamis dengan tipe data SUPER, lihatMenggunakan penyembunyian data dinamis dengan jalur tipe data SUPER.

Kami menyarankan Anda mengatur opsi `r_enable_case_sensitive_super_attribute` konfigurasi ke true saat bekerja dengan data SUPER. Untuk informasi selengkapnya, lihat enable_case_sensitive_super_atribut.

Jenis VARBYTE

Gunakan kolom VARBYTE, VARBINARY, atau BINARY VARIING untuk menyimpan nilai biner panjang variabel dengan batas tetap.

```
varbyte [ (n) ]
```

Jumlah maksimum byte (n) dapat berkisar antara 1 — 16.777.216. Defaultnya adalah 64.000.

Beberapa contoh di mana Anda mungkin ingin menggunakan tipe data VARBYTE adalah sebagai berikut:

- Menggabungkan tabel pada kolom VARBYTE.
- Membuat tampilan terwujud yang berisi kolom VARBYTE. Refresh tambahan dari tampilan terwujud yang berisi kolom VARBYTE didukung. Namun, fungsi agregat selain COUNT, MIN, dan MAX dan GROUP BY pada kolom VARBYTE tidak mendukung penyegaran tambahan.

Untuk memastikan bahwa semua byte adalah karakter yang dapat dicetak, Amazon Redshift menggunakan format hex untuk mencetak nilai VARBYTE. Misalnya, SQL berikut mengubah string heksadesimal 6162 menjadi nilai biner. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal6162.

```
select from_hex('6162');

 from_hex
----------
 6162
```

Amazon Redshift mendukung casting antara VARBYTE dan tipe data berikut:

- CHAR
- VARCHAR
- SMALLINT
- INTEGER

- BIGINT

Saat casting dengan CHAR dan VARCHAR UTF-8 formatnya digunakan. Untuk informasi lebih lanjut tentang UTF-8 format, lihatTO_VARBYTE. Saat casting dari SMALLINT, INTEGER, dan BIGINT jumlah byte dari tipe data asli dipertahankan. Itu adalah dua byte untuk SMALLINT, empat byte untuk INTEGER, dan delapan byte untuk BIGINT.

Pernyataan SQL berikut melemparkan string VARCHAR ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`616263`.

```
select 'abc'::varbyte;

 varbyte
---------
 616263
```

Pernyataan SQL berikut memberikan nilai CHAR dalam kolom ke VARBYTE. Contoh ini membuat tabel dengan kolom CHAR (10) (c), menyisipkan nilai karakter yang lebih pendek dari panjang 10. Cast yang dihasilkan mempersembahkan hasil dengan karakter spasi (hex'20') ke ukuran kolom yang ditentukan. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal.

```
create table t (c char(10));
insert into t values ('aa'), ('abc');
select c::varbyte from t;
          c
----------------------
 6161202020202020202020
 6162632020202020202020
```

Pernyataan SQL berikut melemparkan string SMALLINT ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`0005`, yaitu dua byte atau empat karakter heksadesimal.

```
select 5::smallint::varbyte;

 varbyte
---------
```

```
 0005
```

Pernyataan SQL berikut memberikan INTEGER ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`00000005`, yaitu empat byte atau delapan karakter heksadesimal.

```
select 5::int::varbyte;

 varbyte
----------
 00000005
```

Pernyataan SQL berikut mentransmisikan BIGINT ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`0000000000000005`, yaitu delapan byte atau 16 karakter heksadesimal.

```
select 5::bigint::varbyte;

     varbyte
------------------
 0000000000000005
```

Fitur Amazon Redshift yang mendukung tipe data VARBYTE meliputi:

- Operator VARBYTE
- MEMBANTAH
- LENS
- Fungsi LENGTH
- OCTET_LENGTH
- Fungsi SUBSTRING
- DARI_HEX
- TO_HEX
- DARI_VARBYTE
- TO_VARBYTE
- GETBIT
- Memuat kolom tipe data VARBYTE

- Membongkar kolom tipe data VARBYTE

Batasan saat menggunakan tipe data VARBYTE dengan Amazon Redshift

Berikut ini adalah batasan saat menggunakan tipe data VARBYTE dengan Amazon Redshift:

- Amazon Redshift Spectrum mendukung tipe data VARBYTE hanya untuk file Parquet dan ORC.
- Editor kueri Amazon Redshift dan editor kueri Amazon Redshift v2 belum sepenuhnya mendukung tipe data VARBYTE. Oleh karena itu, gunakan klien SQL yang berbeda saat bekerja dengan ekspresi VARBYTE.

  Sebagai solusi untuk menggunakan editor kueri, jika panjang data Anda di atau di bawah 16.000.000 byte dan isinya valid UTF-8, Anda dapat mentransmisikan nilai VARBYTE ke VARCHAR, misalnya:

  ```
  select to_varbyte('6162', 'hex')::varchar;
  ```
- Anda tidak dapat menggunakan tipe data VARBYTE dengan fungsi yang ditentukan pengguna (UDF) Python atau Lambda.
- Anda tidak dapat membuat kolom HLLSKETCH dari kolom VARBYTE atau menggunakan APPROXIMATE COUNT DISTINCT pada kolom VARBYTE.
- Nilai VARBYTE yang lebih besar dari 1 MB hanya dapat dicerna dari format file berikut:
  - Parquet
  - Teks
  - Nilai yang dipisahkan koma (CSV)

Kompatibilitas dan konversi tipe

Berikut ini, Anda dapat menemukan diskusi tentang cara kerja aturan konversi tipe dan kompatibilitas tipe data di Amazon Redshift.

Kompatibilitas

Pencocokan tipe data dan pencocokan nilai literal dan konstanta dengan tipe data terjadi selama berbagai operasi database, termasuk yang berikut:

- Operasi bahasa manipulasi data (DML) pada tabel

- Kueri UNION, INTERSECT, dan EXCEPT
- Ekspresi CASE
- Evaluasi predikat, seperti LIKE dan IN
- Evaluasi fungsi SQL yang melakukan perbandingan atau ekstraksi data
- Perbandingan dengan operator matematika

Hasil operasi ini tergantung pada aturan konversi tipe dan kompatibilitas tipe data. Kompati bilitas menyiratkan bahwa pencocokan satu-ke-satu dari nilai tertentu dan tipe data tertentu tidak selalu diperlukan. Karena beberapa tipe data kompatibel, konversi implisit, atau paksaan, dimungkinkan (untuk informasi lebih lanjut, lihatJenis konversi implisit). Ketika tipe data tidak kompatibel, terkadang Anda dapat mengonversi nilai dari satu tipe data ke tipe data lainnya dengan menggunakan fungsi konversi eksplisit.

Kompatibilitas umum dan aturan konversi

Perhatikan aturan kompatibilitas dan konversi berikut:

- Secara umum, tipe data yang termasuk dalam kategori tipe yang sama (seperti tipe data numerik yang berbeda) kompatibel dan dapat dikonversi secara implisit.

  Misalnya, dengan konversi implisit Anda dapat memasukkan nilai desimal ke dalam kolom integer. Desimal dibulatkan untuk menghasilkan bilangan bulat. Atau Anda dapat mengekstrak nilai numerik, seperti`2008`, dari tanggal dan memasukkan nilai itu ke dalam kolom integer.
- Tipe data numerik memberlakukan kondisi luapan yang terjadi saat Anda mencoba memasukkan nilai di luar jangkauan. Misalnya, nilai desimal dengan presisi 5 tidak cocok dengan kolom desimal yang didefinisikan dengan presisi 4. Bilangan bulat atau seluruh bagian desimal tidak pernah terpotong; Namun, bagian pecahan dari desimal dapat dibulatkan ke atas atau ke bawah, sebagaimana mestinya. Namun, hasil pemertian eksplisit nilai yang dipilih dari tabel tidak dibulatkan.
- Berbagai jenis string karakter kompatibel; string kolom VARCHAR yang berisi data byte tunggal dan string kolom CHAR sebanding dan secara implisit dapat dikonversi. String VARCHAR yang berisi data multibyte tidak sebanding. Juga, Anda dapat mengonversi string karakter ke tanggal, waktu, stempel waktu, atau nilai numerik jika string adalah nilai literal yang sesuai; spasi depan atau akhir diabaikan. Sebaliknya, Anda dapat mengonversi tanggal, waktu, stempel waktu, atau nilai numerik ke string karakter panjang tetap atau panjang variabel.

> **Note**
>
> String karakter yang ingin Anda tuang ke tipe numerik harus berisi representasi karakter dari angka. Misalnya, Anda dapat mentransmisikan string `'1.0'` atau `'5.9'` nilai desimal, tetapi Anda tidak dapat mentransmisikan string `'ABC'` ke jenis numerik apa pun.

- Jika Anda membandingkan nilai DECIMAL dengan string karakter, Amazon Redshift mencoba mengonversi string karakter menjadi nilai DECIMAL. Saat membandingkan semua nilai numerik lainnya dengan string karakter, nilai numerik dikonversi ke string karakter. Untuk menerapkan konversi yang berlawanan (misalnya, mengubah string karakter menjadi bilangan bulat, atau mengubah nilai DECIMAL menjadi string karakter), gunakan fungsi eksplisit, seperti PEMERAN.
- Untuk mengonversi nilai DECIMAL atau NUMERIC 64-bit ke presisi yang lebih tinggi, Anda harus menggunakan fungsi konversi eksplisit seperti fungsi CAST atau CONVERT.
- Saat mengonversi DATE atau TIMESTAMP menjadi TIMESTAMPTZ, atau mengkonversi TIME ke TIMETZ, zona waktu diatur ke zona waktu sesi saat ini. Zona waktu sesi adalah UTC secara default. Untuk informasi selengkapnya tentang menyetel zona waktu sesi, lihat timezone.
- Demikian pula, TIMESTAMPTZ dikonversi ke DATE, TIME, atau TIMESTAMP berdasarkan zona waktu sesi saat ini. Zona waktu sesi adalah UTC secara default. Setelah konversi, informasi zona waktu dijatuhkan.
- String karakter yang mewakili stempel waktu dengan zona waktu yang ditentukan dikonversi ke TIMESTAMPTZ menggunakan zona waktu sesi saat ini, yang merupakan UTC secara default. Demikian juga, string karakter yang mewakili waktu dengan zona waktu yang ditentukan dikonversi ke TIMETZ menggunakan zona waktu sesi saat ini, yang merupakan UTC secara default.

Jenis konversi implisit

Ada dua jenis konversi implisit:

- Konversi implisit dalam tugas, seperti menyetel nilai dalam perintah INSERT atau UPDATE.
- Konversi implisit dalam ekspresi, seperti melakukan perbandingan dalam klausa WHERE.

Tabel berikut mencantumkan tipe data yang dapat dikonversi secara implisit dalam tugas atau ekspresi. Anda juga dapat menggunakan fungsi konversi eksplisit untuk melakukan konversi ini.

Dari jenis	Untuk mengetik
BIG (INT8)	BOOLEAN
	CHAR
	DESIMAL (NUMERIK)
	PRESISI GANDA (FLOAT8)
	BILANGAN BULAT (INT, INT4)
	NYATA (FLOAT4)
	SMALL (INT2)
	VARCHAR
CHAR	VARCHAR
DATE	CHAR
	VARCHAR
	TIMESTAMP
	TIMESTAMPTZ
DESIMAL (NUMERIK)	BIG (INT8)
	CHAR
	PRESISI GANDA (FLOAT8)
	BILANGAN BULAT (INT, INT4)
	NYATA (FLOAT4)
	SMALL (INT2)
	VARCHAR

Dari jenis	Untuk mengetik
PRESISI GANDA (FLOAT8)	BIG (INT8)
	CHAR
	DESIMAL (NUMERIK)
	BILANGAN BULAT (INT, INT4)
	NYATA (FLOAT4)
	SMALL (INT2)
	VARCHAR
BILANGAN BULAT (INT, INT4)	BIG (INT8)
	BOOLEAN
	CHAR
	DESIMAL (NUMERIK)
	PRESISI GANDA (FLOAT8)
	NYATA (FLOAT4)
	SMALL (INT2)
	VARCHAR
NYATA (FLOAT4)	BIG (INT8)
	CHAR
	DESIMAL (NUMERIK)
	BILANGAN BULAT (INT, INT4)
	SMALL (INT2)

Dari jenis	Untuk mengetik
	VARCHAR
SMALL (INT2)	BIG (INT8)
	BOOLEAN
	CHAR
	DESIMAL (NUMERIK)
	PRESISI GANDA (FLOAT8)
	BILANGAN BULAT (INT, INT4)
	NYATA (FLOAT4)
	VARCHAR
TIMESTAMP	CHAR
	DATE
	VARCHAR
	TIMESTAMPTZ
	TIME
TIMESTAMPTZ	CHAR
	DATE
	VARCHAR
	TIMESTAMP
	TIMEZ
TIME	VARCHAR

Dari jenis	Untuk mengetik
	TIMETZ
	INTERVAL HARI KE DETIK
TIMEZ	VARCHAR
	TIME
GEOMETRY	GEOGRAPHY
GEOGRAPHY	GEOMETRY

> **Note**
>
> Konversi implisit antara TIMESTAMPTZ, TIMESTAMP, DATE, TIME, TIMETZ, atau string karakter menggunakan zona waktu sesi saat ini. Untuk informasi tentang pengaturan zona waktu saat ini, lihattimezone.
> Tipe data GEOMETRY dan GEOGRAPHY tidak dapat dikonversi secara implisit ke tipe data lain, kecuali satu sama lain. Untuk informasi selengkapnya, lihat Fungsi CAST.
> Tipe data VARBYTE tidak dapat dikonversi secara implisit ke tipe data lainnya. Untuk informasi selengkapnya, lihat Fungsi CAST.

Menggunakan pengetikan dinamis untuk tipe data SUPER

Amazon Redshift menggunakan pengetikan dinamis untuk memproses data SUPER tanpa skema tanpa perlu mendeklarasikan tipe data sebelum Anda menggunakannya dalam kueri Anda. Pengetikan dinamis menggunakan hasil navigasi ke kolom data SUPER tanpa harus secara eksplisit menempatkannya ke jenis Amazon Redshift. Untuk informasi selengkapnya tentang menggunakan pengetikan dinamis untuk tipe data SUPER, lihatPengetikan dinamis.

Anda dapat mentransmisikan nilai SUPER ke dan dari tipe data lain dengan beberapa pengecualian. Untuk informasi selengkapnya, lihat Batasan.

Urutan kolasi

Amazon Redshift tidak mendukung urutan kolasi khusus lokal atau yang ditentukan pengguna. Secara umum, hasil predikat apa pun dalam konteks apa pun dapat dipengaruhi oleh kurangnya aturan khusus lokal untuk menyortir dan membandingkan nilai data. Misalnya, ekspresi ORDER BY dan fungsi seperti MIN, MAX, dan RANK mengembalikan hasil berdasarkan urutan UTF8 biner dari data yang tidak memperhitungkan karakter khusus lokal.

Ekspresi

Topik

- Ekspresi sederhana
- Ekspresi majemuk
- Daftar ekspresi
- Subkueri skalar
- Ekspresi fungsi

Ekspresi adalah kombinasi dari satu atau lebih nilai, operator, atau fungsi yang mengevaluasi nilai. Tipe data dari suatu ekspresi umumnya adalah komponennya.

Ekspresi sederhana

Ekspresi sederhana adalah salah satu dari berikut ini:

- Nilai konstan atau literal
- Nama kolom atau referensi kolom
- Fungsi skalar
- Fungsi agregat (set)
- Fungsi jendela
- Sebuah subkueri skalar

Contoh ekspresi sederhana meliputi:

```
5+12
dateid
```

```
sales.qtysold * 100
sqrt (4)
max (qtysold)
(select max (qtysold) from sales)
```

Ekspresi majemuk

Ekspresi majemuk adalah serangkaian ekspresi sederhana yang digabungkan oleh operator aritmatika. Ekspresi sederhana yang digunakan dalam ekspresi majemuk harus mengembalikan nilai numerik.

Sintaksis

```
expression
operator
expression | (compound_expression)
```

Argumen

ekspresi

Ekspresi sederhana yang mengevaluasi nilai.

operator

Ekspresi aritmatika majemuk dapat dibangun menggunakan operator berikut, dalam urutan prioritas ini:

- (): tanda kurung untuk mengontrol urutan evaluasi
- +, -: positif dan negatif sign/operator
- ^, |/, ||/: eksponensiasi, akar kuadrat, akar kubus
- *,/,%: perkalian, pembagian, dan operator modulo
- @: nilai absolut
- +, -: penambahan dan pengurangan
- &, |, #, ~, <<, >>: AND, OR, NOT, geser kiri, geser kanan operator bitwise
- ||: penggabungan

(ekspresi gabungan)

Ekspresi majemuk dapat bersarang menggunakan tanda kurung.

Contoh

Contoh ekspresi majemuk meliputi yang berikut ini.

```
('SMITH' || 'JONES')
sum(x) / y
sqrt(256) * avg(column)
rank() over (order by qtysold) / 100
(select (pricepaid - commission) from sales where dateid = 1882) * (qtysold)
```

Beberapa fungsi juga dapat bersarang dalam fungsi lain. Misalnya, fungsi skalar apa pun dapat bersarang dalam fungsi skalar lain. Contoh berikut mengembalikan jumlah nilai absolut dari sekumpulan angka:

```
sum(abs(qtysold))
```

Fungsi jendela tidak dapat digunakan sebagai argumen untuk fungsi agregat atau fungsi jendela lainnya. Ekspresi berikut akan mengembalikan kesalahan:

```
avg(rank() over (order by qtysold))
```

Fungsi jendela dapat memiliki fungsi agregat bersarang. Ekspresi berikut menjumlahkan kumpulan nilai dan kemudian memberi peringkat:

```
rank() over (order by sum(qtysold))
```

Daftar ekspresi

Daftar ekspresi adalah kombinasi ekspresi, dan dapat muncul dalam kondisi keanggotaan dan perbandingan (klausa WHERE) dan dalam klausa GROUP BY.

Sintaksis

```
expression , expression , ... | (expression, expression, ...)
```

Argumen

ekspresi

Ekspresi sederhana yang mengevaluasi nilai. Daftar ekspresi dapat berisi satu atau lebih ekspresi yang dipisahkan koma atau satu atau lebih set ekspresi yang dipisahkan koma. Ketika ada

beberapa set ekspresi, setiap himpunan harus berisi jumlah ekspresi yang sama, dan dipisahkan oleh tanda kurung. Jumlah ekspresi di setiap set harus sesuai dengan jumlah ekspresi sebelum operator dalam kondisi.

Contoh

Berikut ini adalah contoh daftar ekspresi dalam kondisi:

```
(1, 5, 10)
('THESE', 'ARE', 'STRINGS')
(('one', 'two', 'three'), ('blue', 'yellow', 'green'))
```

Jumlah ekspresi di setiap himpunan harus sesuai dengan angka di bagian pertama pernyataan:

```
select * from venue
where (venuecity, venuestate) in (('Miami', 'FL'), ('Tampa', 'FL'))
order by venueid;

venueid |        venuename        | venuecity | venuestate | venueseats
---------+-------------------------+-----------+------------+------------
28 | American Airlines Arena | Miami     | FL         |          0
54 | St. Pete Times Forum    | Tampa     | FL         |          0
91 | Raymond James Stadium   | Tampa     | FL         |      65647
(3 rows)
```

Subkueri skalar

Subkueri skalar adalah kueri SELECT reguler dalam tanda kurung yang mengembalikan tepat satu nilai: satu baris dengan satu kolom. Kueri dijalankan dan nilai yang dikembalikan digunakan dalam kueri luar. Jika subquery mengembalikan nol baris, nilai ekspresi subquery adalah null. Jika mengembalikan lebih dari satu baris, Amazon Redshift mengembalikan kesalahan. Subkueri dapat merujuk ke variabel dari kueri induk, yang akan bertindak sebagai konstanta selama satu pemanggilan subkueri.

Anda dapat menggunakan subquery skalar di sebagian besar pernyataan yang memanggil ekspresi. Subkueri skalar bukanlah ekspresi yang valid dalam kasus berikut:

- Sebagai nilai default untuk ekspresi
- Dalam klausa GROUP BY dan HAVING

Contoh

Subkueri berikut menghitung harga rata-rata yang dibayarkan per penjualan sepanjang tahun 2008, kemudian kueri luar menggunakan nilai itu dalam output untuk membandingkan dengan harga rata-rata per penjualan per kuartal:

```
select qtr, avg(pricepaid) as avg_saleprice_per_qtr,
(select avg(pricepaid)
from sales join date on sales.dateid=date.dateid
where year = 2008) as avg_saleprice_yearly
from sales join date on sales.dateid=date.dateid
where year = 2008
group by qtr
order by qtr;
qtr  | avg_saleprice_per_qtr | avg_saleprice_yearly
-------+-----------------------+----------------------
1     |                647.64 |               642.28
2     |                646.86 |               642.28
3     |                636.79 |               642.28
4     |                638.26 |               642.28
(4 rows)
```

Ekspresi fungsi

Sintaksis

Apa pun bawaan dapat digunakan sebagai ekspresi. Sintaks untuk panggilan fungsi adalah nama fungsi diikuti oleh daftar argumennya dalam tanda kurung.

```
function ( [expression [, expression...]] )
```

Argumen

fungsi

Fungsi bawaan apa pun. Untuk beberapa contoh fungsi, lihatReferensi fungsi SQL.

ekspresi

Ekspresi apa pun yang cocok dengan tipe data dan jumlah parameter yang diharapkan oleh fungsi.

Contoh

```
abs (variable)
select avg (qtysold + 3) from sales;
select dateadd (day,30,caldate) as plus30days from date;
```

Ketentuan

Topik

- Sintaksis
- Kondisi perbandingan
- Kondisi logis
- Pattern-matching kondisi
- BETWEEN kondisi rentang
- Kondisi nol
- Kondisi ADA
- Dalam kondisi

Kondisi adalah pernyataan dari satu atau lebih ekspresi dan operator logis yang mengevaluasi menjadi benar, salah, atau tidak diketahui. Kondisi juga kadang-kadang disebut sebagai predikat.

> **Note**
>
> Semua perbandingan string dan kecocokan pola LIKE peka huruf besar-kecil. Misalnya, 'A' dan 'a' tidak cocok. Namun, Anda dapat melakukan pencocokan pola yang tidak peka huruf besar/kecil dengan menggunakan predikat ILIKE.

Sintaksis

```
comparison_condition
| logical_condition
| range_condition
| pattern_matching_condition
| null_condition
| EXISTS_condition
```

```
| IN_condition
```

Kondisi perbandingan

Kondisi perbandingan menyatakan hubungan logis antara dua nilai. Semua kondisi perbandingan adalah operator biner dengan tipe pengembalian Boolean. Amazon Redshift mendukung operator perbandingan yang dijelaskan dalam tabel berikut:

Operator	Sintaksis	Deskripsi
<	`a < b`	Nilai a kurang dari nilai b.
>	`a > b`	Nilai a lebih besar dari nilai b.
<=	`a <= b`	Nilai a kurang dari atau sama dengan nilai b.
>=	`a >= b`	Nilai a lebih besar dari atau sama dengan nilai b.
=	`a = b`	Nilai a sama dengan nilai b.
<> atau! =	`a <> b or a != b`	Nilai a tidak sama dengan nilai b.
APA SAJA | BEBERAPA	`a = ANY(subquery)`	Nilai a sama dengan nilai apa pun yang dikembali kan oleh subquery.
SEMUA	`a <> ALL or != ALL (subquery))`	Nilai a tidak sama dengan nilai apa pun yang dikembalikan oleh subquery.
BENAR | SALAH | TIDAK DIKETAHUI	`a IS TRUE`	Nilai a adalah Boolean TRUE.

Catatan penggunaan

= APA SAJA | BEBERAPA

Kata kunci ANY dan SOME identik dengan kondisi IN, dan mengembalikan true jika perbandingan benar untuk setidaknya satu nilai yang dikembalikan oleh subquery yang

mengembalikan satu atau lebih nilai. Amazon Redshift hanya mendukung kondisi = (sama) untuk ANY dan SOME. Kondisi ketidaksetaraan tidak didukung.

> **Note**
>
> Predikat ALL tidak didukung.

<> SEMUA

Kata kunci ALL identik dengan NOT IN (lihat Dalam kondisi kondisi) dan mengembalikan true jika ekspresi tidak termasuk dalam hasil subquery. Amazon Redshift hanya mendukung <> atau! = (tidak sama) kondisi untuk SEMUA. Kondisi perbandingan lainnya tidak didukung.

ADALAH TRUE/FALSE/UNKNOWN

Non-zero nilai sama dengan TRUE, 0 sama dengan FALSE, dan null sama dengan UNKNOWN. Lihat tipe Tipe Boolean datanya.

Contoh

Berikut adalah beberapa contoh sederhana dari kondisi perbandingan:

```
a = 5
a < b
min(x) >= 5
qtysold = any (select qtysold from sales where dateid = 1882
```

Kueri berikut mengembalikan tempat dengan lebih dari 10000 kursi dari tabel VENUE:

```
select venueid, venuename, venueseats from venue
where venueseats > 10000
order by venueseats desc;

venueid |           venuename            | venueseats
---------+--------------------------------+------------
83 | FedExField                    |      91704
 6 | New York Giants Stadium       |      80242
79 | Arrowhead Stadium             |      79451
78 | INVESCO Field                 |      76125
69 | Dolphin Stadium               |      74916
```

```
67 | Ralph Wilson Stadium           |      73967
76 | Jacksonville Municipal Stadium |      73800
89 | Bank of America Stadium        |      73298
72 | Cleveland Browns Stadium       |      73200
86 | Lambeau Field                  |      72922
...
(57 rows)
```

Contoh ini memilih pengguna (USERID) dari tabel USERS yang menyukai musik rock:

```
select userid from users where likerock = 't' order by 1 limit 5;

userid
--------
3
5
6
13
16
(5 rows)
```

Contoh ini memilih pengguna (USERID) dari tabel USERS di mana tidak diketahui apakah mereka menyukai musik rock:

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;

firstname | lastname | likerock
----------+----------+----------
Rafael    | Taylor   |
Vladimir  | Humphrey |
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett  | Mayer    |
(10 rows
```

Contoh dengan kolom TIME

Tabel contoh berikut TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut mengekstrak jam dari setiap timetz_val.

```
select time_val from time_test where time_val < '3:00';
   time_val
---------------
 00:00:00.5550
 00:58:00
```

Contoh berikut membandingkan dua literal waktu.

```
select time '18:25:33.123456' = time '18:25:33.123456';
 ?column?
----------
 t
```

Contoh dengan kolom TIMETZ

Tabel contoh berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut hanya memilih nilai TIMETZ kurang dari`3:00:00 UTC`. Perbandingan dibuat setelah mengubah nilai ke UTC.

```
select timetz_val from timetz_test where timetz_val < '3:00:00 UTC';

   timetz_val
---------------
 00:00:00.5550+00
```

Contoh berikut membandingkan dua literal TIMETZ. Zona waktu diabaikan untuk perbandingan.

```
select time '18:25:33.123456 PST' < time '19:25:33.123456 EST';

 ?column?
----------
 t
```

Kondisi logis

Kondisi logis menggabungkan hasil dari dua kondisi untuk menghasilkan satu hasil. Semua kondisi logis adalah operator biner dengan tipe pengembalian Boolean.

Sintaksis

```
expression
{ AND | OR }
expression
NOT expression
```

Kondisi logis menggunakan logika Boolean tiga nilai di mana nilai nol mewakili hubungan yang tidak diketahui. Tabel berikut menjelaskan hasil untuk kondisi logis, di mana `E1` dan `E2` mewakili ekspresi:

E1	E2	E1 DAN E2	E1 ATAU E2	BUKAN E2
BETUL	BETUL	BETUL	BETUL	SALAH
BETUL	SALAH	SALAH	BETUL	BETUL
BETUL	TIDAK DIKETAHUI	TIDAK DIKETAHUI	BETUL	TIDAK DIKETAHUI

E1	E2	E1 DAN E2	E1 ATAU E2	BUKAN E2
SALAH	BETUL	SALAH	BETUL	
SALAH	SALAH	SALAH	SALAH	
SALAH	TIDAK DIKETAHUI	SALAH	TIDAK DIKETAHUI	
TIDAK DIKETAHUI	BETUL	TIDAK DIKETAHUI	BETUL	
TIDAK DIKETAHUI	SALAH	SALAH	TIDAK DIKETAHUI	
TIDAK DIKETAHUI	TIDAK DIKETAHUI	TIDAK DIKETAHUI	TIDAK DIKETAHUI	

Operator NOT dievaluasi sebelum AND, dan operator AND dievaluasi sebelum operator OR. Setiap tanda kurung yang digunakan dapat mengesampingkan urutan evaluasi default ini.

Contoh

Contoh berikut mengembalikan USERID dan USERNAME dari tabel USERS di mana pengguna menyukai Las Vegas dan olahraga:

```
select userid, username from users
where likevegas = 1 and likesports = 1
order by userid;

userid | username
--------+----------
1 | JSG99FHE
67 | TWU10MZT
87 | DUF19VXU
92 | HYP36WEQ
109 | FPL38HZK
120 | DMJ24GUZ
123 | QZR22XGQ
130 | ZQC82ALK
133 | LBN45WCH
```

```
144 | UCX04JKN
165 | TEY68OEB
169 | AYQ83HGO
184 | TVX65AZX
...
(2128 rows)
```

Contoh berikutnya mengembalikan USERID dan USERNAME dari tabel USERS di mana pengguna menyukai Las Vegas, atau olahraga, atau keduanya. Query ini mengembalikan semua output dari contoh sebelumnya ditambah pengguna yang hanya menyukai Las Vegas atau olahraga.

```
select userid, username from users
where likevegas = 1 or likesports = 1
order by userid;

userid | username
--------+----------
1 | JSG99FHE
2 | PGL08LJI
3 | IFT66TXU
5 | AEB55QTM
6 | NDQ15VBM
9 | MSD36KVR
10 | WKW41AIW
13 | QTF33MCG
15 | OWU78MTR
16 | ZMG93CDD
22 | RHT62AGI
27 | KOY02CVE
29 | HUH27PKK
...
(18968 rows)
```

Kueri berikut menggunakan tanda kurung di sekitar `OR` kondisi untuk menemukan tempat di New York atau California tempat Macbeth dilakukan:

```
select distinct venuename, venuecity
from venue join event on venue.venueid=event.venueid
where (venuestate = 'NY' or venuestate = 'CA') and eventname='Macbeth'
order by 2,1;

venuename                |   venuecity
```

```
----------------------------------------+---------------
Geffen Playhouse                        | Los Angeles
Greek Theatre                           | Los Angeles
Royce Hall                              | Los Angeles
American Airlines Theatre               | New York City
August Wilson Theatre                   | New York City
Belasco Theatre                         | New York City
Bernard B. Jacobs Theatre               | New York City
...
```

Menghapus tanda kurung dalam contoh ini mengubah logika dan hasil kueri.

Contoh berikut menggunakan NOT operator:

```
select * from category
where not catid=1
order by 1;

catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
2 | Sports   | NHL       | National Hockey League
3 | Sports   | NFL       | National Football League
4 | Sports   | NBA       | National Basketball Association
5 | Sports   | MLS       | Major League Soccer
...
```

Contoh berikut menggunakan NOT kondisi diikuti oleh AND kondisi:

```
select * from category
where (not catid=1) and catgroup='Sports'
order by catid;

catid | catgroup | catname |             catdesc
-------+----------+---------+---------------------------------
2 | Sports   | NHL     | National Hockey League
3 | Sports   | NFL     | National Football League
4 | Sports   | NBA     | National Basketball Association
5 | Sports   | MLS     | Major League Soccer
(4 rows)
```

Pattern-matching kondisi

Topik

- SUKA
- SIMILAR TO
- Operator POSIX

Operator pencocokan pola mencari string untuk pola yang ditentukan dalam ekspresi kondisional dan mengembalikan true atau false tergantung pada apakah ia menemukan kecocokan. Amazon Redshift menggunakan tiga metode untuk pencocokan pola:

- Ekspresi LIKE

  Operator LIKE membandingkan ekspresi string, seperti nama kolom, dengan pola yang menggunakan karakter wildcard % (persen) dan _ (garis bawah). Pencocokan pola LIKE selalu mencakup seluruh string. LIKE melakukan kecocokan peka huruf besar/kecil dan ILIKE melakukan kecocokan yang tidak peka huruf besar/kecil.
- MIRIP DENGAN ekspresi reguler

  Operator SIMILAR TO mencocokkan ekspresi string dengan pola ekspresi reguler standar SQL, yang dapat mencakup satu set metakarakter pencocokan pola yang mencakup dua yang didukung oleh operator LIKE. SIMILAR TO cocok dengan seluruh string dan melakukan kecocokan peka huruf besar-kecil.
- POSIX-style ekspresi reguler

  Ekspresi reguler POSIX menyediakan sarana yang lebih kuat untuk pencocokan pola daripada operator LIKE dan SIMILAR TO. Pola ekspresi reguler POSIX dapat mencocokkan bagian mana pun dari string dan melakukan kecocokan peka huruf besar-kecil.

Pencocokan ekspresi reguler, menggunakan operator SIMILAR TO atau POSIX, mahal secara komputasi. Sebaiknya gunakan LIKE bila memungkinkan, terutama saat memproses sejumlah besar baris. Misalnya, kueri berikut secara fungsional identik, tetapi kueri yang menggunakan LIKE berjalan beberapa kali lebih cepat daripada kueri yang menggunakan ekspresi reguler:

```
select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die%';
```

SUKA

Operator LIKE membandingkan ekspresi string, seperti nama kolom, dengan pola yang menggunakan karakter wildcard% (persen) dan _ (garis bawah). Pencocokan pola LIKE selalu mencakup seluruh string. Untuk mencocokkan urutan di mana saja dalam string, pola harus dimulai dan diakhiri dengan tanda persen.

LIKE peka huruf besar/kecil; ILIKE tidak peka huruf besar/kecil.

Sintaksis

```
expression [ NOT ] LIKE | ILIKE pattern [ ESCAPE 'escape_char' ]
```

Argumen

ekspresi

Eksp UTF-8 resi karakter yang valid, seperti nama kolom.

SUKA | ILIKE

LIKE melakukan kecocokan pola peka huruf besar-kecil. ILIKE melakukan pencocokan pola yang tidak peka huruf besar/kecil untuk karakter single-byte UTF-8 (ASCII). Untuk melakukan pencocokan pola yang tidak peka huruf besar/kecil untuk karakter multibyte, gunakan fungsi LOWER pada ekspresi dan pola dengan kondisi LIKE.

Berbeda dengan predikat perbandingan, seperti = dan <>, predikat LIKE dan ILIKE tidak secara implisit mengabaikan spasi akhir. Untuk mengabaikan spasi akhir, gunakan RTRIM atau secara eksplisit memberikan kolom CHAR ke VARCHAR.

~~Operator setara dengan LIKE, dan ~~* setara dengan ILIKE. Juga !~~* operator !~~ dan setara dengan NOT LIKE dan NOT ILIKE.

pola

Ekspresi UTF-8 karakter yang valid dengan pola yang akan dicocokkan.

escape_char

Ekspresi karakter yang akan lolos dari karakter metakarakter dalam pola. Defaultnya adalah dua garis miring terbalik ('\\').

Jika pola tidak mengandung metakarakter, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator yang sama.

Salah satu ekspresi karakter dapat berupa tipe data CHAR atau VARCHAR. Jika berbeda, Amazon Redshift mengon versi pola ke tipe data ekspresi.

LIKE mendukung metakarakter pencocokan pola berikut:

Operator	Deskripsi
%	Cocokkan urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.

Contoh

Tabel berikut menunjukkan contoh pencocokan pola menggunakan LIKE:

Ekspresi	Pengembalian
'abc' LIKE 'abc'	True
'abc' LIKE 'a%'	Benar
'abc' LIKE '_B_'	Salah
'abc' ILIKE '_B_'	Benar
'abc' LIKE 'c%'	False

Contoh berikut menemukan semua kota yang namanya dimulai dengan “E”:

```
select distinct city from users
where city like 'E%' order by city;
city
---------------
East Hartford
East Lansing
East Rutherford
```

```
East St. Louis
Easthampton
Easton
Eatontown
Eau Claire
...
```

Contoh berikut menemukan pengguna yang nama belakangnya berisi “sepuluh”:

```
select distinct lastname from users
where lastname like '%ten%' order by lastname;
lastname
-------------
Christensen
Wooten
...
```

Contoh berikut menunjukkan bagaimana untuk mencocokkan beberapa pola.

```
select distinct lastname from tickit.users
where lastname like 'Chris%' or lastname like '%Wooten' order by lastname;
lastname
-------------
Christensen
Christian
Wooten
...
```

Contoh berikut menemukan kota yang karakter ketiga dan keempatnya adalah “ea”. Perintah menggunakan ILIKE untuk menunjukkan ketidakpekaan huruf besar/kecil:

```
select distinct city from users where city ilike '__EA%' order by city;
city
-------------
Brea
Clearwater
Great Falls
Ocean City
Olean
Wheaton
(6 rows)
```

Contoh berikut menggunakan string escape default (\\) untuk mencari string yang menyertakan “start_” (teks `start` diikuti oleh garis bawah): _

```
select tablename, "column" from pg_table_def
where "column" like '%start\\_%'
limit 5;

     tablename     |    column
-------------------+---------------
 stl_s3client      | start_time
 stl_tr_conflict   | xact_start_ts
 stl_undone        | undo_start_ts
 stl_unload_log    | start_time
 stl_vacuum_detail | start_row
(5 rows)
```

Contoh berikut menentukan '^' sebagai karakter escape, kemudian menggunakan karakter escape untuk mencari string yang menyertakan “start_” (teks `start` diikuti oleh garis bawah_):

```
select tablename, "column" from pg_table_def
where "column" like '%start^_%' escape '^'
limit 5;

     tablename     |    column
-------------------+---------------
 stl_s3client      | start_time
 stl_tr_conflict   | xact_start_ts
 stl_undone        | undo_start_ts
 stl_unload_log    | start_time
 stl_vacuum_detail | start_row
(5 rows)
```

Contoh berikut menggunakan ~~* operator untuk melakukan pencarian case-insensitive (ILIKE) untuk kota yang dimulai dengan “Ag”.

```
select distinct city from users where city ~~* 'Ag%' order by city;

city
------------
Agat
Agawam
Agoura Hills
```

```
Aguadilla
```

SIMILAR TO

Operator SIMILAR TO cocok dengan ekspresi string, seperti nama kolom, dengan pola ekspresi reguler standar SQL. Pola ekspresi reguler SQL dapat mencakup satu set metakarakter pencocokan pola, termasuk dua yang didukung oleh operator. SUKA

Operator SIMILAR TO mengembalikan true hanya jika polanya cocok dengan seluruh string, tidak seperti perilaku ekspresi reguler POSIX, di mana pola dapat mencocokkan bagian mana pun dari string.

MIRIP TO melakukan kecocokan peka huruf besar-kecil.

> **Note**
>
> Pencocokan ekspresi reguler menggunakan SIMILAR TO mahal secara komputasi. Sebaiknya gunakan LIKE bila memungkinkan, terutama saat memproses sejumlah besar baris. Misalnya, kueri berikut secara fungsional identik, tetapi kueri yang menggunakan LIKE berjalan beberapa kali lebih cepat daripada kueri yang menggunakan ekspresi reguler:
>
> ```
> select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
> select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die
> %';
> ```

Sintaksis

```
expression [ NOT ] SIMILAR TO pattern [ ESCAPE 'escape_char' ]
```

Argumen

ekspresi

Eksp UTF-8 resi karakter yang valid, seperti nama kolom.

SIMILAR TO

SIMILAR TO melakukan kecocokan pola peka huruf besar-kecil untuk seluruh string dalam ekspresi.

pola

Eksp UTF-8 resi karakter yang valid mewakili pola ekspresi reguler standar SQL.

escape_char

Ekspresi karakter yang akan lolos dari metakarakter dalam pola. Defaultnya adalah dua garis miring terbalik ('\\').

Jika pola tidak mengandung metakarakter, maka pola hanya mewakili string itu sendiri.

Salah satu ekspresi karakter dapat berupa tipe data CHAR atau VARCHAR. Jika berbeda, Amazon Redshift mengon versi pola ke tipe data ekspresi.

SIMILAR TO mendukung metakarakter pencocokan pola berikut:

Operator	Deskripsi
%	Cocokkan urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.
|	Menunjukkan pergantian (salah satu dari dua alternatif).
*	Ulangi item sebelumnya nol atau lebih kali.
+	Ulangi item sebelumnya satu kali atau lebih.
?	Ulangi item sebelumnya nol atau satu kali.
{m}	Ulangi item sebelumnya persis m kali.
{m,}	Ulangi item sebelumnya m atau lebih kali.
{m,n}	Ulangi item sebelumnya setidaknya m dan tidak lebih dari n kali.
()	Kurung mengelompokkan item menjadi satu item logis.
[...]	Ekspresi tanda kurung menentukan kelas karakter, seperti dalam ekspresi reguler POSIX.

Contoh

Tabel berikut menunjukkan contoh pencocokan pola menggunakan SIMILAR TO:

Ekspresi	Pengembalian
`'abc' SIMILAR TO 'abc'`	True
`'abc' SIMILAR TO '_b_'`	Benar
`'abc' SIMILAR TO '_A_'`	Salah
`'abc' SIMILAR TO '%(b|d)%'`	Benar
`'abc' SIMILAR TO '(b|c)%'`	Salah
`'AbcAbcdefgefg12efgefg12' SIMILAR TO '((Ab)?c)+d((efg)+(12))+'`	Benar
`'aaaaaab11111xy' SIMILAR TO 'a{6}_[0-9]{5}(x|y){2}'`	Benar
`'$0.87' SIMILAR TO '$[0-9]+(.[0-9][0-9])?'`	True

Contoh berikut menemukan kota yang namanya mengandung “E” atau “H”:

```
SELECT DISTINCT city FROM users
WHERE city SIMILAR TO '%E%|%H%' ORDER BY city LIMIT 5;

      city
-----------------
 Agoura Hills
 Auburn Hills
 Benton Harbor
 Beverly Hills
 Chicago Heights
```

Contoh berikut menggunakan string escape default ('\\') untuk mencari string yang menyertakan "_“:

```
SELECT tablename, "column" FROM pg_table_def
WHERE "column" SIMILAR TO '%start\\_%'
ORDER BY tablename, "column" LIMIT 5;

        tablename         |       column
--------------------------+---------------------
 stcs_abort_idle          | idle_start_time
 stcs_abort_idle          | txn_start_time
 stcs_analyze_compression | start_time
 stcs_auto_worker_levels  | start_level
 stcs_auto_worker_levels  | start_wlm_occupancy
```

Contoh berikut menentukan '^' sebagai string escape, kemudian menggunakan string escape untuk mencari string yang menyertakan "_“:

```
SELECT tablename, "column" FROM pg_table_def
WHERE "column" SIMILAR TO '%start^_%' ESCAPE '^'
ORDER BY tablename, "column" LIMIT 5;

        tablename         |       column
--------------------------+---------------------
 stcs_abort_idle          | idle_start_time
 stcs_abort_idle          | txn_start_time
 stcs_analyze_compression | start_time
 stcs_auto_worker_levels  | start_level
 stcs_auto_worker_levels  | start_wlm_occupancy
```

Operator POSIX

Ekspresi reguler POSIX adalah urutan karakter yang menentukan pola kecocokan. Sebuah string cocok dengan ekspresi reguler jika itu adalah anggota dari set reguler yang dijelaskan oleh ekspresi reguler.

Ekspresi reguler POSIX menyediakan sarana yang lebih kuat untuk pencocokan pola daripada SIMILAR TO operator SUKA and. Pola ekspresi reguler POSIX dapat mencocokkan bagian mana pun dari string, tidak seperti operator SIMILAR TO, yang mengembalikan true hanya jika polanya cocok dengan seluruh string.

> **Note**
>
> Pencocokan ekspresi reguler menggunakan operator POSIX mahal secara komputasi. Sebaiknya gunakan LIKE bila memungkinkan, terutama saat memproses sejumlah besar baris. Misalnya, kueri berikut secara fungsional identik, tetapi kueri yang menggunakan LIKE berjalan beberapa kali lebih cepat daripada kueri yang menggunakan ekspresi reguler:
>
> ```
> select count(*) from event where eventname ~ '.*(Ring|Die).*';
> select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die
> %';
> ```

Sintaksis

```
expression [ ! ] ~ pattern
```

Argumen

ekspresi

Eksp UTF-8 resi karakter yang valid, seperti nama kolom.

!

Operator negasi. Tidak cocok dengan ekspresi reguler.

~

Lakukan kecocokan peka huruf besar/kecil untuk setiap substring ekspresi.

> **Note**
>
> A ~~ adalah sinonim untukSUKA.

pola

Sebuah string literal yang mewakili pola ekspresi reguler.

Jika pola tidak mengandung karakter wildcard, maka pola hanya mewakili string itu sendiri.

Untuk mencari string yang menyertakan metakarakter, seperti `'. * | ? '`, dan seterusnya, hindari karakter menggunakan dua garis miring terbalik (''). `\\` Tidak seperti `SIMILAR TO` dan`LIKE`, sintaks ekspresi reguler POSIX tidak mendukung karakter escape yang ditentukan pengguna.

Salah satu ekspresi karakter dapat berupa tipe data CHAR atau VARCHAR. Jika berbeda, Amazon Redshift mengon versi pola ke tipe data ekspresi.

Semua ekspresi karakter dapat berupa tipe data CHAR atau VARCHAR. Jika ekspresi berbeda dalam tipe data, Amazon Redshift mengubahnya menjadi tipe data ekspresi.

Pencocokan pola POSIX mendukung metakarakter berikut:

POSIX	Deskripsi
.	Cocokkan dengan karakter tunggal apa pun.
*	Cocokkan nol atau lebih kejadian.
+	Cocokkan satu atau lebih kejadian.
?	Cocokkan nol atau satu kejadian.
|	Menentukan kecocokan alternatif; misalnya, `E | H` berarti `E` atau`H`.
^	Cocokkan dengan karakter awal baris.
$	Cocokkan dengan karakter akhir baris.
$	Cocokkan dengan ujung string.
[]	Kurung menentukan daftar yang cocok, yang harus cocok dengan satu ekspresi dalam daftar. Sebuah caret (^) mendahului daftar yang tidak cocok, yang cocok dengan karakter apa pun kecuali untuk ekspresi yang diwakili dalam daftar.
( )	Tanda kurung mengelompokkan item menjadi satu item logis.
{m}	Ulangi item sebelumnya persis m kali.
{m,}	Ulangi item sebelumnya m atau lebih kali.

POSIX	Deskripsi
`{m,n}`	Ulangi item sebelumnya setidaknya `m` dan tidak lebih dari `n` kali.
`[: :]`	Cocokkan karakter apa pun dalam kelas karakter POSIX. Di kelas karakter berikut, Amazon Redshift hanya mendukung karakter ASCII:`[:alnum:]` ,,, `[:alpha:]` `[:lower:]` `[:upper:]`

Amazon Redshift mendukung kelas karakter POSIX berikut.

Kelas Karakter	Deskripsi
`[[:alnum:]]`	Semua karakter alfanumerik ASCII
`[[:alpha:]]`	Semua karakter alfabet ASCII
`[[:blank:]]`	Semua karakter spasi kosong
`[[:cntrl:]]`	Semua karakter kontrol (nonprinting)
`[[:digit:]]`	Semua digit numerik
`[[:lower:]]`	Semua karakter alfabet ASCII huruf kecil
`[[:punct:]]`	Semua karakter tanda baca
`[[:space:]]`	Semua karakter spasi (nonprinting)
`[[:upper:]]`	Semua karakter alfabet ASCII huruf besar
`[[:xdigit:]]`	Semua karakter heksadesimal yang valid

Amazon Redshift mendukung Perl-influenced operator berikut dalam ekspresi reguler. Keluar dari operator menggunakan dua garis miring terbalik ('\\').

Operator	Deskripsi	Ekspresi kelas karakter setara
\\d	Karakter digit	`[[:digit:]]`

Operator	Deskripsi	Ekspresi kelas karakter setara
\\D	Karakter nondigit	[^[:digit:]]
\\w	Karakter kata	[[:word:]]
\\W	Karakter bukan kata	[^[:word:]]
\\s	Karakter spasi putih	[[:space:]]
\\S	Karakter bukan spasi putih	[^[:space:]]
\\b	Sebuah kata batas	

Contoh

Tabel berikut menunjukkan contoh pencocokan pola menggunakan operator POSIX:

Ekspresi	Pengembalian
'abc' ~ 'abc'	True
'abc' ~ 'a'	Benar
'abc' ~ 'A'	Salah
'abc' ~ '.*(b|d).*'	Benar
'abc' ~ '(b|c).*'	Benar
'AbcAbcdefgefg12efgefg12' ~ '((Ab)?c)+d((efg)+(12))+'	Benar
'aaaaaab11111xy' ~ 'a{6}.[1]{5}(x|y){2}'	Benar
'$0.87' ~ '\\$[0-9]+(\\.[0-9][0-9])?'	Benar
'ab c' ~ '[[:space:]]'	Benar

Ekspresi	Pengembalian
`'ab c' ~ '\\s'`	Benar
`' ' ~ '\\S'`	False

Contoh berikut menemukan kota yang namanya mengandung E atauH:

```
SELECT DISTINCT city FROM users
WHERE city ~ '.*E.*|.*H.*' ORDER BY city LIMIT 5;

      city
-----------------
 Agoura Hills
 Auburn Hills
 Benton Harbor
 Beverly Hills
 Chicago Heights
```

Contoh berikut menemukan kota yang namanya tidak mengandung E atauH:

```
SELECT DISTINCT city FROM users WHERE city !~ '.*E.*|.*H.*' ORDER BY city LIMIT 5;

      city
-----------------
 Aberdeen
 Abilene
 Ada
 Agat
 Agawam
```

Contoh berikut menggunakan string escape ('\\') untuk mencari string yang menyertakan titik.

```
SELECT venuename FROM venue
WHERE venuename ~ '.*\\..*'
ORDER BY venueid;

          venuename
------------------------------
 St. Pete Times Forum
 Jobing.com Arena
```

```
 Hubert H. Humphrey Metrodome
 U.S. Cellular Field
 Superpages.com Center
 E.J. Nutter Center
 Bernard B. Jacobs Theatre
 St. James Theatre
```

BETWEEN kondisi rentang

BETWEENKondisi menguji ekspresi untuk dimasukkan dalam rentang nilai, menggunakan kata kunci BETWEEN danAND.

Sintaksis

```
expression [ NOT ] BETWEEN expression AND expression
```

Ekspresi dapat berupa tipe data numerik, karakter, atau datetime, tetapi harus kompatibel. Kisarannya inklusif.

Contoh

Contoh pertama menghitung berapa banyak transaksi yang terdaftar penjualan 2, 3, atau 4 tiket:

```
select count(*) from sales
where qtysold between 2 and 4;

count
--------
104021
(1 row)
```

Kondisi rentang mencakup nilai awal dan akhir.

```
select min(dateid), max(dateid) from sales
where dateid between 1900 and 1910;

min  | max
-----+-----
1900 | 1910
```

Ekspresi pertama dalam kondisi rentang harus nilai yang lebih rendah dan ekspresi kedua nilai yang lebih besar. Contoh berikut akan selalu mengembalikan nol baris karena nilai ekspresi:

```
select count(*) from sales
where qtysold between 4 and 2;

count
-------
0
(1 row)
```

Namun, menerapkan pengubah NOT akan membalikkan logika dan menghasilkan hitungan semua baris:

```
select count(*) from sales
where qtysold not between 4 and 2;

count
--------
172456
(1 row)
```

Kueri berikut mengembalikan daftar tempat dengan 20000 hingga 50000 kursi:

```
select venueid, venuename, venueseats from venue
where venueseats between 20000 and 50000
order by venueseats desc;

venueid |       venuename               | venueseats
---------+-------------------------------+------------
116 | Busch Stadium                 |      49660
106 | Rangers BallPark in Arlington |      49115
96 | Oriole Park at Camden Yards   |      48876
...
(22 rows)
```

Contoh berikut menunjukkan penggunaan BETWEEN untuk nilai tanggal:

```
select salesid, qtysold, pricepaid, commission, saletime
from sales
where eventid between 1000 and 2000
   and saletime between '2008-01-01' and '2008-01-03'
```

```
order by saletime asc;

salesid | qtysold | pricepaid | commission |   saletime
--------+---------+-----------+------------+--------------
  65082 |       4 |       472 |       70.8 | 1/1/2008 06:06
 110917 |       1 |       337 |      50.55 | 1/1/2008 07:05
 112103 |       1 |       241 |      36.15 | 1/2/2008 03:15
 137882 |       3 |      1473 |     220.95 | 1/2/2008 05:18
  40331 |       2 |        58 |        8.7 | 1/2/2008 05:57
 110918 |       3 |      1011 |     151.65 | 1/2/2008 07:17
  96274 |       1 |       104 |       15.6 | 1/2/2008 07:18
 150499 |       3 |       135 |      20.25 | 1/2/2008 07:20
  68413 |       2 |       158 |       23.7 | 1/2/2008 08:12
```

Perhatikan bahwa meskipun rentang BETWEEN termasuk, tanggal default memiliki nilai waktu 00:00:00. Satu-satunya baris 3 Januari yang valid untuk kueri sampel adalah baris dengan waktu penjualan. `1/3/2008 00:00:00`

Kondisi nol

Kondisi null menguji nol, ketika nilai hilang atau tidak diketahui.

Sintaksis

```
expression IS [ NOT ] NULL
```

Argumen

ekspresi

Ekspresi apa pun seperti kolom.

IS NULL

Benar jika nilai ekspresi adalah null dan false ketika memiliki nilai.

IS NOT NULL

Adalah salah ketika nilai ekspresi adalah null dan true ketika memiliki nilai.

Contoh

Contoh ini menunjukkan berapa kali tabel SALES berisi null di bidang QTYSOLD:

```
select count(*) from sales
where qtysold is null;
count
-------
0
(1 row)
```

Kondisi ADA

EXIST kondisi menguji keberadaan baris dalam subquery, dan mengembalikan true jika subquery mengembalikan setidaknya satu baris. Jika NOT ditentukan, kondisi mengembalikan true jika subquery tidak mengembalikan baris.

Sintaksis

```
[ NOT ] EXISTS (table_subquery)
```

Argumen

ADA

Benar ketika table_subquery mengembalikan setidaknya satu baris.

TIDAK ADA

Benar ketika table_subquery tidak mengembalikan baris.

tabel_subquery

Sebuah subquery yang mengevaluasi ke tabel dengan satu atau lebih kolom dan satu atau lebih baris.

Contoh

Contoh ini mengembalikan semua pengidentifikasi tanggal, masing-masing satu kali, untuk setiap tanggal yang memiliki penjualan dalam bentuk apa pun:

```
select dateid from date
where exists (
select 1 from sales
where date.dateid = sales.dateid
```

```
)
order by dateid;

dateid
--------
1827
1828
1829
...
```

Dalam kondisi

Kondisi IN menguji nilai keanggotaan dalam satu set nilai atau dalam subkueri.

Sintaksis

```
expression [ NOT ] IN (expr_list | table_subquery)
```

Argumen

ekspresi

Ekspresi numerik, karakter, atau datetime yang dievaluasi terhadap expr_list atau table_subquery dan harus kompatibel dengan tipe data daftar atau subkueri tersebut.

expr_list

Satu atau lebih ekspresi dipisahkan koma, atau satu atau lebih kumpulan ekspresi dipisahkan koma yang dibatasi oleh tanda kurung.

tabel_subquery

Suatu subkueri yang mengevaluasi ke tabel dengan satu atau lebih baris, tetapi terbatas hanya pada satu kolom dalam daftar pilihannya.

DI | TIDAK MASUK

IN mengembalikan true jika ekspresi adalah anggota dari daftar ekspresi atau query. NOT IN mengembalikan true jika ekspresi bukan anggota. IN dan NOT IN mengembalikan NULL dan tidak ada baris yang dikembalikan dalam kasus berikut: Jika ekspresi menghasilkan null; atau jika tidak ada nilai expr_list atau table_subquery yang cocok dan setidaknya satu dari baris perbandingan ini menghasilkan null.

Contoh

Kondisi berikut hanya berlaku untuk nilai-nilai yang tercantum:

```
qtysold in (2, 4, 5)
date.day in ('Mon', 'Tues')
date.month not in ('Oct', 'Nov', 'Dec')
```

Optimasi untuk Daftar IN Besar

Untuk mengoptimalkan kinerja kueri, daftar IN yang mencakup lebih dari 10 nilai dievaluasi secara internal sebagai larik skalar. Daftar IN dengan nilai kurang dari 10 dievaluasi sebagai serangkaian predikat OR. Optimasi ini didukung untuk tipe data SMALLINT, INTEGER, BIGINT, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, TIMESTAMP, dan TIMESTAMPTZ.

Lihatlah output EXPLAIN untuk kueri untuk melihat efek dari pengoptimalan ini. Contoh:

```
explain select * from sales
QUERY PLAN
------------------------------------------------------------------------
XN Seq Scan on sales  (cost=0.00..6035.96 rows=86228 width=53)
Filter: (salesid = ANY ('{1,2,3,4,5,6,7,8,9,10,11}'::integer[]))
(2 rows)
```

Perintah SQL

Bahasa SQL terdiri dari perintah yang Anda gunakan untuk membuat dan memanipulasi objek database, menjalankan kueri, memuat tabel, dan memodifikasi data dalam tabel.

Amazon Redshift didasarkan pada PostgreSQL. Amazon Redshift dan PostgreSQL memiliki sejumlah perbedaan penting yang harus Anda ketahui saat merancang dan mengembangkan aplikasi gudang data Anda. Untuk informasi selengkapnya tentang perbedaan Amazon Redshift SQL dari PostgreSQL, lihat. Amazon Redshift dan PostgreSQL

> **Note**
>
> Ukuran maksimum untuk satu pernyataan SQL adalah 16 MB.

Topik

- MEMBATALKAN
- ALTER DATABASE
- ALTER DATA
- UBAH HAK ISTIMEWA DEFAULT
- MENGUBAH SKEMA EKSTERNAL
- UBAH TAMPILAN EKSTERNAL
- ALTER FUNCTION
- MENGUBAH KELOMPOK
- UBAH PENYEDIA IDENTITAS
- MENGUBAH KEBIJAKAN MASKING
- MENGUBAH TAMPILAN YANG TERWUJUD
- MENGUBAH KEBIJAKAN RLS
- MENGUBAH PERAN
- MENGUBAH PROSEDUR
- ALTER SCHEMA
- MENGUBAH SISTEM
- ALTER TABLE
- UBAH TABEL DITAMBAHKAN
- UBAH TEMPLAT
- ALTER USER
- MENGANALISA
- MENGANALISIS KOMPRESI
- LAMPIRKAN KEBIJAKAN MASKING
- LAMPIRKAN KEBIJAKAN RLS
- MULAI
- PANGGILAN
- CANCEL (BATALKAN)
- TUTUP
- KOMENTAR

- COMMIT
- MENYALIN
- BUAT BASIS DATA
- BUAT DATASHARE
- BUAT FUNGSI EKSTERNAL
- BUAT MODEL EKSTERNAL
- BUAT SKEMA EKSTERNAL
- CREATE EXTERNAL TABLE
- BUAT TAMPILAN EKSTERNAL
- CREATE FUNCTION
- BUAT GRUP
- BUAT PENYEDIA IDENTITAS
- BUAT PERPUSTAKAAN
- BUAT KEBIJAKAN MASKING
- BUAT TAMPILAN YANG TERWUJUD
- BUAT MODEL
- BUAT PROSEDUR
- BUAT KEBIJAKAN RLS
- CREATE ROLE
- BUAT SKEMA
- CREATE TABLE
- CREATE TABLE AS
- BUAT TEMPLAT
- BUAT PENGGUNA
- CREATE VIEW
- DEALOKASI
- MENYATAKAN
- DELETE
- DESKRIPSI DATA

- PENYEDIA IDENTITAS DESC
- LEPASKAN KEBIJAKAN MASKING
- LEPASKAN KEBIJAKAN RLS
- DROP DATABASE
- DROP DATA
- JATUHKAN TAMPILAN EKSTERNAL
- FUNGSI DROP
- KELOMPOK DROP
- JATUHKAN PENYEDIA IDENTITAS
- JATUHKAN PERPUSTAKAAN
- KEBIJAKAN DROP MASKING
- MODEL JATUHKAN
- JATUHKAN TAMPILAN TERWUJUD
- PROSEDUR DROP
- KEBIJAKAN DROP RLS
- DROP ROLE
- DROP SCHEMA
- MEJA DROP
- MENJATUHKAN TEMPLAT
- JATUHKAN PENGGUNA
- MENJATUHKAN TAMPILAN
- AKHIR
- EXECUTE
- EXPLAIN
- AMBIL
- HIBAH
- INSERT
- INSERT (tabel eksternal)
- KUNCI

- MERGE
- MEMPERSIAPKAN
- MENYEGARKAN TAMPILAN YANG TERWUJUD
- MENGATUR ULANG
- MENCABUT
- ROLLBACK
- SELECT
- PILIH KE
- SET
- MENGATUR OTORISASI SESI
- MENGATUR KARAKTERISTIK SESI
- MENUNJUKKAN
- TAMPILKAN PENGUNCIAN PENGGUNA
- TAMPILKAN HIBAH KOLOM
- TAMPILKAN KOLOM
- TUNJUKKAN KENDALA
- TAMPILKAN TABEL EKSTERNAL
- TAMPILKAN DATABASE
- TAMPILKAN FUNGSI
- TUNJUKKAN HIBAH
- TUNJUKKAN MODEL
- TAMPILKAN DATA SHARE
- TAMPILKAN PARAMETER
- TAMPILKAN KEBIJAKAN
- TAMPILKAN PROSEDUR
- TUNJUKKAN PROSEDUR
- TAMPILKAN SKEMA
- TAMPILKAN TABEL
- TAMPILKAN TABEL
- TAMPILKAN TEMPLATE

- TAMPILKAN TEMPLAT
- TAMPILKAN TAMPILAN
- START TRANSACTION
- MEMOTONG
- MEMBONGKAR
- UPDATE
- USE
- VAKUM

MEMBATALKAN

Menghentikan transaksi yang sedang berjalan dan membuang semua pembaruan yang dilakukan oleh transaksi tersebut. ABORT tidak berpengaruh pada transaksi yang sudah selesai.

Perintah ini melakukan fungsi yang sama dengan perintah ROLLBACK. Untuk informasi, lihat ROLLBACK.

Sintaksis

```
ABORT [ WORK | TRANSACTION ]
```

Parameter

BEKERJA

Kata kunci opsional.

TRANSAKSI

Kata kunci opsional; KERJA dan TRANSAKSI adalah sinonim.

Contoh

Contoh berikut membuat tabel kemudian memulai transaksi di mana data dimasukkan ke dalam tabel. Perintah ABORT kemudian memutar kembali penyisipan data untuk membiarkan tabel kosong.

Perintah berikut membuat tabel contoh yang disebut MOVIE_GROSS:

```
create table movie_gross( name varchar(30), gross bigint );
```

Kumpulan perintah berikutnya memulai transaksi yang menyisipkan dua baris data ke dalam tabel:

```
begin;

insert into movie_gross values ( 'Raiders of the Lost Ark', 23400000);

insert into movie_gross values ( 'Star Wars', 10000000 );
```

Selanjutnya, perintah berikut memilih data dari tabel untuk menunjukkan bahwa itu berhasil dimasukkan:

```
select * from movie_gross;
```

Output perintah menunjukkan bahwa kedua baris berhasil dimasukkan:

```
         name           |  gross
------------------------+----------
Raiders of the Lost Ark | 23400000
Star Wars               | 10000000
(2 rows)
```

Perintah ini sekarang mengembalikan perubahan data ke tempat transaksi dimulai:

```
abort;
```

Memilih data dari tabel sekarang menunjukkan tabel kosong:

```
select * from movie_gross;

 name | gross
------+-------
(0 rows)
```

ALTER DATABASE

Mengubah atribut database.

Hak istimewa yang diperlukan

Untuk menggunakan ALTER DATABASE, salah satu hak istimewa berikut diperlukan..

- Superuser
- Pengguna dengan hak istimewa ALTER DATABASE
- Pemilik database

Sintaksis

```
ALTER DATABASE database_name
{
  RENAME TO new_name
  | OWNER TO new_owner
  | [ CONNECTION LIMIT { limit | UNLIMITED } ]
    [ COLLATE { CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI } ]
    [ ISOLATION LEVEL { SNAPSHOT | SERIALIZABLE } ]
| INTEGRATION
 {
  REFRESH { { ALL | INERROR | REMEDIABLE } TABLES [ IN SCHEMA schema [, ...] ] | TABLE
 schema.table [, ...] }
   | SET
     [ QUERY_ALL_STATES [=] { TRUE | FALSE } ]
     [ ACCEPTINVCHARS [=] { TRUE | FALSE } ]
     [ REFRESH_INTERVAL <interval> ]
     [ TRUNCATECOLUMNS [=] { TRUE | FALSE } ]
     [ HISTORY_MODE [=] {TRUE | FALSE} [ FOR { {ALL} TABLES [IN SCHEMA schema [, ...] ]
 | TABLE schema.table [, ...] } ] ]
     [ AUTO_REMEDIATION [=] { TRUE | FALSE } ]
 }
}
```

Parameter

database_name

Nama database yang akan diubah. Biasanya, Anda mengubah database yang saat ini tidak terhubung; dalam hal apa pun, perubahan hanya berlaku di sesi berikutnya. Anda dapat mengubah pemilik database saat ini, tetapi Anda tidak dapat mengganti namanya:

```
alter database tickit rename to newtickit;
```

```
ERROR:  current database may not be renamed
```

GANTI NAMA MENJADI

Mengganti nama database yang ditentukan. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi. Anda tidak dapat mengganti nama dev, padb_harvest, template0, template1, atau sys:internal database, dan Anda tidak dapat mengganti nama database saat ini. Hanya pemilik database atau a yang superuser (p. 1078) dapat mengganti nama database; pemilik non-superuser juga harus memiliki hak istimewa CREATEDB.

nama_baru

Nama database baru.

PEMILIK UNTUK

Mengubah pemilik database yang ditentukan. Anda dapat mengubah pemilik database saat ini atau database lainnya. Hanya pengguna super yang dapat mengubah pemilik.

pemilik baru

Pemilik database baru. Pemilik baru harus pengguna database yang ada dengan hak menulis. Untuk informasi selengkapnya tentang hak pengguna, lihatHIBAH.

BATAS KONEKSI {limit | TIDAK TERBATAS}

Jumlah maksimum koneksi database pengguna diizinkan untuk membuka secara bersamaan. Batas tidak diberlakukan untuk pengguna super. Gunakan kata kunci UNLIMITED untuk mengizinkan jumlah maksimum koneksi bersamaan. Batas jumlah koneksi untuk setiap pengguna mungkin juga berlaku. Untuk informasi selengkapnya, lihat BUAT PENGGUNA. Defaultnya adalah UNLIMITED. Untuk melihat koneksi saat ini, kueri STV_SESSION tampilan sistem.

> **Note**
>
> Jika batas koneksi pengguna dan database berlaku, slot koneksi yang tidak digunakan harus tersedia yang berada dalam kedua batas saat pengguna mencoba menyambung.

SUSUN {CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI}

Klausa yang menentukan apakah pencarian atau perbandingan string peka huruf besar-kecil atau tidak peka huruf besar-kecil.

Anda dapat mengubah sensitivitas huruf besar dari database saat ini meskipun kosong.

Anda harus memiliki izin ALTER untuk database saat ini untuk mengubah sensitivitas huruf besar/kecil. Superuser atau pemilik database dengan izin CREATE DATABASE juga dapat mengubah sensitivitas huruf besar database.

CASE_SENSITIVE dan CS dapat dipertukarkan dan menghasilkan hasil yang sama. Demikian pula, CASE_INSENSITIVE dan CI dapat dipertukarkan dan menghasilkan hasil yang sama.

Untuk memeriksa kolasi database saat ini, gunakan DB_COLLATION fungsi.

TINGKAT ISOLASI {SNAPSHOT | DAPAT DISERIALKAN}

Klausa yang menentukan tingkat isolasi yang digunakan ketika query dijalankan terhadap database. Untuk informasi lebih lanjut tentang tingkat isolasi, lihatTingkat isolasi di Amazon Redshift.

- Isolasi SNAPSHOT — menyediakan tingkat isolasi dengan perlindungan terhadap konflik pembaruan dan penghapusan.
- Isolasi SERIALIZABLE - menyediakan serialisasi penuh untuk transaksi bersamaan.

Pertimbangkan item berikut saat mengubah tingkat isolasi database:

- Anda harus memiliki hak istimewa superuser atau CREATE DATABASE ke database saat ini untuk mengubah tingkat isolasi database.
- Anda tidak dapat mengubah tingkat isolasi `dev` database.
- Anda tidak dapat mengubah tingkat isolasi dalam blok transaksi.
- Perintah ubah tingkat isolasi gagal jika pengguna lain terhubung ke database.
- Perintah ubah level isolasi dapat mengubah pengaturan tingkat isolasi sesi saat ini.

INTEGRASI

Mengubah database integrasi Zero-ETL.

REFRESH {{ALL | INERROR | DAPAT DIPERBAIKI} TABEL [DALAM SKEMA SKEMA [,...]] | TABEL schema.table [,...]}

Klausa yang menentukan tabel mana yang diperbarui Amazon Redshift. Anda dapat menargetkan semua tabel, tabel dengan kesalahan, atau tabel yang terpengaruh oleh baris duplikat. Refresh sepenuhnya mereplikasi tabel dari database sumber.

`REMEDIABLE`menargetkan tabel dalam `Synced` keadaan yang telah dipengaruhi oleh baris duplikat. Anda dapat memeriksa tabel mana yang terpengaruh oleh query. SVV_INTEGRASI_TABLE_STATE

Untuk informasi selengkapnya, lihat Zero-ETL integrasi di Panduan Manajemen Amazon Redshift. Untuk informasi lebih lanjut tentang status integrasi, lihat SVV_INTEGRASI_TABLE_STATE danSVV_INTEGRASI.

QUERY_ALL_STATES [=] {BENAR | SALAH}

Klausa QUERY_ALL_STATES menetapkan apakah tabel integrasi Zero-ETL dapat ditanyakan di semua status (,,, dan). `Synced` `Failed` `ResyncRequired` `ResyncInitiated` Secara default, tabel integrasi Zero-ETL hanya dapat ditanyakan dalam status. `Synced`

ACCEPTINVCHARS [=] {BENAR | SALAH}

Klausa ACCEPTINVCHARS menetapkan apakah tabel integrasi Zero-ETL berlanjut dengan penyerapan ketika karakter tidak valid terdeteksi untuk tipe data VARCHAR. Ketika karakter tidak valid ditemukan, karakter yang tidak valid diganti dengan ? karakter default.

REFRESH_INTERVAL <interval>

Klausa REFRESH_INTERVAL menetapkan perkiraan interval waktu, dalam detik, yang ditunggu Amazon Redshift setelah siklus refresh selesai untuk memulai siklus berikutnya. Setiap siklus menyegarkan data dari sumber integrasi Zero-ETL ke database target, menerapkan semua perubahan yang terakumulasi sejak akhir siklus sebelumnya. Nilai 0 memulai siklus berikutnya segera setelah siklus sebelumnya selesai (replikasi hampir real-time). Nilai ruang yang lebih tinggi berputar lebih jauh, mengurangi overhead penyegaran dengan mengorbankan kesegaran data langsung. Amazon Redshift menunggu di antara siklus hanya ketika konsumsi telah menyusul; ketika perubahan menumpuk lebih cepat daripada yang dapat diterapkan, siklus berjalan tanpa menunggu.

`interval`Dapat diatur ke 0-432.000 detik (5 hari) untuk integrasi Zero-ETL yang tipe sumbernya adalah Aurora MySQL, Aurora PostgreSQL, atau mesin RDS yang didukung, dan defaultnya adalah 0. Untuk integrasi Amazon DynamoDB Zero-ETL, `interval` dapat diatur ke 900—432.000 detik (15 menit—5 hari), dan defaultnya adalah 900 detik (15 menit); 0 tidak didukung.

> **Note**
>
> Untuk integrasi penulisan intensif selain DynamoDB yang menghasilkan volume perubahan yang tinggi, atur REFRESH_INTERVAL ke nilai kecil bukan nol (misalnya, 60-120 detik). Mengelompokkan lebih banyak perubahan ke dalam setiap siklus mengurangi keseluruhan overhead komputasi replikasi.

Untuk informasi selengkapnya tentang membuat database dengan integrasi Zero-ETL, lihat Membuat database tujuan di Amazon Redshift di Panduan Manajemen Amazon Redshift.

TRUNCATECOLUMNS [=] {BENAR | SALAH}

Klausa TRUNCATECOLUMNS menetapkan apakah tabel integrasi Zero-ETL berlanjut dengan penyerapan ketika nilai untuk kolom VARCHAR atau atribut kolom SUPER berada di luar batas. KetikaTRUE, nilai-nilai dipotong agar sesuai dengan kolom dan nilai atribut JSON yang meluap dipotong agar sesuai dengan kolom SUPER.

HISTORY_MODE [=] {TRUE | FALSE} [UNTUK {{SEMUA} TABEL [DALAM SKEMA SKEMA [,...]] | TABEL schema.table [,...]}]

Klausa yang menentukan apakah Amazon Redshift akan menyetel mode riwayat untuk semua tabel atau tabel dalam skema tertentu yang berpartisipasi dalam integrasi Zero-ETL. Opsi ini hanya berlaku untuk database yang dibuat untuk integrasi Zero-ETL.

Klausa HISTORY_MODE dapat diatur ke TRUE atau. FALSE Nilai default-nya FALSE. Mengaktifkan dan menonaktifkan mode riwayat hanya berlaku untuk tabel yang berada dalam Synced status. Untuk informasi tentang HISTORY_MODE, lihat Mode Riwayat di Panduan Manajemen Amazon Redshift.

AUTO_REMEDIASI [=] {BENAR | SALAH}

Menentukan apakah Amazon Redshift secara otomatis menyinkronkan ulang tabel yang terpengaruh oleh baris duplikat. Saat disetel keTRUE, Amazon Redshift menandai tabel yang terpengaruh untuk disinkronisasi ulang tanpa memerlukan intervensi manual. Nilai default-nya FALSE.

Anda dapat memantau tabel mana yang telah ditandai dengan menanyakan. SVV_INTEGRASI_TABLE_STATE Pengaturan saat ini terlihat di `auto_remediation` kolomSVV_INTEGRASI.

Catatan penggunaan

Perintah ALTER DATABASE berlaku untuk sesi berikutnya bukan sesi saat ini. Anda harus menyambung kembali ke database yang diubah untuk melihat efek dari perubahan.

Contoh

Contoh berikut mengganti nama database bernama TICKIT_SANDBOX menjadi TICKIT_TEST:

```
alter database tickit_sandbox rename to tickit_test;
```

Contoh berikut mengubah pemilik database TICKIT (database saat ini) menjadi DWUSER:

```
alter database tickit owner to dwuser;
```

Contoh berikut mengubah sensitivitas huruf besar database dari database sampledb:

```
ALTER DATABASE sampledb COLLATE CASE_INSENSITIVE;
```

Contoh berikut mengubah database bernama **sampledb** dengan tingkat isolasi SNAPSHOT.

```
ALTER DATABASE sampledb ISOLATION LEVEL SNAPSHOT;
```

Contoh berikut menyegarkan tabel **schema1.sample_table1** dan **schema2.sample_table2** dalam database **sample_integration_db** dalam integrasi Zero-ETL Anda.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH TABLE schema1.sample_table1,
 schema2.sample_table2;
```

Contoh berikut menyegarkan semua tabel yang disinkronkan dan gagal dalam integrasi Zero-ETL Anda.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH ALL tables;
```

Contoh berikut menetapkan interval refresh untuk integrasi Zero-ETL menjadi 600 detik..

```
ALTER DATABASE sample_integration_db INTEGRATION SET REFRESH_INTERVAL 600;
```

Contoh berikut menyegarkan semua tabel yang ada di `ErrorState` dalam skema**sample_schema**.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH INERROR TABLES in SCHEMA
 sample_schema;
```

Contoh berikut mengaktifkan mode riwayat untuk tabel`myschema.table1`.

```
ALTER DATABASE sample_integration_db INTEGRATION SET HISTORY_MODE = true FOR TABLE
 myschema.table1
```

Contoh berikut mengaktifkan mode riwayat untuk semua tabel dimyschema.

```
ALTER DATABASE sample_integration_db INTEGRATION SET HISTORY_MODE = true for ALL TABLES
 IN SCHEMA myschema
```

Contoh berikut memungkinkan remediasi otomatis baris duplikat untuk database integrasi Zero-ETL.

```
ALTER DATABASE sample_integration_db INTEGRATION SET AUTO_REMEDIATION = true;
```

Contoh berikut menyegarkan semua tabel yang dipengaruhi oleh baris duplikat dalam database integrasi Zero-ETL.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH REMEDIABLE TABLES;
```

Contoh berikut menyegarkan tabel yang dipengaruhi oleh baris duplikat dalam skemamyschema.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH REMEDIABLE TABLES IN SCHEMA
 myschema;
```

ALTER DATA

Mengubah definisi datashare. Anda dapat menambahkan objek atau menghapus objek menggunakan ALTER DATASHARE. Anda hanya dapat mengubah datashare di database saat ini. Tambahkan atau hapus objek dari database terkait ke datashare. Pemilik datashare dengan izin yang diperlukan pada objek datashare yang akan ditambahkan atau dihapus dapat mengubah datashare.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk ALTER DATASHARE:

- Pengguna super.
- Pengguna dengan hak istimewa ALTER DATASHARE.
- Pengguna yang memiliki hak istimewa ALTER atau ALL di datashare.
- Untuk menambahkan objek tertentu ke datashare, pengguna harus memiliki hak istimewa pada objek. Untuk kasus ini, pengguna harus menjadi pemilik objek atau memiliki hak istimewa SELECT, USAGE, atau ALL pada objek.

Sintaksis

Sintaks berikut menggambarkan cara menambah atau menghapus objek ke datashare.

```
ALTER DATASHARE datashare_name { ADD | REMOVE } {
TABLE schema.table [, ...]
| SCHEMA schema [, ...]
| FUNCTION schema.sql_udf (argtype,...) [, ...]
| ALL TABLES IN SCHEMA schema [, ...]
| ALL FUNCTIONS IN SCHEMA schema [, ...] }
```

Sintaks berikut menggambarkan cara mengkonfigurasi properti datashare.

```
ALTER DATASHARE datashare_name {
[ SET PUBLICACCESSIBLE [=] TRUE | FALSE ]
[ SET INCLUDENEW [=] TRUE | FALSE FOR SCHEMA schema ] }
```

Parameter

datahare_name

Nama datashare yang akan diubah.

TAMBAHKAN | HAPUS

Klausa yang menentukan apakah akan menambahkan objek ke atau menghapus objek dari datashare.

Skema TABEL. tabel [,...]

Nama tabel atau tampilan dalam skema yang ditentukan untuk ditambahkan ke datashare.

Skem a SKEMA [,...]

Nama skema yang akan ditambahkan ke datashare.

Skema FUNGSI. sql_udf (tipe arg,...) [,...]

Nama fungsi SQL yang ditentukan pengguna dengan tipe argumen untuk ditambahkan ke datashare.

SEMUA TABEL DALAM skema SK EMA [,...]

Klausa yang menentukan apakah akan menambahkan semua tabel dan tampilan dalam skema yang ditentukan ke datashare.

SEMUA FUNGSI DALAM SK EMA SKEMA [,...]}

Klausa yang menentukan menambahkan semua fungsi dalam skema yang ditentukan ke datashare.

[SETEL PUBLICAVAILABLE [=] BENAR | SALAH]

Klausa yang menentukan apakah datashare dapat dibagikan ke cluster yang dapat diakses publik.

[SET INCLUDENEW [=] BENAR | SALAH UNTUK SKEMA SKEMA]

Klausa yang menentukan apakah akan menambahkan tabel masa depan, tampilan, atau fungsi yang ditentukan pengguna SQL (UDF) yang dibuat dalam skema yang ditentukan ke datashare. Tabel saat ini, tampilan, atau UDF SQL dalam skema yang ditentukan tidak ditambahkan ke datashare. Hanya pengguna super yang dapat mengubah properti ini untuk setiap pasangan datashare-schema. Secara default, klausa INCLUDENEW adalah false.

Catatan penggunaan ALTER DATASHARE

- Pengguna berikut dapat mengubah datashare:
  - Seorang pengguna super
  - Pemilik datashare
  - Pengguna yang memiliki hak istimewa ALTER atau ALL di datashare
- Untuk menambahkan objek tertentu ke datashare, pengguna harus memiliki hak istimewa yang benar pada objek. Pengguna harus menjadi pemilik objek atau memiliki hak istimewa SELECT, USAGE, atau ALL pada objek.
- Anda dapat berbagi skema, tabel, tampilan reguler, tampilan pengikatan terlambat, tampilan terwujud, dan fungsi yang ditentukan pengguna SQL (UDF). Tambahkan skema ke datashare terlebih dahulu sebelum menambahkan objek dalam skema.

  Saat Anda menambahkan skema, Amazon Redshift tidak menambahkan semua objek di bawahnya. Anda harus menambahkannya secara eksplisit.
- Sebaiknya buat berbagi AWS Data Exchange data dengan pengaturan yang dapat diakses publik diaktifkan.
- Secara umum, kami menyarankan agar Anda tidak mengubah AWS Data Exchange datashare untuk mematikan aksesibilitas publik menggunakan pernyataan ALTER DATASHARE. Jika Anda melakukannya, mereka Akun AWS yang memiliki akses ke datashare kehilangan akses jika cluster

mereka dapat diakses publik. Melakukan perubahan jenis ini dapat melanggar ketentuan produk data AWS Data Exchange. Untuk pengecualian untuk rekomendasi ini, lihat berikut.

Contoh berikut menunjukkan kesalahan saat AWS Data Exchange datashare dibuat dengan pengaturan dimatikan.

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
ERROR:  Alter of ADX-managed datashare salesshare requires session variable
 datashare_break_glass_session_var to be set to value 'c670ba4db22f4b'
```

Untuk mengizinkan perubahan AWS Data Exchange datashare untuk mematikan pengaturan yang dapat diakses publik, atur variabel berikut dan jalankan pernyataan ALTER DATASHARE lagi.

```
SET datashare_break_glass_session_var to 'c670ba4db22f4b';
```

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
```

Dalam hal ini, Amazon Redshift menghasilkan nilai satu kali acak untuk menyetel variabel sesi agar memungkinkan ALTER DATASHARE SET PUBLICACCESSIBLE FALSE untuk datashare. AWS Data Exchange

Contoh

Contoh berikut menambahkan `public` skema ke datashare`salesshare`.

```
ALTER DATASHARE salesshare ADD SCHEMA public;
```

Contoh berikut menambahkan `public.tickit_sales_redshift` tabel ke datashare`salesshare`.

```
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
```

Contoh berikut menambahkan semua tabel ke datashare`salesshare`.

```
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

Contoh berikut menghapus `public.tickit_sales_redshift` tabel dari datashare`salesshare`.

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

UBAH HAK ISTIMEWA DEFAULT

Mendefinisikan kumpulan izin akses default yang akan diterapkan ke objek yang dibuat di masa mendatang oleh pengguna yang ditentukan. Secara default, pengguna hanya dapat mengubah izin akses default mereka sendiri. Hanya pengguna super yang dapat menentukan izin default untuk pengguna lain.

Anda dapat menerapkan hak istimewa default ke peran, pengguna, atau grup pengguna. Anda dapat mengatur izin default secara global untuk semua objek yang dibuat dalam database saat ini, atau untuk objek yang dibuat hanya dalam skema yang ditentukan.

Izin default hanya berlaku untuk objek baru. Menjalankan ALTER DEFAULT PRIVILEGES tidak mengubah izin pada objek yang ada. Untuk memberikan izin pada semua objek saat ini dan yang akan datang yang dibuat oleh pengguna mana pun dalam database atau skema, lihat Izin tercakupan.

Untuk melihat informasi tentang hak istimewa default untuk pengguna database, kueri tabel katalog PG_DEFAULT_ACL sistem.

Untuk informasi selengkapnya tentang hak istimewa, lihatHIBAH.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk ALTER DEFAULT PRIVILEGES:

- Superuser
- Pengguna dengan hak istimewa ALTER DEFAULT PRIVILEGES
- Pengguna mengubah hak akses default mereka sendiri
- Pengguna menetapkan hak istimewa untuk skema yang mereka miliki hak akses

Sintaksis

```
ALTER DEFAULT PRIVILEGES
    [ FOR USER target_user [, ...] ]
    [ IN SCHEMA schema_name [, ...] ]
    grant_or_revoke_clause
```

```
where grant_or_revoke_clause is one of:

GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | TRUNCATE } [,...] |
 ALL [ PRIVILEGES ] }
 ON TABLES
 TO { user_name [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
 ON FUNCTIONS
 TO { user_name [ WITH GRANT OPTION ] |  ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
 ON PROCEDURES
 TO { user_name [ WITH GRANT OPTION ] |  ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

REVOKE [ GRANT OPTION FOR ] { { SELECT | INSERT | UPDATE | DELETE | REFERENCES |
 TRUNCATE } [,...] | ALL [ PRIVILEGES ] }
 ON TABLES
 FROM user_name [, ...] [ RESTRICT ]

REVOKE  { { SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRUNCATE } [,...] | ALL
 [ PRIVILEGES ] }
 ON TABLES
 FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ] { EXECUTE | ALL [ PRIVILEGES ] }
 ON FUNCTIONS
 FROM user_name [, ...] [ RESTRICT ]

REVOKE { EXECUTE | ALL [ PRIVILEGES ] }
 ON FUNCTIONS
 FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ] { EXECUTE | ALL [ PRIVILEGES ] }
 ON PROCEDURES
 FROM user_name [, ...] [ RESTRICT ]

REVOKE { EXECUTE | ALL [ PRIVILEGES ] }
 ON PROCEDURES
 FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]
```

Parameter

UNTUK PENGGUNA target_user

Tidak wajib. Nama pengguna yang hak istimewa defaultnya ditentukan. Hanya pengguna super yang dapat menentukan hak istimewa default untuk pengguna lain. Nilai default adalah pengguna saat ini.

DALAM SKEMA schema_nama

Tidak wajib. Jika klausa IN SCHEMA muncul, hak istimewa default yang ditentukan diterapkan ke objek baru yang dibuat dalam schema_ name yang ditentukan. Dalam hal ini, pengguna atau grup pengguna yang menjadi target ALTER DEFAULT PRIVILEGES harus memiliki hak istimewa CREATE untuk skema yang ditentukan. Hak istimewa default yang khusus untuk skema ditambahkan ke hak istimewa default global yang ada. Secara default, hak istimewa default diterapkan secara global ke seluruh database.

HIBAH

Kumpulan hak istimewa untuk diberikan kepada pengguna atau grup tertentu untuk semua tabel baru dan tampilan, fungsi, atau prosedur tersimpan yang dibuat oleh pengguna yang ditentukan. Anda dapat mengatur hak istimewa dan opsi yang sama dengan klausa GRANT yang Anda dapat dengan HIBAH perintah.

DENGAN OPSI HIBAH

Klausa yang menunjukkan bahwa pengguna yang menerima hak istimewa pada gilirannya dapat memberikan hak istimewa yang sama kepada orang lain. Anda tidak dapat memberikan WITH GRANT OPTION ke grup atau PUBLIK.

KE nama pengguna | Nama peran | nama grup GRUP

Nama pengguna, peran, atau grup pengguna yang menerapkan hak istimewa default yang ditentukan.

MENCABUT

Kumpulan hak istimewa untuk dicabut dari pengguna atau grup tertentu untuk semua tabel baru, fungsi, atau prosedur tersimpan yang dibuat oleh pengguna yang ditentukan. Anda dapat mengatur hak istimewa dan opsi yang sama dengan klausa REVOKE yang dapat Anda lakukan dengan MENCABUT perintah.

OPSI HIBAH UNTUK

Klausa yang hanya mencabut opsi untuk memberikan hak istimewa tertentu kepada pengguna lain dan tidak mencabut hak istimewa itu sendiri. Anda tidak dapat mencabut OPSI GRANT dari grup atau dari PUBLIK.

DARI nama pengguna | Nama peran | nama grup GRUP

Nama pengguna, peran, atau grup pengguna dari mana hak istimewa yang ditentukan dicabut secara default.

MEMBATASI

Opsi RESTRICT hanya mencabut hak istimewa yang diberikan pengguna secara langsung. Ini adalah opsi default.

Contoh

Misalkan Anda ingin mengizinkan setiap pengguna dalam grup pengguna `report_readers` untuk melihat semua tabel dan tampilan yang dibuat oleh penggunareport_admin. Dalam hal ini, jalankan perintah berikut sebagai superuser.

```
alter default privileges for user report_admin grant select on tables to group
 report_readers;
```

Dalam contoh berikut, perintah pertama memberikan hak istimewa SELECT pada semua tabel baru yang Anda buat. Setiap kali Anda membuat tampilan baru, Anda harus secara eksplisit memberikan hak istimewa ke tampilan atau menjalankan kembali `alter default privileges` perintah lagi.

```
alter default privileges grant select on tables to public;
```

Contoh berikut memberikan hak istimewa INSERT ke grup `sales_admin` pengguna untuk semua tabel dan tampilan baru yang Anda buat dalam `sales` skema.

```
alter default privileges in schema sales grant insert on tables to group sales_admin;
```

Contoh berikut membalikkan perintah ALTER DEFAULT PRIVILEGES dalam contoh sebelumnya.

```
alter default privileges in schema sales revoke insert on tables from group
 sales_admin;
```

Secara default, grup pengguna PUBLIK memiliki izin eksekusi untuk semua fungsi baru yang ditentukan pengguna. Untuk mencabut izin `public` eksekusi untuk fungsi baru Anda dan kemudian memberikan izin eksekusi hanya untuk grup `dev_test` pengguna, jalankan perintah berikut.

```
alter default privileges revoke execute on functions from public;
alter default privileges grant execute on functions to group dev_test;
```

MENGUBAH SKEMA EKSTERNAL

Mengubah skema eksternal yang ada di database saat ini. Hanya pemilik skema, pengguna super, atau pengguna dengan hak ALTER pada skema yang dapat mengubahnya. Hanya skema eksternal yang dibuat dari DATA CATALOG, KAFKA, atau MSK yang dapat diubah.

Pemilik skema ini adalah penerbit perintah CREATE EXTERNAL SCHEMA. Untuk mentransfer kepemilikan skema eksternal, gunakan ALTER SCHEMA untuk mengubah pemilik. Untuk memberikan akses ke skema ke pengguna lain atau grup pengguna, gunakan perintah GRANT.

Anda tidak dapat menggunakan perintah GRANT atau REVOKE untuk izin pada tabel eksternal. Sebagai gantinya, berikan atau cabut izin pada skema eksternal.

Untuk informasi selengkapnya, lihat berikut ini:

- ALTER SCHEMA
- HIBAH
- MENCABUT
- BUAT SKEMA EKSTERNAL
- Mengaktifkan otentikasi MTL untuk skema eksternal yang ada

Untuk melihat detail skema eksternal, kueri tampilan sistem SVV_EXTERNAL_SCHEMAS. Untuk informasi selengkapnya, lihat SVV_EXTERNAL_SCHEMAS.

Sintaks

```
ALTER EXTERNAL SCHEMA schema_name
[ IAM_ROLE [ default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' ] ]
[ AUTHENTICATION [ none | iam | mtls] ]
[ AUTHENTICATION_ARN 'acm-certificate-arn' | SECRET_ARN 'asm-secret-arn' ]
```

```
[ URI 'Kafka bootstrap URL' ]
```

Jika Anda memiliki skema eksternal yang ada yang Anda gunakan untuk penyerapan streaming dan Anda ingin menerapkan TLS bersama untuk otentikasi, Anda dapat menjalankan perintah seperti berikut, yang menentukan otentikasi MTL dan sertifikat ACM ARN di ACM.

```
ALTER EXTERNAL SCHEMA schema_name
AUTHENTICATION mtls
AUTHENTICATION_ARN 'arn:aws:acm:us-east-1:444455556666:certificate/certificate_ID';
```

Atau Anda dapat memodifikasi otentikasi MTL, dengan mengacu pada ARN rahasia di Secrets Manager.

```
ALTER EXTERNAL SCHEMA schema_name
AUTHENTICATION mtls
SECRET_ARN 'arn:aws:secretsmanager:us-east-1:012345678910:secret:myMTLSSecret';
```

Contoh berikut menunjukkan cara memodifikasi URI untuk ALTER EXTERNAL SCHEMA:

```
ALTER EXTERNAL SCHEMA schema_name
URI 'lkc-ghidef-67890.centralus.azure.glb.confluent.cloud:9092';
```

Contoh berikut menunjukkan cara memodifikasi peran IAM untuk ALTER EXTERNAL SCHEMA:

```
ALTER EXTERNAL SCHEMA schema_name
IAM_ROLE 'arn:aws:iam::012345678901:role/testrole';
```

Parameter

<role-name>IAM_ROLE [default | 'SESI' | 'arn:aws:iam: :< akun-id>:role/ ']AWS

Gunakan `default` kata kunci agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default.

Gunakan `'SESSION'` jika Anda terhubung ke cluster Amazon Redshift menggunakan identitas federasi dan mengakses tabel dari skema eksternal yang dibuat menggunakan perintah ini.

Lihat MEMBUAT SKEMA EKSTERNAL untuk informasi selengkapnya.

OTENTIKASI

Jenis otentikasi yang ditentukan untuk penyerapan streaming. Penyerapan streaming dengan jenis otentikasi berfungsi dengan Apache Kafka, Confluent Cloud, dan Amazon Managed Streaming untuk Apache Kafka. Lihat MEMBUAT SKEMA EKSTERNAL untuk informasi selengkapnya.

OTENTIKASI_ARN

ARN AWS Certificate Manager sertifikat yang digunakan oleh Amazon Redshift untuk otentikasi mtls dengan Apache Kafka, Confluent Cloud, atau Amazon Managed Streaming untuk Apache Kafka (Amazon MSK). ARN tersedia di konsol ACM ketika Anda memilih sertifikat yang dikeluarkan.

SECRET_ARN

Nama Sumber Daya Amazon (ARN) dari rahasia yang didukung dibuat menggunakan AWS Secrets Manager. Untuk informasi tentang cara membuat dan mengambil ARN untuk rahasia, lihat Mengel ola rahasia dengan AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna, dan Mengambil Nama Sumber Daya Amazon (ARN) dari rahasia di Amazon Red shift.

URI

URL bootstrap dari cluster Apache Kafka, Confluent Cloud atau Amazon Managed Streaming untuk Apache Kafka (Amazon MSK). Titik akhir harus dapat dijangkau (dapat dirutekan) dari cluster Amazon Redshift. Lihat MEMBUAT SKEMA EKSTERNAL untuk informasi selengkapnya.

UBAH TAMPILAN EKSTERNAL

Gunakan perintah ALTER EXTERNAL VIEW untuk memperbarui tampilan eksternal Anda. Bergantung pada parameter yang Anda gunakan, mesin SQL lain seperti Amazon Athena dan Amazon EMR Spark yang juga dapat mereferensikan tampilan ini mungkin terpengaruh. Untuk informasi selengkapnya tentang tampilan Katalog Data, lihat AWS Glue Data Catalog tampilan.

Sintaksis

```
ALTER EXTERNAL VIEW schema_name.view_name
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
 external_schema_name.view_name}
[FORCE] { AS (query_definition) | REMOVE DEFINITION }
```

Parameter

nama_skema.nama_tampilan

Skema yang dilampirkan ke AWS Glue database Anda, diikuti dengan nama tampilan.

nama_katalog.schema_nama.view_name | awsdatacatalog.dbname.view_name | nama_schema_eksternal.nama_nama.view_name

Notasi skema yang akan digunakan saat mengubah tampilan. Anda dapat menentukan untuk menggunakan AWS Glue Data Catalog, database Glue yang Anda buat, atau skema eksternal yang Anda buat. Lihat MEMBUAT DATABASE dan MEMBUAT SKEMA EKSTERNAL untuk informasi selengkapnya.

KEKUATAN

Apakah AWS Lake Formation harus memperbarui definisi tampilan bahkan jika objek yang direferensikan dalam tabel tidak konsisten dengan mesin SQL lainnya. Jika Lake Formation memperbarui tampilan, tampilan dianggap basi untuk mesin SQL lainnya sampai mesin tersebut diperbarui juga.

Definisi kueri AS

Definisi kueri SQL yang dijalankan Amazon Redshift untuk mengubah tampilan.

HAPUS DEFINISI

Apakah akan menjatuhkan dan membuat ulang tampilan. Tampilan harus dijatuhkan dan dibuat ulang untuk menandainya sebagai`PROTECTED`.

Contoh

Contoh berikut mengubah tampilan Katalog Data bernama sample_schema.glue_data_catalog_view.

```
ALTER EXTERNAL VIEW sample_schema.glue_data_catalog_view
FORCE
REMOVE DEFINITION
```

ALTER FUNCTION

Mengganti nama fungsi atau mengubah pemilik. Baik nama fungsi dan tipe data diperlukan. Hanya pemilik atau superuser yang dapat mengganti nama fungsi. Hanya pengguna super yang dapat mengubah pemilik fungsi.

Sintaksis

```
ALTER FUNCTION function_name ( { [ py_arg_name py_arg_data_type | sql_arg_data_type }
 [ , ... ] ] )
    RENAME TO new_name
```

```
ALTER FUNCTION function_name ( { [ py_arg_name py_arg_data_type | sql_arg_data_type }
 [ , ... ] ] )
    OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
```

Parameter

function_name

Nama fungsi yang akan diubah. Entah menentukan nama fungsi di jalur pencarian saat ini, atau gunakan format `schema_name.function_name` untuk menggunakan skema tertentu.

py_arg_nama_py_arg_data_type | sql_arg_data_type

Tidak wajib. Daftar nama argumen input dan tipe data untuk fungsi yang ditentukan pengguna Python, atau daftar tipe data argumen input untuk fungsi yang ditentukan pengguna SQL.

nama_baru

Nama baru untuk fungsi yang ditentukan pengguna.

pemilik baru | PENGGUNA_SAAT INI | SESSION_USER

Pemilik baru untuk fungsi yang ditentukan pengguna.

Contoh

Contoh berikut mengubah nama fungsi dari `first_quarter_revenue` menjadi`quarterly_revenue`.

```
ALTER FUNCTION first_quarter_revenue(bigint, numeric, int)
        RENAME TO quarterly_revenue;
```

Contoh berikut mengubah pemilik `quarterly_revenue` fungsi menjadi`etl_user`.

```
ALTER FUNCTION quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

MENGUBAH KELOMPOK

Mengubah grup pengguna. Gunakan perintah ini untuk menambahkan pengguna ke grup, menghapus pengguna dari grup, atau mengganti nama grup.

Sintaksis

```
ALTER GROUP group_name
{
ADD USER username [, ... ] |
DROP USER username [, ... ] |
RENAME TO new_name
}
```

Parameter

group_name

Nama grup pengguna yang akan dimodifikasi.

MENAMBAHKAN

Menambahkan pengguna ke grup pengguna.

MENJATUHKAN

Menghapus pengguna dari grup pengguna.

username

Nama pengguna yang akan ditambahkan ke grup atau dihapus dari grup.

GANTI NAMA MENJADI

Mengganti nama grup pengguna. Nama grup yang dimulai dengan dua garis bawah disediakan untuk penggunaan internal Amazon Redshift. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

nama_baru

Nama baru grup pengguna.

Contoh

Contoh berikut menambahkan pengguna bernama DWUSER ke grup ADMIN_GROUP.

```
ALTER GROUP admin_group
ADD USER dwuser;
```

Contoh berikut mengganti nama grup ADMIN_GROUP menjadi ADMINISTRATORS.

```
ALTER GROUP admin_group
RENAME TO administrators;
```

Contoh berikut menambahkan dua pengguna ke grup ADMIN_GROUP.

```
ALTER GROUP admin_group
ADD USER u1, u2;
```

Contoh berikut menjatuhkan dua pengguna dari grup ADMIN_GROUP.

```
ALTER GROUP admin_group
DROP USER u1, u2;
```

UBAH PENYEDIA IDENTITAS

Mengubah penyedia identitas untuk menetapkan parameter dan nilai baru. Ketika Anda menjalankan perintah ini, semua nilai parameter yang ditetapkan sebelumnya dihapus sebelum nilai baru ditetapkan. Hanya pengguna super yang dapat mengubah penyedia identitas.

Sintaksis

```
ALTER IDENTITY PROVIDER identity_provider_name
[PARAMETERS parameter_string]
[NAMESPACE namespace]
[IAM_ROLE iam_role]
[AUTO_CREATE_ROLES
    [ TRUE [ { INCLUDE | EXCLUDE } GROUPS LIKE filter_pattern] |
      FALSE
    ]
[DISABLE | ENABLE]
```

Parameter

nama penyedia identitas

Nama penyedia identitas baru. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

parameter_string

String yang berisi objek JSON yang diformat dengan benar yang berisi parameter dan nilai yang diperlukan untuk penyedia identitas tertentu.

ruang nama

Namespace organisasi.

iam_role

Peran IAM yang menyediakan izin untuk koneksi ke Pusat Identitas IAM. Parameter ini hanya berlaku jika tipe penyedia identitas adalah AWSIDC.

auto_create_role

Mengaktifkan atau menonaktifkan fitur peran buat otomatis. Jika nilainya TRUE, Amazon Redshift mengaktifkan fitur pembuatan peran otomatis. Jika nilainya SALAH, Amazon Redshift menonaktifkan fitur peran buat otomatis. Jika nilai untuk parameter ini tidak ditentukan, Amazon Redshift menentukan nilai menggunakan logika berikut:

- Jika `AUTO_CREATE_ROLES` disediakan tetapi nilainya tidak ditentukan, nilainya disetel ke TRUE.
- Jika `AUTO_CREATE_ROLES` tidak disediakan dan penyedia identitas adalah AWSIDC, nilainya disetel ke FALSE.
- Jika `AUTO_CREATE_ROLES` tidak disediakan dan penyedia identitasnya adalah Azure, nilainya disetel ke TRUE.

Untuk memasukkan grup, tentukan`INCLUDE`. Defaultnya kosong, yang berarti sertakan semua grup `AUTO_CREATE_ROLES` jika aktif.

Untuk mengecualikan grup, tentukan`EXCLUDE`. Defaultnya kosong, yang berarti jangan mengecualikan grup apa pun jika `AUTO_CREATE_ROLES` aktif.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama grup. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan setiap urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.

Jika filter_pattern tidak mengandung metacharacters, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator sama.

filter_pattern mendukung karakter berikut:

- Karakter alfabet huruf besar dan huruf kecil (dan a-z) A-Z
- Angka (0-9)
- Karakter khusus berikut:

```
_ % ^ * + ? { } , $
```

NONAKTIFKAN atau AKTIFKAN

Mengaktifkan atau menonaktifkan penyedia identitas. Defaultnya adalah ENABLE.

Contoh

Contoh berikut mengubah penyedia identitas bernama o auth_standard. Ini berlaku khusus ketika Microsoft Azure AD adalah penyedia identitas.

```
ALTER IDENTITY PROVIDER oauth_standard
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
"client_id":"87f4aa26-78b7-410e-bf29-57b39929ef9a",
"client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]
}'
```

Contoh berikut menunjukkan cara mengatur namespace penyedia identitas. Ini dapat berlaku untuk Microsoft Azure AD, jika mengikuti pernyataan seperti sampel sebelumnya, atau ke penyedia identitas lain. Ini juga dapat berlaku untuk kasus di mana Anda menghubungkan cluster yang disediakan Amazon Redshift atau grup kerja tanpa server Amazon Redshift yang ada ke Pusat Identitas IAM, jika Anda memiliki koneksi yang diatur melalui aplikasi terkelola.

```
ALTER IDENTITY PROVIDER "my-redshift-idc-application"
NAMESPACE 'MYCO';
```

Contoh berikut menetapkan peran IAM dan berfungsi dalam kasus penggunaan untuk mengonfigurasi integrasi Redshift dengan Pusat Identitas IAM.

```
ALTER IDENTITY PROVIDER "my-redshift-idc-application"
IAM_ROLE 'arn:aws:iam::123456789012:role/myadministratorrole';
```

Untuk informasi selengkapnya tentang menyiapkan koneksi ke Pusat Identitas IAM dari Redshift, lihat Menghubun gkan Redshift dengan Pusat Identitas IAM untuk memberi pengguna pengalaman masuk tunggal.

Menonaktifkan penyedia identitas

Contoh pernyataan berikut menunjukkan cara menonaktifkan penyedia identitas. Ketika dinonaktifkan, pengguna federasi dari penyedia identitas tidak dapat masuk ke cluster sampai diaktifkan lagi.

```
ALTER IDENTITY PROVIDER "redshift-idc-app" DISABLE;
```

MENGUBAH KEBIJAKAN MASKING

Mengubah kebijakan penyembunyian data dinamis yang ada. Untuk informasi selengkapnya tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat mengubah kebijakan masking.

Sintaksis

```
ALTER MASKING POLICY
{ policy_name | database_name.policy_name }
```

```
USING (masking_expression);
```

Parameter

nama_kebijakan

Nama kebijakan masking. Ini harus menjadi nama kebijakan masking yang sudah ada di database.

database_name

Nama database tempat kebijakan dibuat. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

ekspresi penyamaran

Ekspresi SQL digunakan untuk mengubah kolom target. Ini dapat ditulis menggunakan fungsi manipulasi data seperti fungsi manipulasi String, atau dalam hubungannya dengan fungsi yang ditentukan pengguna yang ditulis dalam SQL, Python, atau dengan AWS Lambda.

Ekspresi harus sesuai dengan kolom input ekspresi asli dan tipe data. Misalnya, jika kolom input kebijakan masking asli adalah `sample_1 FLOAT` dan`sample_2 VARCHAR(10)`, Anda tidak akan dapat mengubah kebijakan masking untuk mengambil kolom ketiga, atau membuat kebijakan mengambil FLOAT dan BOOLEAN. Jika Anda menggunakan konstanta sebagai ekspresi masking, Anda harus secara eksplisit menempatkannya ke tipe yang cocok dengan tipe input.

Anda harus memiliki izin PENGGUNAAN pada fungsi yang ditentukan pengguna yang Anda gunakan dalam ekspresi masking.

Untuk penggunaan ALTER MASKING POLICY di Katalog Izin Federasi Amazon Redshift, lihat M [engelola kontrol akses dengan izin federasi Amazon Redshift.]

MENGUBAH TAMPILAN YANG TERWUJUD

Mengubah atribut tampilan yang terwujud.

Sintaksis

```
ALTER MATERIALIZED VIEW mv_name
```

```
{
AUTO REFRESH { YES | NO }
| ALTER DISTKEY column_name
| ALTER DISTSTYLE ALL
| ALTER DISTSTYLE EVEN
| ALTER DISTSTYLE KEY DISTKEY column_name
| ALTER DISTSTYLE AUTO
| ALTER [COMPOUND] SORTKEY ( column_name [,...] )
| ALTER SORTKEY AUTO
| ALTER SORTKEY NONE
| ROW LEVEL SECURITY { ON | OFF } [ CONJUNCTION TYPE { AND | OR } ] [FOR DATASHARES]
};
```

Parameter

mv_nama_

Nama pandangan yang terwujud untuk diubah.

PENYEGARAN OTOMATIS {YA | TIDAK}

Klausa yang mengaktifkan atau menonaktifkan penyegaran otomatis tampilan yang terwujud. Untuk informasi selengkapnya tentang penyegaran otomatis tampilan yang terwujud, lihatMenyegarkan tampilan yang terwujud.

UBAH DISTSTYLE SEMUA

Klausa yang mengubah gaya distribusi yang ada dari suatu relasi denganALL. Pertimbangkan hal berikut:

- ALTER DISTSTYLE, ALTER SORTKEY, dan VACUUM tidak dapat berjalan secara bersamaan pada relasi yang sama.
  - Jika VACUUM saat ini sedang berjalan, maka menjalankan ALTER DISTSTYLE ALL mengembalikan kesalahan.
  - Jika ALTER DISTSTYLE ALL berjalan, maka vakum latar belakang tidak dimulai pada relasi.
- Perintah ALTER DISTSTYLE ALL tidak didukung untuk hubungan dengan kunci pengurutan yang disisipkan dan tabel sementara.
- Jika gaya distribusi sebelumnya didefinisikan sebagai AUTO, maka relasi tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

Untuk informasi lebih lanjut tentang DISTSTYLE ALL, buka. BUAT TAMPILAN YANG TERWUJUD

UBAH DISTSTYLE BAHKAN

Klausa yang mengubah gaya distribusi yang ada dari suatu relasi dengan`EVEN`. Pertimbangkan hal berikut:

- ALTER DISTSYTLE, ALTER SORTKEY, dan VACUUM tidak dapat berjalan secara bersamaan pada relasi yang sama.
  - Jika VACUUM saat ini sedang berjalan, maka menjalankan ALTER DISTSTYLE EVEN mengembalikan kesalahan.
  - Jika ALTER DISTSTYLE EVEN berjalan, maka vakum latar belakang tidak dimulai pada relasi.
- Perintah ALTER DISTSTYLE EVEN tidak didukung untuk hubungan dengan kunci pengurutan yang disisipkan dan tabel sementara.
- Jika gaya distribusi sebelumnya didefinisikan sebagai AUTO, maka relasi tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

Untuk informasi lebih lanjut tentang DISTSTYLE EVEN, kunjungiBUAT TAMPILAN YANG TERWUJUD.

UBAH NAMA_kolom DISTKEY atau ALTER DISTSTYLE KEY NAMa_kolom DISTKEY

Klausa yang mengubah kolom yang digunakan sebagai kunci distribusi relasi. Pertimbangkan hal berikut:

- VACUUM dan ALTER DISTKEY tidak dapat berjalan secara bersamaan pada relasi yang sama.
  - Jika VACUUM sudah berjalan, maka ALTER DISTKEY mengembalikan kesalahan.
  - Jika ALTER DISTKEY berjalan, maka vakum latar belakang tidak dimulai pada relasi.
  - Jika ALTER DISTKEY berjalan, maka vakum latar depan mengembalikan kesalahan.
- Anda hanya dapat menjalankan satu perintah ALTER DISTKEY pada relasi pada satu waktu.
- Perintah ALTER DISTKEY tidak didukung untuk hubungan dengan kunci pengurutan yang disisipkan.
- Jika gaya distribusi sebelumnya didefinisikan sebagai AUTO, maka relasi tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

Saat menentukan DISTSTYLE KEY, data didistribusikan oleh nilai-nilai di kolom DISTKEY. Untuk informasi lebih lanjut tentang DISTSTYLE, buka. BUAT TAMPILAN YANG TERWUJUD

UBAH DISTSTYLE AUTO

Klausa yang mengubah gaya distribusi yang ada dari relasi ke AUTO.

Ketika Anda mengubah gaya distribusi menjadi AUTO, gaya distribusi relasi diatur ke berikut:

- Relasi kecil dengan DISTSTYLE ALL diubah menjadi AUTO (ALL).
- Relasi kecil dengan DISTSTYLE EVEN diubah menjadi AUTO (ALL).
- Relasi kecil dengan DISTSTYLE KEY dikonversi ke AUTO (ALL).
- Relasi besar dengan DISTSTYLE ALL diubah menjadi AUTO (EVEN).
- Relasi besar dengan DISTSTYLE EVEN diubah menjadi AUTO (EVEN).
- Relasi besar dengan DISTSTYLE KEY dikonversi ke AUTO (KEY) dan DISTKEY dipertahankan. Dalam hal ini, Amazon Redshift tidak membuat perubahan pada relasi.

Jika Amazon Redshift menentukan bahwa gaya atau kunci distribusi baru akan meningkatkan kinerja kueri, maka Amazon Redshift mungkin mengubah gaya distribusi atau kunci relasi Anda di masa mendatang. Misalnya, Amazon Redshift dapat mengonversi relasi dengan DISTSTYLE AUTO (KEY) menjadi AUTO (EVEN), atau sebaliknya. Untuk informasi selengkapnya tentang perilaku saat kunci distribusi diubah, termasuk redistribusi data dan kunci, buka rekomendasi Amazon Redshift Advisor.

Untuk informasi lebih lanjut tentang DISTSTYLE AUTO, buka. BUAT TAMPILAN YANG TERWUJUD

Untuk melihat gaya distribusi relasi, kueri tampilan katalog sistem SVV_TABLE_INFO. Untuk informasi selengkapnya, kunjungi SVV_TABLE_INFO. Untuk melihat rekomendasi Amazon Redshift Advisor untuk hubungan, kueri tampilan katalog sistem SVV_ALTER_TABLE_RECOMMENDATIONS. Untuk informasi selengkapnya, kunjungi SVV_ALTER_TABLE_REKOMENDASI. Untuk melihat tindakan yang dilakukan oleh Amazon Redshift, kueri tampilan katalog sistem SVL_AUTO_WORKER_ACTION. Untuk informasi selengkapnya, kunjungi SVL_AUTO_WORKER_ACTION.

UBAH [COMPOUND] SORTKEY (nama_ kolom [,...])

Klausa yang mengubah atau menambahkan kunci pengurutan yang digunakan untuk relasi. ALTER SORTKEY tidak didukung untuk tabel sementara.

Ketika Anda mengubah kunci pengurutan, pengkodean kompresi kolom di kunci pengurutan baru atau asli dapat berubah. Jika tidak ada pengkodean yang ditentukan secara eksplisit untuk relasi, maka Amazon Redshift secara otomatis menetapkan pengkodean kompresi sebagai berikut:

- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, atau DOUBLE PRECISION diberi kompresi RAW.

- Kolom yang didefinisikan sebagai SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai CHAR atau VARCHAR diberi kompresi LZO.

Pertimbangkan hal berikut:

- Anda dapat menentukan maksimum 400 kolom untuk kunci pengurutan per relasi.
- Anda dapat mengubah kunci pengurutan yang disisipkan menjadi kunci pengurutan majemuk atau tanpa kunci pengurutan. Namun, Anda tidak dapat mengubah kunci pengurutan majemuk menjadi kunci pengurutan yang disisipkan.
- Jika kunci pengurutan sebelumnya didefinisikan sebagai AUTO, maka relasi tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.
- Amazon Redshift merekomendasikan penggunaan penyandian RAW (tanpa kompresi) untuk kolom yang didefinisikan sebagai kunci pengurutan. Ketika Anda mengubah kolom untuk memilihnya sebagai kunci pengurutan, kompresi kolom diubah menjadi kompresi RAW (tanpa kompresi). Ini dapat meningkatkan jumlah penyimpanan yang dibutuhkan oleh relasi. Seberapa besar ukuran relasi meningkat tergantung pada definisi relasi spesifik dan isi relasi. Untuk informasi lebih lanjut tentang kompresi, bukaPengkodean kompresi.

Ketika data dimuat ke dalam relasi, data dimuat dalam urutan kunci pengurutan. Saat Anda mengubah tombol pengurutan, Amazon Redshift menyusun ulang data. Untuk informasi lebih lanjut tentang SORTKEY, buka. BUAT TAMPILAN YANG TERWUJUD

ALTER SORTKEY OTOMATIS

Klausa yang mengubah atau menambahkan kunci sortir dari relasi target ke AUTO. ALTER SORTKEY AUTO tidak didukung untuk tabel sementara.

Saat Anda mengubah kunci pengurutan menjadi AUTO, Amazon Redshift mempertahankan kunci pengurutan yang ada dari relasi tersebut.

Jika Amazon Redshift menentukan bahwa kunci pengurutan baru akan meningkatkan kinerja kueri, maka Amazon Redshift mungkin mengubah kunci pengurutan relasi Anda di masa mendatang.

Untuk informasi lebih lanjut tentang SORTKEY AUTO, buka. BUAT TAMPILAN YANG TERWUJUD

Untuk melihat kunci pengurutan relasi, kueri tampilan katalog sistem SVV_TABLE_INFO. Untuk informasi selengkapnya, kunjungi SVV_TABLE_INFO. Untuk melihat

rekomendasi Amazon Redshift Advisor untuk hubungan, kueri tampilan katalog sistem SVV_ALTER_TABLE_RECOMMENDATIONS. Untuk informasi selengkapnya, kunjungi SVV_ALTER_TABLE_REKOMENDASI. Untuk melihat tindakan yang dilakukan oleh Amazon Redshift, kueri tampilan katalog sistem SVL_AUTO_WORKER_ACTION. Untuk informasi selengkapnya, kunjungi SVL_AUTO_WORKER_ACTION.

ALTER SORTKEY TIDAK ADA

Klausa yang menghapus kunci pengurutan dari relasi target.

Jika kunci pengurutan sebelumnya didefinisikan sebagai AUTO, maka relasi tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

KEAMANAN TINGKAT BARIS {ON | OFF} [TIPE KONJUNGSI {DAN | ATAU}] [UNTUK BERBAGI DATA]

Klausa yang mengaktifkan atau menonaktifkan keamanan tingkat baris untuk relasi.

Saat keamanan tingkat baris diaktifkan untuk relasi, Anda hanya dapat membaca baris yang diizinkan untuk diakses oleh kebijakan keamanan tingkat baris. Jika tidak ada kebijakan yang memberi Anda akses ke relasi, Anda tidak dapat melihat baris apa pun dari relasi. Hanya pengguna super dan pengguna atau peran yang memiliki `sys:secadmin` peran yang dapat mengatur klausa KEAMANAN TINGKAT BARIS. Untuk informasi selengkapnya, lihat Row-level keamanan.

- [TIPE KONJUNGSI {DAN | ATAU}]

  Klausa yang Anda gunakan untuk memilih jenis konjungsi kebijakan keamanan tingkat baris untuk relasi. Ketika beberapa kebijakan keamanan tingkat baris dilampirkan ke relasi, Anda dapat menggabungkan kebijakan dengan klausa AND atau OR. Secara default, Amazon Redshift menggabungkan kebijakan RLS dengan klausa AND. Pengguna super, pengguna, atau peran yang memiliki `sys:secadmin` peran dapat menggunakan klausa ini untuk menentukan jenis konjungsi kebijakan keamanan tingkat baris untuk relasi. Untuk informasi selengkapnya, lihat Menggabungkan beberapa kebijakan per pengguna.

- UNTUK BERBAGI DATA

  Klausa yang menentukan apakah suatu RLS-protected relasi dapat diakses melalui datashares. Secara default, RLS-protected relasi tidak dapat diakses melalui datashare. Perintah ALTER MATERIALIZED VIEW ROW LEVEL SECURITY yang dijalankan dengan klausa ini hanya memengaruhi properti aksesibilitas datashare relasi. Properti ROW LEVEL SECURITY tidak diubah.

Jika Anda membuat RLS-protected relasi dapat diakses melalui berbagi data, relasi tersebut tidak memiliki keamanan tingkat baris di database yang dibagikan data sisi konsumen. Hubungan mempertahankan properti RLS di sisi produsen.

Contoh

Contoh berikut memungkinkan tampilan ter `tickets_mv` wujud untuk diperbarui secara otomatis.

```
ALTER MATERIALIZED VIEW tickets_mv AUTO REFRESH YES
```

Contoh DISTSTYLE dan SORTKEY untuk ALTER MATERIALIZED VIEW

Contoh dalam topik ini menunjukkan cara melakukan perubahan DISTSTYLE dan SORTKEY, menggunakan ALTER MATERIALIZED VIEW.

Contoh kueri berikut menunjukkan cara mengubah kolom DISTSTYLE KEY DISTKEY menggunakan tabel dasar sampel:

```
CREATE TABLE base_inventory(
  inv_date_sk int4 NOT NULL,
  inv_item_sk int4 NOT NULL,
  inv_warehouse_sk int4 NOT NULL,
  inv_quantity_on_hand int4
);

INSERT INTO base_inventory VALUES(1,1,1,1);

CREATE materialized VIEW inventory diststyle even AS SELECT * FROM base_inventory;
SELECT "table", diststyle FROM svv_table_info WHERE "table" = 'inventory';

ALTER materialized VIEW inventory ALTER diststyle KEY distkey inv_warehouse_sk;
SELECT "table", diststyle FROM svv_table_info WHERE "table" = 'inventory';

ALTER materialized VIEW inventory ALTER distkey inv_item_sk;
SELECT "table", diststyle FROM svv_table_info WHERE "table" = 'inventory';

DROP TABLE base_inventory CASCADE;
```

Ubah tampilan yang terwujud menjadi DISTSTYLE ALL:

```
CREATE TABLE base_inventory(
  inv_date_sk int4 NOT NULL,
  inv_item_sk int4 NOT NULL,
  inv_warehouse_sk int4 NOT NULL,
  inv_quantity_on_hand int4
);

INSERT INTO base_inventory VALUES(1,1,1,1);

CREATE materialized VIEW inventory diststyle even AS SELECT * FROM base_inventory;
SELECT "table", diststyle FROM svv_table_info WHERE "table" = 'inventory';

ALTER MATERIALIZED VIEW inventory ALTER diststyle ALL;
SELECT "table", diststyle FROM svv_table_info WHERE "table" = 'inventory';

DROP TABLE base_inventory CASCADE;
```

Perintah berikut menunjukkan contoh ALTER MATERIALIZED VIEW SORTKEY menggunakan tabel dasar sampel:

```
CREATE TABLE base_inventory (c0 int, c1 int);

INSERT INTO base_inventory VALUES(1,1);

CREATE materialized VIEW inventory interleaved sortkey(c0, c1) AS SELECT * FROM
 base_inventory;
SELECT "table", sortkey1 FROM svv_table_info WHERE "table" = 'inventory';

ALTER materialized VIEW inventory ALTER sortkey(c0, c1);
SELECT "table", diststyle, sortkey_num FROM svv_table_info WHERE "table" = 'inventory';

ALTER materialized VIEW inventory ALTER sortkey NONE;
SELECT "table", diststyle, sortkey_num FROM svv_table_info WHERE "table" = 'inventory';

ALTER materialized VIEW inventory ALTER sortkey(c0);
SELECT "table", diststyle, sortkey_num FROM svv_table_info WHERE "table" = 'inventory';

DROP TABLE base_inventory CASCADE;
```

MENGUBAH KEBIJAKAN RLS

Mengubah kebijakan keamanan tingkat baris yang ada pada tabel.

Pengguna super dan pengguna atau peran yang memiliki `sys:secadmin` peran dapat mengubah kebijakan.

Sintaksis

```
ALTER RLS POLICY
{ policy_name | database_name.policy_name }
USING ( using_predicate_exp );
```

Parameter

nama_kebijakan

Nama kebijakan .

database_name

Nama database tempat kebijakan dibuat. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

MENGGUNAKAN (menggunakan_predic ate_exp)

Menentukan filter yang diterapkan pada klausa WHERE dari query. Amazon Redshift menerapkan predikat kebijakan sebelum predikat pengguna tingkat kueri. Misalnya, **`current_user = 'joe' and price > 10`** membatasi Joe untuk hanya melihat catatan dengan harga lebih besar dari $10.

Ekspresi memiliki akses ke variabel yang dideklarasikan dalam klausa WITH dari pernyataan CREATE RLS POLICY yang digunakan untuk membuat kebijakan dengan nama policy_name.

Untuk penggunaan ALTER RLS POLICY di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

Contoh

Contoh berikut mengubah kebijakan RLS.

```
-- First create an RLS policy that limits access to rows where catgroup is 'concerts'.
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'concerts');
```

```
-- Then, alter the RLS policy to only show rows where catgroup is 'piano concerts'.
ALTER RLS POLICY policy_concerts
USING (catgroup = 'piano concerts');
```

MENGUBAH PERAN

Mengganti nama peran atau mengubah pemilik. Untuk daftar peran yang ditentukan sistem Amazon Redshift, lihat. the section called “Peran yang ditentukan sistem Amazon Redshift”

Izin yang diperlukan

Berikut ini adalah izin yang diperlukan untuk ALTER ROLE:

- Superuser
- Pengguna dengan izin ALTER ROLE

Sintaksis

```
ALTER ROLE role [ WITH ]
  { { RENAME TO role } | { OWNER TO user_name } }[, ...]
  [ EXTERNALID TO external_id ]
```

Parameter

peran

Nama peran yang akan diubah.

GANTI NAMA MENJADI

Nama baru untuk peran tersebut.

PEMILIK KE user_name

Pemilik baru untuk peran tersebut.

EXTERNALID KE external_id

ID eksternal baru untuk peran, yang terkait dengan penyedia identitas. Untuk informasi selengkapnya, lihat Federasi penyedia identitas asli (IdP) untuk Amazon Red shift.

Contoh

Contoh berikut mengubah nama peran dari `sample_role1` menjadi`sample_role2`.

```
ALTER ROLE sample_role1 RENAME TO sample_role2;
```

Contoh berikut mengubah pemilik peran.

```
ALTER ROLE sample_role1 WITH OWNER TO user1
```

Sintaks ALTER ROLE mirip dengan ALTER PROCEDURE berikut.

```
ALTER PROCEDURE first_quarter_revenue(bigint, numeric) RENAME TO quarterly_revenue;
```

Contoh berikut mengubah pemilik prosedur menjadi`etl_user`.

```
ALTER PROCEDURE quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

Contoh berikut memperbarui peran `sample_role1` dengan ID eksternal baru yang terkait dengan penyedia identitas.

```
ALTER ROLE sample_role1 EXTERNALID TO "XYZ456";
```

MENGUBAH PROSEDUR

Mengganti nama prosedur atau mengubah pemilik. Baik nama prosedur dan tipe data, atau tanda tangan, diperlukan. Hanya pemilik atau superuser yang dapat mengganti nama prosedur. Hanya pengguna super yang dapat mengubah pemilik prosedur.

Sintaksis

```
ALTER PROCEDURE sp_name [ ( [ [ argname ] [ argmode ] argtype [, ...] ] ) ]
    RENAME TO new_name
```

```
ALTER PROCEDURE sp_name [ ( [ [ argname ] [ argmode ] argtype [, ...] ] ) ]
    OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
```

Parameter

sp_nama

Nama prosedur yang akan diubah. Entah menentukan hanya nama prosedur di jalur pencarian saat ini, atau gunakan format `schema_name.sp_procedure_name` untuk menggunakan skema tertentu.

[argname] [argmode] argtype

Daftar nama argumen, mode argumen, dan tipe data. Hanya tipe data input yang diperlukan, yang digunakan untuk mengidentifikasi prosedur yang disimpan. Atau, Anda dapat memberikan tanda tangan lengkap yang digunakan untuk membuat prosedur termasuk parameter input dan output dengan mode mereka.

nama_baru

Nama baru untuk prosedur tersimpan.

pemilik baru | PENGGUNA_SAAT INI | SESSION_USER

Pemilik baru untuk prosedur yang disimpan.

Contoh

Contoh berikut mengubah nama prosedur dari `first_quarter_revenue` menjadi`quarterly_revenue`.

```
ALTER PROCEDURE first_quarter_revenue(volume INOUT bigint, at_price IN numeric,
 result OUT int) RENAME TO quarterly_revenue;
```

Contoh ini setara dengan berikut ini.

```
ALTER PROCEDURE first_quarter_revenue(bigint, numeric) RENAME TO quarterly_revenue;
```

Contoh berikut mengubah pemilik prosedur menjadi`etl_user`.

```
ALTER PROCEDURE quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

ALTER SCHEMA

Mengubah definisi skema yang ada. Gunakan perintah ini untuk mengganti nama skema atau mengubah pemilik skema. Misalnya, ganti nama skema yang ada untuk mempertahankan salinan cadangan skema tersebut ketika Anda berencana untuk membuat versi baru skema tersebut. Untuk informasi lebih lanjut tentang skema, lihatBUAT SKEMA.

Untuk melihat kuota skema yang dikonfigurasi, lihatSVV_SCHEMA_KUOTA_STATUS.

Untuk melihat catatan di mana kuota skema terlampaui, lihatPELANGGARAN STL_SCHEMA_KUOTA_.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk ALTER SCHEMA:

- Superuser
- Pengguna dengan hak istimewa ALTER SCHEMA
- Pemilik skema

Saat Anda mengubah nama skema, perhatikan bahwa objek yang menggunakan nama lama, seperti prosedur tersimpan atau tampilan terwujud, harus diperbarui untuk menggunakan nama baru.

Sintaksis

```
ALTER SCHEMA schema_name
{
RENAME TO new_name |
OWNER TO new_owner |
QUOTA { quota [MB | GB | TB] | UNLIMITED }
}
```

Parameter

nama skema

Nama skema database yang akan diubah.

GANTI NAMA MENJADI

Klausa yang mengganti nama skema.

nama_baru

Nama baru skema. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

PEMILIK UNTUK

Klausa yang mengubah pemilik skema.

pemilik baru

Pemilik baru skema.

KUOTA

Jumlah maksimum ruang disk yang dapat digunakan skema tertentu. Ruang ini adalah ukuran kolektif dari semua tabel di bawah skema yang ditentukan. Amazon Redshift mengonversi nilai yang dipilih menjadi megabyte. Gigabyte adalah unit pengukuran default ketika Anda tidak menentukan nilai.

Untuk informasi selengkapnya tentang mengonfigurasi kuota skema, lihatBUAT SKEMA.

Contoh

Contoh berikut mengganti nama skema SALES menjadi US_SALES.

```
alter schema sales
rename to us_sales;
```

Contoh berikut memberikan kepemilikan skema US_SALES kepada pengguna DWUSER.

```
alter schema us_sales
owner to dwuser;
```

Contoh berikut mengubah kuota menjadi 300 GB dan menghapus kuota.

```
alter schema us_sales QUOTA 300 GB;
alter schema us_sales QUOTA UNLIMITED;
```

MENGUBAH SISTEM

Mengubah opsi konfigurasi tingkat sistem untuk cluster Amazon Redshift atau kelompok kerja Redshift Serverless.

Hak istimewa yang diperlukan

Salah satu tipe pengguna berikut dapat menjalankan perintah ALTER SYSTEM:

- Superuser
- Pengguna admin

Sintaksis

```
ALTER SYSTEM SET system-level-configuration = {true| t | on | false | f | off}
```

Parameter

konfigurasi tingkat sistem

Konfigurasi tingkat sistem. Nilai yang valid: `data_catalog_auto_mount` dan`metadata_security`.

{benar| t | aktif | salah | f | mati}

Nilai untuk mengaktifkan atau menonaktifkan konfigurasi tingkat sistem. A`true`,`t`, atau on menunjukkan untuk mengaktifkan konfigurasi. A`false`,`f`, atau `off` menunjukkan untuk menonaktifkan konfigurasi.

Catatan penggunaan

Untuk cluster yang disediakan, perubahan `data_catalog_auto_mount` akan berlaku pada reboot cluster berikutnya. Untuk informasi selengkapnya, lihat Mem-boot ulang cluster di Panduan Manajemen Amazon Redshift.

Untuk workgroup tanpa server, perubahan `data_catalog_auto_mount` tidak segera berlaku.

Contoh

Contoh berikut mengaktifkan pemasangan otomatis. AWS Glue Data Catalog

```
ALTER SYSTEM SET data_catalog_auto_mount = true;
```

Contoh berikut mengaktifkan keamanan metadata.

```
ALTER SYSTEM SET metadata_security = true;
```

Menyetel namespace identitas default

Contoh ini khusus untuk bekerja dengan penyedia identitas. Anda dapat mengintegrasikan Redshift dengan IAM Identity Center dan penyedia identitas untuk memusatkan manajemen identitas untuk Redshift dan layanan lainnya. AWS

Contoh berikut menunjukkan cara mengatur namespace identitas default untuk sistem. Melakukan hal ini selanjutnya membuatnya lebih mudah untuk menjalankan pernyataan GRANT dan CREATE, karena Anda tidak harus menyertakan namespace sebagai awalan untuk setiap identitas.

```
ALTER SYSTEM SET default_identity_namespace = 'MYCO';
```

Setelah menjalankan perintah, Anda dapat menjalankan pernyataan seperti berikut:

```
GRANT SELECT ON TABLE mytable TO alice;

GRANT UPDATE ON TABLE mytable TO salesrole;

CREATE USER bob password 'md50c983d1a624280812631c5389e60d48c';
```

Efek dari pengaturan namespace identitas default adalah bahwa setiap identitas tidak memerlukannya sebagai awalan. Dalam contoh `alice` ini, diganti dengan`MYCO:alice`. Ini terjadi dengan identitas apa pun yang disertakan. Untuk informasi selengkapnya tentang menggunakan penyedia identitas dengan Redshift, lihat Menghubun gkan Redshift dengan Pusat Identitas IAM untuk memberi pengguna pengalaman masuk tunggal.

Untuk informasi selengkapnya tentang pengaturan yang berkaitan dengan konfigurasi Redshift dengan Pusat Identitas IAM, lihat dan. SET UBAH PENYEDIA IDENTITAS

ALTER TABLE

Perintah ini mengubah definisi tabel Amazon Redshift atau tabel eksternal Amazon Redshift Spectrum. Perintah ini memperbarui nilai dan properti yang ditetapkan oleh CREATE TABLE atauCREATE EXTERNAL TABLE. Anda dapat menggunakan ALTER TABLE pada tampilan untuk keamanan tingkat baris (RLS).

Anda tidak dapat menjalankan ALTER TABLE pada tabel eksternal dalam blok transaksi (BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

ALTER TABLE mengunci tabel untuk operasi baca dan tulis sampai transaksi yang menyertakan operasi ALTER TABLE selesai, kecuali dinyatakan secara khusus dalam dokumentasi bahwa Anda dapat menanyakan data atau melakukan operasi lain pada tabel saat diubah.

Hak istimewa yang diperlukan

Pengguna yang mengubah tabel membutuhkan hak istimewa yang tepat agar perintah berhasil. Tergantung pada perintah ALTER TABLE, salah satu hak istimewa berikut diperlukan.

- Superuser
- Pengguna dengan hak istimewa ALTER TABLE
- Pemilik tabel dengan hak istimewa USAGE pada skema

Sintaksis

```
ALTER TABLE table_name
{
ADD table_constraint
| DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
| OWNER TO new_owner
| RENAME TO new_name
| RENAME COLUMN column_name TO new_name
| ALTER COLUMN column_name TYPE updated_varchar_data_type_size
| ALTER COLUMN column_name ENCODE new_encode_type
| ALTER COLUMN column_name ENCODE encode_type,
| ALTER COLUMN column_name ENCODE encode_type, .....;
| ALTER DISTKEY column_name
| ALTER DISTSTYLE ALL
| ALTER DISTSTYLE EVEN
| ALTER DISTSTYLE KEY DISTKEY column_name
| ALTER DISTSTYLE AUTO
| ALTER [COMPOUND] SORTKEY ( column_name [,...] )
| ALTER SORTKEY AUTO
| ALTER SORTKEY NONE
| ALTER ENCODE AUTO
| ADD [ COLUMN ] column_name column_type
  [ DEFAULT default_expr ]
  [ ENCODE encoding ]
  [ NOT NULL | NULL ]
  [ COLLATE { CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI } ] |
| DROP [ COLUMN ] column_name [ RESTRICT | CASCADE ]
```

```
| ROW LEVEL SECURITY { ON | OFF } [ CONJUNCTION TYPE { AND | OR } ] [ FOR DATASHARES ]
| MASKING { ON | OFF } FOR DATASHARES }

where table_constraint is:

[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] )
| PRIMARY KEY ( column_name [, ... ] )
| FOREIGN KEY (column_name [, ... ] )
   REFERENCES  reftable [ ( refcolumn ) ]}

The following options apply only to external tables:

SET LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }
| SET FILE FORMAT format |
| SET TABLE PROPERTIES ('property_name'='property_value')
| PARTITION ( partition_column=partition_value [, ...] )
  SET LOCATION { 's3://bucket/folder' |'s3://bucket/manifest_file' }
| ADD [IF NOT EXISTS]
    PARTITION ( partition_column=partition_value [, ...] ) LOCATION
 { 's3://bucket/folder' |'s3://bucket/manifest_file' }
    [, ... ]
| DROP PARTITION ( partition_column=partition_value [, ...] )
```

Untuk mengurangi waktu untuk menjalankan perintah ALTER TABLE, Anda dapat menggabungkan beberapa klausa dari perintah ALTER TABLE.

Amazon Redshift mendukung kombinasi klausa ALTER TABLE berikut:

```
ALTER TABLE tablename ALTER SORTKEY (column_list), ALTER DISTKEY column_Id;
ALTER TABLE tablename ALTER DISTKEY column_Id, ALTER SORTKEY (column_list);
ALTER TABLE tablename ALTER SORTKEY (column_list), ALTER DISTSTYLE ALL;
ALTER TABLE tablename ALTER DISTSTYLE ALL, ALTER SORTKEY (column_list);
```

Parameter

table_name

Nama tabel yang akan diubah. Entah menentukan hanya nama tabel, atau gunakan format schema_name.table_name untuk menggunakan skema tertentu. Tabel eksternal harus memenuhi syarat dengan nama skema eksternal. Anda juga dapat menentukan nama tampilan jika Anda menggunakan pernyataan ALTER TABLE untuk mengganti nama tampilan atau

mengubah pemiliknya. Panjang maksimum untuk nama tabel adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Anda dapat menggunakan karakter UTF-8 multibyte hingga maksimal empat byte. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

TAMBAHKAN table_restriint

Klausa yang menambahkan batasan yang ditentukan ke tabel. Untuk deskripsi nilai table_restriint yang valid, lihat. CREATE TABLE

> **Note**
>
> Anda tidak dapat menambahkan batasan kunci primer ke kolom yang dapat dibatalkan. Jika kolom awalnya dibuat dengan batasan NOT NULL, Anda dapat menambahkan batasan kunci primer.

DROP CONSTRAINT Constraint_name

Klausa yang menghapus batasan bernama dari tabel. Untuk melepaskan kendala, tentukan nama kendala, bukan jenis kendala. Untuk melihat nama kendala tabel, jalankan kueri berikut.

```
select constraint_name, constraint_type
from information_schema.table_constraints;
```

MEMBATASI

Klausa yang menghapus hanya batasan yang ditentukan. RESTRICT adalah opsi untuk DROP CONSTRAINT. RESTRICT tidak dapat digunakan dengan CASCADE.

KASKADE

Klausa yang menghapus batasan yang ditentukan dan apa pun yang bergantung pada batasan itu. CASCADE adalah opsi untuk DROP CONSTRAINT. CASCADE tidak dapat digunakan dengan RESTRICT.

PEMILIK KE new_owner

Klausa yang mengubah pemilik tabel (atau tampilan) ke nilai new_owner.

GANTI NAMA MENJADI new_name

Klausa yang mengganti nama tabel (atau tampilan) ke nilai yang ditentukan dalam new _name. Panjang nama tabel maksimum adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte.

Anda tidak dapat mengganti nama tabel permanen menjadi nama yang dimulai dengan '#'. Nama tabel yang dimulai dengan '#' menunjukkan tabel sementara.

Anda tidak dapat mengganti nama tabel eksternal.

UBAH KOLOM N_Nama KOLOM TIPE updated_varchar_data_type_size

Klausa yang mengubah ukuran kolom yang didefinisikan sebagai tipe data VARCHAR. Klausa ini hanya mendukung perubahan ukuran tipe data VARCHAR. Pertimbangkan batasan berikut:

- Anda tidak dapat mengubah kolom dengan pengkodean kompresi BYTEDICT, RUNLENGTH, TEXT255, atau TEXT32K.
- Anda tidak dapat mengurangi ukuran kurang dari ukuran maksimum data yang ada.
- Anda tidak dapat mengubah kolom dengan nilai default.
- Anda tidak dapat mengubah kolom dengan UNIQUE, PRIMARY KEY, atau FOREIGN KEY.
- Anda tidak dapat mengubah kolom dalam blok transaksi (BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

UBAH NAMA_ kolom ENCODE new_encode_type

Klausa yang mengubah pengkodean kompresi kolom. Jika Anda menentukan pengkodean kompresi untuk kolom, tabel tidak lagi disetel ke ENCODE AUTO. Untuk informasi tentang pengkodean kompresi, lihatKompresi kolom untuk mengurangi ukuran data yang disimpan.

Saat Anda mengubah pengkodean kompresi untuk kolom, tabel tetap tersedia untuk kueri.

Pertimbangkan batasan berikut:

- Anda tidak dapat mengubah kolom ke pengkodean yang sama seperti yang didefinisikan saat ini untuk kolom.
- Anda tidak dapat mengubah pengkodean untuk kolom dalam tabel dengan sortkey yang disisipkan.

UBAH NAMA_ kolom ENCODE encode_type, ALTER COLUMn_ name ENCODE encode_type, ... ;

Klausa yang mengubah pengkodean kompresi beberapa kolom dalam satu perintah. Untuk informasi tentang pengkodean kompresi, lihatKompresi kolom untuk mengurangi ukuran data yang disimpan.

Saat Anda mengubah pengkodean kompresi untuk kolom, tabel tetap tersedia untuk kueri.

Pertimbangkan batasan berikut:

- Anda tidak dapat mengubah kolom ke jenis pengkodean yang sama atau berbeda beberapa kali dalam satu perintah.
- Anda tidak dapat mengubah kolom ke pengkodean yang sama seperti yang didefinisikan saat ini untuk kolom.
- Anda tidak dapat mengubah pengkodean untuk kolom dalam tabel dengan sortkey yang disisipkan.

UBAH DISTSTYLE SEMUA

Klausa yang mengubah gaya distribusi tabel yang ada menjadiALL. Pertimbangkan hal berikut:

- ALTER DISTSTYLE, ALTER SORTKEY, dan VACUUM tidak dapat berjalan secara bersamaan di tabel yang sama.
  - Jika VACUUM saat ini sedang berjalan, maka menjalankan ALTER DISTSTYLE ALL mengembalikan kesalahan.
  - Jika ALTER DISTSTYLE ALL berjalan, maka penyedot debu latar belakang tidak dimulai di atas meja.
- Perintah ALTER DISTSTYLE ALL tidak didukung untuk tabel dengan kunci pengurutan yang disisipkan dan tabel sementara.
- Jika gaya distribusi sebelumnya didefinisikan sebagai AUTO, maka tabel tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

Untuk informasi lebih lanjut tentang DISTSTYLE ALL, lihatCREATE TABLE.

UBAH DISTSTYLE BAHKAN

Klausa yang mengubah gaya distribusi tabel yang ada menjadiEVEN. Pertimbangkan hal berikut:

- ALTER DISTSYTLE, ALTER SORTKEY, dan VACUUM tidak dapat berjalan secara bersamaan di tabel yang sama.

    - Jika VACUUM saat ini sedang berjalan, maka menjalankan ALTER DISTSTYLE EVEN mengembalikan kesalahan.
    - Jika ALTER DISTSTYLE EVEN berjalan, maka vakum latar belakang tidak dimulai di atas meja.
- Perintah ALTER DISTSTYLE EVEN tidak didukung untuk tabel dengan kunci pengurutan yang disisipkan dan tabel sementara.
- Jika gaya distribusi sebelumnya didefinisikan sebagai AUTO, maka tabel tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

Untuk informasi lebih lanjut tentang DISTSTYLE EVEN, lihatCREATE TABLE.

UBAH NAMA_kolom DISTKEY atau ALTER DISTSTYLE KEY NAMa_kolom DISTKEY

Klausa yang mengubah kolom yang digunakan sebagai kunci distribusi tabel. Pertimbangkan hal berikut:

- VACUUM dan ALTER DISTKEY tidak dapat berjalan secara bersamaan di meja yang sama.
    - Jika VACUUM sudah berjalan, maka ALTER DISTKEY mengembalikan kesalahan.
    - Jika ALTER DISTKEY berjalan, maka vakum latar belakang tidak dimulai di atas meja.
    - Jika ALTER DISTKEY berjalan, maka vakum latar depan mengembalikan kesalahan.
- Anda hanya dapat menjalankan satu perintah ALTER DISTKEY pada tabel pada satu waktu.
- Perintah ALTER DISTKEY tidak didukung untuk tabel dengan kunci pengurutan yang disisipkan.
- Jika gaya distribusi sebelumnya didefinisikan sebagai AUTO, maka tabel tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

Saat menentukan DISTSTYLE KEY, data didistribusikan oleh nilai-nilai di kolom DISTKEY. Untuk informasi lebih lanjut tentang DISTSTYLE, lihatCREATE TABLE.

UBAH DISTSTYLE AUTO

Klausa yang mengubah gaya distribusi tabel yang ada menjadi AUTO.

Ketika Anda mengubah gaya distribusi menjadi AUTO, gaya distribusi tabel diatur ke berikut:

- Sebuah tabel kecil dengan DISTSTYLE ALL diubah menjadi AUTO (ALL).
- Meja kecil dengan DISTSTYLE EVEN diubah menjadi AUTO (ALL).
- Sebuah tabel kecil dengan DISTSTYLE KEY dikonversi ke AUTO (ALL).
- Tabel besar dengan DISTSTYLE ALL diubah menjadi AUTO (EVEN).

- Meja besar dengan DISTSTYLE EVEN diubah menjadi AUTO (EVEN).
- Tabel besar dengan DISTSTYLE KEY dikonversi ke AUTO (KEY) dan DISTKEY dipertahankan. Dalam hal ini, Amazon Redshift tidak membuat perubahan pada tabel.

Jika Amazon Redshift menentukan bahwa gaya atau kunci distribusi baru akan meningkatkan kinerja kueri, maka Amazon Redshift mungkin mengubah gaya distribusi atau kunci tabel Anda di masa mendatang. Misalnya, Amazon Redshift dapat mengonversi tabel dengan DISTSTYLE AUTO (KEY) menjadi AUTO (EVEN), atau sebaliknya. Untuk informasi selengkapnya tentang perilaku saat kunci distribusi diubah, termasuk redistribusi data dan kunci, lihat rekomendasi Amazon Redshift Advisor.

Untuk informasi lebih lanjut tentang DISTSTYLE AUTO, lihatCREATE TABLE.

Untuk melihat gaya distribusi tabel, kueri tampilan katalog sistem SVV_TABLE_INFO. Untuk informasi selengkapnya, lihat SVV_TABLE_INFO. Untuk melihat rekomendasi Amazon Redshift Advisor untuk tabel, kueri tampilan katalog sistem SVV_ALTER_TABLE_RECOMMENDATIONS. Untuk informasi selengkapnya, lihat SVV_ALTER_TABLE_REKOMENDASI. Untuk melihat tindakan yang dilakukan oleh Amazon Redshift, kueri tampilan katalog sistem SVL_AUTO_WORKER_ACTION. Untuk informasi selengkapnya, lihat SVL_AUTO_WORKER_ACTION.

UBAH [COMPOUND] SORTKEY (nama_ kolom [,...])

Klausa yang mengubah atau menambahkan kunci pengurutan yang digunakan untuk tabel. ALTER SORTKEY tidak didukung untuk tabel sementara.

Ketika Anda mengubah kunci pengurutan, pengkodean kompresi kolom di kunci pengurutan baru atau asli dapat berubah. Jika tidak ada pengkodean yang ditentukan secara eksplisit untuk tabel, maka Amazon Redshift secara otomatis menetapkan pengkodean kompresi sebagai berikut:

- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, atau DOUBLE PRECISION diberi kompresi RAW.
- Kolom yang didefinisikan sebagai SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai CHAR atau VARCHAR diberi kompresi LZO.

Pertimbangkan hal berikut:

- Anda dapat menentukan maksimum 400 kolom untuk kunci pengurutan per tabel.

- Anda dapat mengubah kunci pengurutan yang disisipkan menjadi kunci pengurutan majemuk atau tanpa kunci pengurutan. Namun, Anda tidak dapat mengubah kunci pengurutan majemuk menjadi kunci pengurutan yang disisipkan.
- Jika kunci pengurutan sebelumnya didefinisikan sebagai AUTO, maka tabel tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.
- Amazon Redshift merekomendasikan penggunaan penyandian RAW (tanpa kompresi) untuk kolom yang didefinisikan sebagai kunci pengurutan. Ketika Anda mengubah kolom untuk memilihnya sebagai kunci pengurutan, kompresi kolom diubah menjadi kompresi RAW (tanpa kompresi). Ini dapat meningkatkan jumlah penyimpanan yang dibutuhkan oleh tabel. Seberapa besar ukuran tabel meningkat tergantung pada definisi tabel tertentu dan isi tabel. Untuk informasi selengkapnya tentang kompresi, lihat Pengkodean kompresi

Ketika data dimuat ke dalam tabel, data dimuat dalam urutan kunci pengurutan. Saat Anda mengubah tombol pengurutan, Amazon Redshift menyusun ulang data. Untuk informasi lebih lanjut tentang SORTKEY, lihat. CREATE TABLE

ALTER SORTKEY OTOMATIS

Klausa yang mengubah atau menambahkan kunci pengurutan tabel target ke AUTO. ALTER SORTKEY AUTO tidak didukung untuk tabel sementara.

Saat Anda mengubah kunci pengurutan menjadi AUTO, Amazon Redshift mempertahankan kunci pengurutan tabel yang ada.

Jika Amazon Redshift menentukan bahwa kunci pengurutan baru akan meningkatkan kinerja kueri, maka Amazon Redshift mungkin mengubah kunci pengurutan tabel Anda di masa mendatang.

Untuk informasi lebih lanjut tentang SORTKEY AUTO, lihat. CREATE TABLE

Untuk melihat kunci pengurutan tabel, kueri tampilan katalog sistem SVV_TABLE_INFO. Untuk informasi selengkapnya, lihat SVV_TABLE_INFO. Untuk melihat rekomendasi Amazon Redshift Advisor untuk tabel, kueri tampilan katalog sistem SVV_ALTER_TABLE_RECOMMENDATIONS. Untuk informasi selengkapnya, lihat SVV_ALTER_TABLE_REKOMENDASI. Untuk melihat tindakan yang dilakukan oleh Amazon Redshift, kueri tampilan katalog sistem SVL_AUTO_WORKER_ACTION. Untuk informasi selengkapnya, lihat SVL_AUTO_WORKER_ACTION.

ALTER SORTKEY TIDAK ADA

Klausa yang menghapus kunci pengurutan tabel target.

Jika kunci pengurutan sebelumnya didefinisikan sebagai AUTO, maka tabel tidak lagi menjadi kandidat untuk pengoptimalan tabel otomatis.

UBAH PENGKODEAN OTOMATIS

Klausa yang mengubah jenis pengkodean kolom tabel target menjadi AUTO. Saat Anda mengubah pengkodean ke AUTO, Amazon Redshift mempertahankan jenis pengkodean kolom yang ada dalam tabel. Kemudian, jika Amazon Redshift menentukan bahwa jenis pengkodean baru dapat meningkatkan kinerja kueri, Amazon Redshift dapat mengubah jenis pengkodean kolom tabel.

Jika Anda mengubah satu atau beberapa kolom untuk menentukan pengkodean, Amazon Redshift tidak lagi secara otomatis menyesuaikan pengkodean untuk semua kolom dalam tabel. Kolom mempertahankan pengaturan encode saat ini.

Tindakan berikut tidak mempengaruhi pengaturan ENCODE AUTO untuk tabel:

- Mengganti nama tabel.
- Mengubah pengaturan DISTSTYLE atau SORTKEY untuk tabel.
- Menambahkan atau menjatuhkan kolom dengan pengaturan ENCODE.
- Menggunakan opsi COMPDATE dari perintah COPY. Untuk informasi selengkapnya, lihat Operasi beban data.

Untuk melihat pengkodean tabel, kueri tampilan katalog sistem SVV_TABLE_INFO. Untuk informasi selengkapnya, lihat SVV_TABLE_INFO.

GANTI NAMA COLUMN COLUMN_NAME MENJADI new_name

Klausa yang mengganti nama kolom ke nilai yang ditentukan dalam new _name. Panjang nama kolom maksimum adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

TAMBAHKAN [KOLOM] nama_ kolom

Klausa yang menambahkan kolom dengan nama yang ditentukan ke tabel. Anda dapat menambahkan hanya satu kolom di setiap pernyataan ALTER TABLE.

Anda tidak dapat menambahkan kolom yang merupakan kunci distribusi (DISTKEY) atau kunci pengurutan (SORTKEY) dari tabel.

Anda tidak dapat menggunakan perintah ALTER TABLE ADD COLUMN untuk mengubah tabel berikut dan atribut kolom:

- UNIK
- KUNCI UTAMA
- REFERENSI (kunci asing)
- IDENTITAS atau DIHASILKAN SECARA DEFAULT SEBAGAI IDENTITAS

Panjang nama kolom maksimum adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.600.

Batasan berikut berlaku saat menambahkan kolom ke tabel eksternal:

- Anda tidak dapat menambahkan kolom ke tabel eksternal dengan batasan kolom DEFAULT, ENCODE, NOT NULL, atau NULL.
- Anda tidak dapat menambahkan kolom ke tabel eksternal yang ditentukan menggunakan format file AVRO.
- Jika pseudocolumn diaktifkan, jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel eksternal adalah 1.598. Jika pseudocolumn tidak diaktifkan, jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.600.

Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.

tipe_kolom

Jenis data kolom yang ditambahkan. Untuk kolom CHAR dan VARCHAR, Anda dapat menggunakan kata kunci MAX alih-alih mendeklarasikan panjang maksimum. MAX menetapkan panjang maksimum ke 4.096 byte untuk CHAR atau 65.535 byte untuk VARCHAR. Ukuran maksimum objek GEOMETRY adalah 1.048.447 byte.

Untuk informasi tentang tipe data yang didukung Amazon Redshift, lihatJenis Data.

DEFAULT default_expr

Klausa yang menetapkan nilai data default untuk kolom. Tipe data default_expr harus sesuai dengan tipe data kolom. Nilai DEFAULT harus berupa ekspresi bebas variabel. Subquery, referensi silang ke kolom lain dalam tabel saat ini, dan fungsi yang ditentukan pengguna tidak diperbolehkan.

Default_expr digunakan dalam operasi INSERT apa pun yang tidak menentukan nilai untuk kolom. Jika tidak ada nilai default yang ditentukan, nilai default untuk kolom adalah null.

Jika operasi COPY menemukan bidang null pada kolom yang memiliki nilai DEFAULT dan batasan NOT NULL, perintah COPY menyisipkan nilai default _expr.

DEFAULT tidak didukung untuk tabel eksternal.

ENCODE pengkodean

Pengkodean kompresi untuk kolom. Secara default, Amazon Redshift secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel jika Anda tidak menentukan pengkodean kompresi untuk kolom apa pun dalam tabel atau jika Anda menentukan opsi ENCODE AUTO untuk tabel.

Jika Anda menentukan pengkodean kompresi untuk kolom apa pun dalam tabel atau jika Anda tidak menentukan opsi ENCODE AUTO untuk tabel, Amazon Redshift secara otomatis menetapkan pengkodean kompresi ke kolom yang tidak Anda tentukan pengkodean kompresi sebagai berikut:

- Semua kolom dalam tabel sementara diberi kompresi RAW secara default.
- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, DOUBLE PRECISION, GEOMETRY, atau GEOGRAPHY diberi kompresi RAW.
- Kolom yang didefinisikan sebagai SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai CHAR, VARCHAR, atau VARBYTE diberi kompresi LZO.

> **Note**
>
> Jika Anda tidak ingin kolom dikompresi, tentukan pengkodean RAW secara eksplisit.

compression encodings (p. 78) berikut didukung:

- AZ64
- BYTEDICT
- DELTA
- DELTA32K
- LZO
- MAYORIT8
- MAYORIT16
- MAYORIT32

- RAW (tanpa kompresi)
- PANJANG
- TEKS255
- TEKS32K
- ZSTD

ENCODE tidak didukung untuk tabel eksternal.

BUKAN NULL | NULL

NOT NULL menentukan bahwa kolom tidak diizinkan untuk berisi nilai null. NULL, default, menentukan bahwa kolom menerima nilai null.

NOT NULL dan NULL tidak didukung untuk tabel eksternal.

SUSUN {CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI}

Klausa yang menentukan apakah pencarian string atau perbandingan pada kolom peka huruf besar/kecil atau tidak peka huruf besar/kecil. Nilai defaultnya sama dengan konfigurasi sensitivitas huruf besar saat ini dari database.

Untuk memeriksa kolasi database saat ini, gunakan DB_COLLATION fungsi.

CASE_SENSITIVE dan CS dapat dipertukarkan dan menghasilkan hasil yang sama. Demikian pula, CASE_INSENSITIVE dan CI dapat dipertukarkan dan menghasilkan hasil yang sama.

JATUHKAN [KOLOM] nama_ kolom

Nama kolom yang akan dihapus dari tabel.

Anda tidak dapat menjatuhkan kolom terakhir dalam tabel. Sebuah tabel harus memiliki setidaknya satu kolom.

Anda tidak dapat melepaskan kolom yang merupakan kunci distribusi (DISTKEY) atau kunci pengurutan (SORTKEY) dari tabel. Perilaku default untuk DROP COLUMN adalah RESTRICT jika kolom memiliki objek dependen, seperti tampilan, kunci primer, kunci asing, atau pembatasan UNIK.

Batasan berikut berlaku saat menjatuhkan kolom dari tabel eksternal:

- Anda tidak dapat melepaskan kolom dari tabel eksternal jika kolom tersebut digunakan sebagai partisi.

- Anda tidak dapat melepaskan kolom dari tabel eksternal yang ditentukan menggunakan format file AVRO.
- RESTRICT dan CASCADE diabaikan untuk tabel eksternal.
- Anda tidak dapat menghapus kolom tabel kebijakan yang direferensikan di dalam definisi kebijakan kecuali Anda menghapus atau melepaskan kebijakan tersebut. Ini juga berlaku ketika opsi CASCADE ditentukan. Anda dapat melepaskan kolom lain di tabel kebijakan.

Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.

MEMBATASI

Ketika digunakan dengan DROP COLUMN, RESTRICT berarti kolom yang akan dijatuhkan tidak dijatuhkan, dalam kasus ini:

- Jika tampilan yang ditentukan mereferensikan kolom yang sedang dijatuhkan
- Jika kunci asing mereferensikan kolom
- Jika kolom mengambil bagian dalam kunci multipart

RESTRICT tidak dapat digunakan dengan CASCADE.

RESTRICT dan CASCADE diabaikan untuk tabel eksternal.

KASKADE

Saat digunakan dengan DROP COLUMN, menghapus kolom yang ditentukan dan apa pun yang bergantung pada kolom itu. CASCADE tidak dapat digunakan dengan RESTRICT.

RESTRICT dan CASCADE diabaikan untuk tabel eksternal.

Opsi berikut hanya berlaku untuk tabel eksternal.

TETAPKAN LOKASI {'s3://bucket/folder/' | 's3://'} bucket/manifest_file

Jalur ke folder Amazon S3 yang berisi file data atau file manifes yang berisi daftar jalur objek Amazon S3. Bucket harus berada di Wil AWS ayah yang sama dengan cluster Amazon Redshift. Untuk daftar Wilayah yang didukung AWS , lihatBatasan Amazon Redshift Spectrum. Untuk informasi selengkapnya tentang menggunakan file manifes, lihat LOKASI di Parameter referensi CREATE EXTERNAL TABLE.

ATUR FORMAT FILE

Format file untuk file data eksternal.

Format yang valid adalah sebagai berikut:

- AVRO
- PARQUET
- RCFILE
- SEQUENCEFILE
- TEXTFILE

ATUR PROPERTI TABEL ('property_name' = 'property_value')

Klausa yang menetapkan definisi tabel untuk properti tabel untuk tabel eksternal.

> **Note**
>
> Properti tabel peka huruf besar/kecil.

'numrow s' = 'hitung_baris'

Properti yang menetapkan nilai numRows untuk definisi tabel. Untuk memperbarui statistik tabel eksternal secara eksplisit, atur properti numRows untuk menunjukkan ukuran tabel. Amazon Redshift tidak menganalisis tabel eksternal untuk menghasilkan statistik tabel yang digunakan pengoptimal kueri untuk menghasilkan rencana kueri. Jika statistik tabel tidak disetel untuk tabel eksternal, Amazon Redshift akan menghasilkan rencana eksekusi kueri. Rencana ini didasarkan pada asumsi bahwa tabel eksternal adalah tabel yang lebih besar dan tabel lokal adalah tabel yang lebih kecil.

'skip.header.line.count'=' hitung_baris '

Properti yang menetapkan jumlah baris untuk dilewati di awal setiap file sumber.

'format- version'=' versi '

Menentukan versi format tabel Iceberg. Nilai yang mungkin: '2' (default), '3'. Format versi 3 mendukung kemampuan tambahan seperti nilai kolom default. Pada waktu CREATE, jika tidak ditentukan, Amazon Redshift membuat tabel sebagai Iceberg v2. Pada tabel yang ada, menyetel properti ini ke '3' meningkatkan tabel dari v2 ke v3.

PARTISI (partisi_kolom = partisi_nilai [,...] TETAPKAN LOKASI {'s3://bucket/folder '| 's3://bucket/manifest _file'}

Klausa yang menetapkan lokasi baru untuk satu atau lebih kolom partisi.

TAMBAHKAN [JIKA TIDAK ADA] PARTISI (partition_column = partition_value [,...]) LOKASI {'s3://bucket/folder '| 's3://bucket/manifest_file'} [,...]

Sebuah klausa yang menambahkan satu atau lebih partisi. Anda dapat menentukan beberapa klausa PARTITION menggunakan pernyataan ALTER TABLE... ADD tunggal.

> **Note**
>
> Jika Anda menggunakan AWS Glue katalog, Anda dapat menambahkan hingga 100 partisi menggunakan pernyataan ALTER TABLE tunggal.

Klausa IF NOT EXISTS menunjukkan bahwa jika partisi yang ditentukan sudah ada, perintah tidak boleh membuat perubahan. Ini juga menunjukkan bahwa perintah harus mengembalikan pesan bahwa partisi ada, daripada berakhir dengan kesalahan. Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika ALTER TABLE mencoba menambahkan partisi yang sudah ada.

JATUHKAN PARTISI (partisi_kolom = partisi_nilai [,...])

Klausa yang menjatuhkan partisi yang ditentukan. Menjatuhkan partisi hanya mengubah metadata tabel eksternal. Data di Amazon S3 tidak terpengaruh.

KEAMANAN TINGKAT BARIS {ON | OFF} [TIPE KONJUNGSI {DAN | ATAU}] [UNTUK BERBAGI DATA]

Klausa yang mengaktifkan atau menonaktifkan keamanan tingkat baris untuk relasi.

Saat keamanan tingkat baris diaktifkan untuk relasi, Anda hanya dapat membaca baris yang diizinkan untuk diakses oleh kebijakan keamanan tingkat baris. Jika tidak ada kebijakan yang memberi Anda akses ke relasi, Anda tidak dapat melihat baris apa pun dari relasi. Hanya pengguna super dan pengguna atau peran yang memiliki `sys:secadmin` peran yang dapat mengatur klausa KEAMANAN TINGKAT BARIS. Untuk informasi selengkapnya, lihat Row-level keamanan. Pernyataan ini didukung pada database yang terhubung atau pada database dengan izin federasi amazon redshift. Klausa FOR DATASHARES tidak didukung pada database dengan izin federasi Amazon Redshift.

- [TIPE KONJUNGSI {DAN | ATAU}]

  Klausa yang Anda gunakan untuk memilih jenis konjungsi kebijakan keamanan tingkat baris untuk relasi. Ketika beberapa kebijakan keamanan tingkat baris dilampirkan ke relasi, Anda dapat menggabungkan kebijakan dengan klausa AND atau OR. Secara default, Amazon Redshift menggabungkan kebijakan RLS dengan klausa AND. Pengguna super, pengguna,

atau peran yang memiliki `sys:secadmin` peran dapat menggunakan klausa ini untuk menentukan jenis konjungsi kebijakan keamanan tingkat baris untuk relasi. Untuk informasi selengkapnya, lihat Menggabungkan beberapa kebijakan per pengguna.

- UNTUK BERBAGI DATA

  Klausa yang menentukan apakah suatu RLS-protected relasi dapat diakses melalui datashares. Secara default, RLS-protected relasi tidak dapat diakses melalui datashare. Perintah ALTER TABLE ROW LEVEL SECURITY yang dijalankan dengan klausa ini hanya memengaruhi properti aksesibilitas datashare relasi. Properti ROW LEVEL SECURITY tidak diubah.

  Jika Anda membuat RLS-protected relasi dapat diakses melalui berbagi data, relasi tersebut tidak memiliki keamanan tingkat baris di database yang dibagikan data sisi konsumen. Hubungan mempertahankan properti RLS di sisi produsen.

MASKING {ON | OFF} UNTUK BERBAGI DATA

Klausa yang menentukan apakah DDM-protected relasi dapat diakses melalui datashares. Secara default, DDM-protected relasi tidak dapat diakses melalui datashare. Jika Anda membuat DDM-protected relasi dapat diakses melalui berbagi data, relasi tidak akan memiliki perlindungan masking di database yang dibagikan data sisi konsumen. Hubungan mempertahankan properti masking di sisi produsen. Hanya pengguna super dan pengguna atau peran yang memiliki peran yang dapat mengatur `sys:secadmin` klausa MASKING FOR DATASHARES. Untuk informasi selengkapnya, lihat Penyembunyian data dinamis.

Contoh

Untuk contoh yang menunjukkan cara menggunakan perintah ALTER TABLE, lihat berikut ini.

- Contoh ALTER TABLE
- ALTER CONTOH TABEL EKSTERNAL
- Contoh ALTER TABLE ADD dan DROP COLUMN

Contoh ALTER TABLE

Contoh berikut menunjukkan penggunaan dasar dari perintah ALTER TABLE.

Mengganti nama tabel atau tampilan

Perintah berikut mengganti nama tabel USERS menjadi USERS_BKUP:

```
alter table users
rename to users_bkup;
```

Anda juga dapat menggunakan jenis perintah ini untuk mengganti nama tampilan.

Mengubah pemilik tabel atau tampilan

Perintah berikut mengubah pemilik tabel VENUE menjadi pengguna DWUSER:

```
alter table venue
owner to dwuser;
```

Perintah berikut membuat tampilan, lalu ubah pemiliknya:

```
create view vdate as select * from date;
alter table vdate owner to vuser;
```

Ubah Nama Kolom

Perintah berikut mengganti nama kolom VENUESEATS dalam tabel VENUE menjadi VENUESIZE:

```
alter table venue
rename column venueseats to venuesize;
```

Menjatuhkan kendala tabel

Untuk melepaskan batasan tabel, seperti kunci primer, kunci asing, atau batasan unik, pertama-tama temukan nama internal kendala tersebut. Kemudian tentukan nama kendala dalam perintah ALTER TABLE. Contoh berikut menemukan batasan untuk tabel CATEGORY, lalu melepas kunci utama dengan nama`category_pkey`.

```
select constraint_name, constraint_type
from information_schema.table_constraints
where constraint_schema ='public'
and table_name = 'category';

constraint_name | constraint_type
----------------+----------------
category_pkey   | PRIMARY KEY
```

```
alter table category
drop constraint category_pkey;
```

Mengubah kolom VARCHAR

Untuk menghemat penyimpanan, Anda dapat menentukan tabel pada awalnya dengan kolom VARCHAR dengan ukuran minimum yang diperlukan untuk kebutuhan data Anda saat ini. Kemudian, untuk mengakomodasi string yang lebih panjang, Anda dapat mengubah tabel untuk meningkatkan ukuran kolom.

Contoh berikut meningkatkan ukuran kolom EVENTNAME menjadi VARCHAR (300).

```
alter table event alter column eventname type varchar(300);
```

Mengubah kolom VARBYTE

Untuk menghemat penyimpanan, Anda dapat menentukan tabel pada awalnya dengan kolom VARBYTE dengan ukuran minimum yang diperlukan untuk kebutuhan data Anda saat ini. Kemudian, untuk mengakomodasi string yang lebih panjang, Anda dapat mengubah tabel untuk meningkatkan ukuran kolom.

Contoh berikut meningkatkan ukuran kolom EVENTNAME menjadi VARBYTE (300).

```
alter table event alter column eventname type varbyte(300);
```

Mengubah pengkodean kompresi untuk kolom

Anda dapat mengubah pengkodean kompresi kolom. Di bawah ini, Anda dapat menemukan serangkaian contoh yang menunjukkan pendekatan ini. Definisi tabel untuk contoh-contoh ini adalah sebagai berikut.

```
create table t1(c0 int encode lzo, c1 bigint encode zstd, c2 varchar(16) encode lzo, c3
 varchar(32) encode zstd);
```

Pernyataan berikut mengubah pengkodean kompresi untuk kolom c0 dari pengkodean LZO ke pengkodean AZ64.

```
alter table t1 alter column c0 encode az64;
```

Pernyataan berikut mengubah pengkodean kompresi untuk kolom c1 dari pengkodean Zstandard ke pengkodean AZ64.

```
alter table t1 alter column c1 encode az64;
```

Pernyataan berikut mengubah pengkodean kompresi untuk kolom c2 dari pengkodean LZO menjadi Byte-dictionary encoding.

```
alter table t1 alter column c2 encode bytedict;
```

Pernyataan berikut mengubah pengkodean kompresi untuk kolom c3 dari pengkodean Zstandard ke pengkodean Runlength.

```
alter table t1 alter column c3 encode runlength;
```

Mengubah kolom DISTSTYLE KEY DISTKEY

Contoh berikut menunjukkan cara mengubah DISTSTYLE dan DISTKEY dari sebuah tabel.

Buat tabel dengan gaya distribusi EVEN. Tampilan SVV_TABLE_INFO menunjukkan bahwa DISTSTYLE adalah EVEN.

```
create table inventory(
  inv_date_sk int4 not null ,
  inv_item_sk int4 not null ,
  inv_warehouse_sk int4 not null ,
  inv_quantity_on_hand int4
) diststyle even;

Insert into inventory values(1,1,1,1);

select "table", "diststyle" from svv_table_info;

   table   |   diststyle
-----------+----------------
 inventory |     EVEN
```

Ubah tabel DISTKEY menjadi`inv_warehouse_sk`. Tampilan SVV_TABLE_INFO menunjukkan `inv_warehouse_sk` kolom sebagai kunci distribusi yang dihasilkan.

```
alter table inventory alter diststyle key distkey inv_warehouse_sk;
```

```
select "table", "diststyle" from svv_table_info;

   table   |       diststyle
-----------+----------------------
 inventory | KEY(inv_warehouse_sk)
```

Ubah tabel DISTKEY menjadi`inv_item_sk`. Tampilan SVV_TABLE_INFO menunjukkan `inv_item_sk` kolom sebagai kunci distribusi yang dihasilkan.

```
alter table inventory alter distkey inv_item_sk;

select "table", "diststyle" from svv_table_info;

   table   |       diststyle
-----------+----------------------
 inventory | KEY(inv_item_sk)
```

Ubah tabel menjadi DISTSTYLE ALL

Contoh berikut menunjukkan cara mengubah tabel menjadi DISTSTYLE ALL.

Buat tabel dengan gaya distribusi EVEN. Tampilan SVV_TABLE_INFO menunjukkan bahwa DISTSTYLE adalah EVEN.

```
create table inventory(
  inv_date_sk int4 not null ,
  inv_item_sk int4 not null ,
  inv_warehouse_sk int4 not null ,
  inv_quantity_on_hand int4
) diststyle even;

Insert into inventory values(1,1,1,1);

select "table", "diststyle" from svv_table_info;

   table   |   diststyle
-----------+---------------
 inventory |     EVEN
```

Ubah tabel DISTSTYLE menjadi ALL. Tampilan SVV_TABLE_INFO menunjukkan DISTSYTLE yang diubah.

```
alter table inventory alter diststyle all;

select "table", "diststyle" from svv_table_info;

   table   |   diststyle
-----------+----------------
 inventory |     ALL
```

Mengubah tabel SORTKEY

Anda dapat mengubah tabel untuk memiliki kunci pengurutan majemuk atau tanpa kunci pengurutan.

Dalam definisi tabel berikut, tabel `t1` didefinisikan dengan sortkey yang disisipkan.

```
create table t1 (c0 int, c1 int) interleaved sortkey(c0, c1);
```

Perintah berikut mengubah tabel dari kunci pengurutan yang disisipkan menjadi kunci pengurutan majemuk.

```
alter table t1 alter sortkey(c0, c1);
```

Perintah berikut mengubah tabel untuk menghapus kunci pengurutan yang disisipkan.

```
alter table t1 alter sortkey none;
```

Dalam definisi tabel berikut, tabel `t1` didefinisikan dengan kolom `c0` sebagai sortkey.

```
create table t1 (c0 int, c1 int) sortkey(c0);
```

Perintah berikut mengubah tabel menjadi kunci peng `t1` urutan majemuk.

```
alter table t1 alter sortkey(c0, c1);
```

Ubah tabel menjadi ENCODE AUTO

Contoh berikut menunjukkan cara mengubah tabel menjadi ENCODE AUTO.

Definisi tabel untuk contoh ini berikut. Kol `c0` om didefinisikan dengan tipe pengkodean AZ64, dan kolom `c1` didefinisikan dengan tipe pengkodean LZO.

```
create table t1(c0 int encode AZ64, c1 varchar encode LZO);
```

Untuk tabel ini, pernyataan berikut mengubah pengkodean menjadi AUTO.

```
alter table t1 alter encode auto;
```

Contoh berikut menunjukkan cara mengubah tabel untuk menghapus pengaturan ENCODE AUTO.

Definisi tabel untuk contoh ini berikut. Kolom tabel didefinisikan tanpa pengkodean. Dalam hal ini, pengkodean default ke ENCODE AUTO.

```
create table t2(c0 int, c1 varchar);
```

Untuk tabel ini, pernyataan berikut mengubah pengkodean kolom c0 menjadi LZO. Pengkodean tabel tidak lagi diatur ke ENCODE AUTO.

```
alter table t2 alter column c0 encode lzo;;
```

Mengubah kontrol keamanan tingkat baris

Perintah berikut mematikan RLS untuk tabel:

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY OFF;
```

Perintah berikut mengaktifkan RLS untuk tabel:

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
```

Perintah berikut mengaktifkan RLS untuk tabel dan membuatnya dapat diakses melalui datashares:

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY FOR DATASHARES OFF;
```

Perintah berikut mengaktifkan RLS untuk tabel dan membuatnya tidak dapat diakses melalui data share:

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
```

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY FOR DATASHARES ON;
```

Perintah berikut mengaktifkan RLS dan menyetel tipe konjungsi RLS ke OR untuk tabel:

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON CONJUNCTION TYPE OR;
```

Perintah berikut mengaktifkan RLS dan menyetel tipe konjungsi RLS ke AND untuk tabel:

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON CONJUNCTION TYPE AND;
```

ALTER CONTOH TABEL EKSTERNAL

Contoh berikut menggunakan bucket Amazon S3 yang terletak di Wilayah AS Timur (Virginia Utara) (`us-east-1`) Wilayah AWS dan tabel contoh yang dibuat Contoh untuk CREATE TABLE. Untuk informasi selengkapnya tentang cara menggunakan partisi dengan tabel eksternal, lihatPartisi tabel eksternal Redshift Spectrum.

Contoh berikut menetapkan properti tabel numRows untuk tabel SPECTRUM.SALES eksternal ke 170.000 baris.

```
alter table spectrum.sales
set table properties ('numRows'='170000');
```

Contoh berikut mengubah lokasi untuk tabel SPECTRUM.SALES eksternal.

```
alter table spectrum.sales
set location 's3://redshift-downloads/tickit/spectrum/sales/';
```

Contoh berikut mengubah format untuk tabel SPECTRUM.SALES eksternal menjadi Parquet.

```
alter table spectrum.sales
set file format parquet;
```

Contoh berikut menambahkan satu partisi untuk tabel SPECTRUM.SALES _PART.

```
alter table spectrum.sales_part
add if not exists partition(saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/';
```

Contoh berikut menambahkan tiga partisi untuk tabel SPECTRUM.SALES _PART.

```
alter table spectrum.sales_part add if not exists
partition(saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/'
partition(saledate='2008-02-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/'
partition(saledate='2008-03-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/';
```

Contoh berikut mengubah SPECTRUM.SALES _PART untuk menghapus partisi dengan. `saledate='2008-01-01''`

```
alter table spectrum.sales_part
drop partition(saledate='2008-01-01');
```

Contoh berikut menetapkan jalur Amazon S3 baru untuk partisi dengan`saledate='2008-01-01'`.

```
alter table spectrum.sales_part
partition(saledate='2008-01-01')
set location 's3://redshift-downloads/tickit/spectrum/sales_partition/
saledate=2008-01-01/';
```

Contoh berikut mengubah nama `sales_date` menjadi`transaction_date`.

```
alter table spectrum.sales rename column sales_date to transaction_date;
```

Contoh berikut menetapkan pemetaan kolom ke pemetaan posisi untuk tabel eksternal yang menggunakan format kolom baris (ORC) yang dioptimalkan.

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='position');
```

Contoh berikut menetapkan pemetaan kolom untuk pemetaan nama untuk tabel eksternal yang menggunakan format ORC.

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='name');
```

Contoh ALTER TABLE ADD dan DROP COLUMN

Contoh berikut menunjukkan cara menggunakan ALTER TABLE untuk menambahkan dan kemudian melepas kolom tabel dasar dan juga cara melepaskan kolom dengan objek dependen.

TAMBAHKAN lalu JATUHKAN kolom dasar

Contoh berikut menambahkan kolom FEEDBACK_SCORE mandiri ke tabel USERS. Kolom ini hanya berisi bilangan bulat, dan nilai default untuk kolom ini adalah NULL (tidak ada skor umpan balik).

Pertama, kueri tabel katalog PG_TABLE_DEF untuk melihat skema tabel USERS:

```
column        | type                    | encoding | distkey | sortkey
--------------+-------------------------+----------+---------+--------
userid        | integer                 | delta    | true    |       1
username      | character(8)            | lzo      | false   |       0
firstname     | character varying(30)   | text32k  | false   |       0
lastname      | character varying(30)   | text32k  | false   |       0
city          | character varying(30)   | text32k  | false   |       0
state         | character(2)            | bytedict | false   |       0
email         | character varying(100)  | lzo      | false   |       0
phone         | character(14)           | lzo      | false   |       0
likesports    | boolean                 | none     | false   |       0
liketheatre   | boolean                 | none     | false   |       0
likeconcerts  | boolean                 | none     | false   |       0
likejazz      | boolean                 | none     | false   |       0
likeclassical | boolean                 | none     | false   |       0
likeopera     | boolean                 | none     | false   |       0
likerock      | boolean                 | none     | false   |       0
likevegas     | boolean                 | none     | false   |       0
likebroadway  | boolean                 | none     | false   |       0
likemusicals  | boolean                 | none     | false   |       0
```

Sekarang tambahkan kolom feedback_score:

```
alter table users
add column feedback_score int
default NULL;
```

Pilih kolom FEEDBACK_SCORE dari USERS untuk memverifikasi bahwa itu telah ditambahkan:

```
select feedback_score from users limit 5;
```

```
feedback_score
----------------
NULL
NULL
NULL
NULL
NULL
```

Jatuhkan kolom untuk mengembalikan DDL asli:

```
alter table users drop column feedback_score;
```

Menjatuhkan kolom dengan objek dependen

Contoh berikut menjatuhkan kolom yang memiliki objek dependen. Akibatnya, objek dependen juga dijatuhkan.

Untuk memulai, tambahkan kolom FEEDBACK_SCORE ke tabel USERS lagi:

```
alter table users
add column feedback_score int
default NULL;
```

Selanjutnya, buat tampilan dari tabel USERS yang disebut USERS_VIEW:

```
create view users_view as select * from users;
```

Sekarang, coba jatuhkan kolom FEEDBACK_SCORE dari tabel USERS. Pernyataan DROP ini menggunakan perilaku default (RESTRICT):

```
alter table users drop column feedback_score;
```

Amazon Redshift menampilkan pesan kesalahan bahwa kolom tidak dapat dijatuhkan karena objek lain bergantung padanya.

Coba jatuhkan kolom FEEDBACK_SCORE lagi, kali ini tentukan CASCADE untuk menghapus semua objek dependen:

```
alter table users
drop column feedback_score cascade;
```

UBAH TABEL DITAMBAHKAN

Menambahkan baris ke tabel target dengan memindahkan data dari tabel sumber yang ada. Data dalam tabel sumber dipindahkan ke kolom yang cocok di tabel target. Urutan kolom tidak masalah. Setelah data berhasil ditambahkan ke tabel target, tabel sumber kosong. ALTER TABLE APPEND biasanya jauh lebih cepat daripada operasi serupa CREATE TABLE AS atau IN INSERT TO karena data dipindahkan, tidak diduplikasi.

> **Note**
>
> ALTER TABLE APPEND memindahkan blok data antara tabel sumber dan tabel target. Untuk meningkatkan kinerja, ALTER TABLE APPEND tidak memadatkan penyimpanan sebagai bagian dari operasi append. Akibatnya, penggunaan penyimpanan meningkat sementara. Untuk merebut kembali ruang, jalankan VAKUM operasi.

Kolom dengan nama yang sama juga harus memiliki atribut kolom yang identik. Jika tabel sumber atau tabel target berisi kolom yang tidak ada di tabel lain, gunakan parameter IGNOREEXTRA atau FILLTARGET untuk menentukan bagaimana kolom tambahan harus dikelola.

Anda tidak dapat menambahkan kolom identitas. Jika kedua tabel menyertakan kolom identitas, perintah gagal. Jika hanya satu tabel yang memiliki kolom identitas, sertakan parameter FILLTARGET atau IGNOREEXTRA. Untuk informasi selengkapnya, lihat ALTER TABLE MENAMBAHKAN catatan penggunaan.

Anda dapat menambahkan kolom GENERATED BY DEFAULT AS IDENTITY. Anda dapat memperbarui kolom yang didefinisikan sebagai GENERATED BY DEFAULT AS IDENTITY dengan nilai-nilai yang Anda berikan. Untuk informasi selengkapnya, lihat ALTER TABLE MENAMBAHKAN catatan penggunaan.

Tabel target harus berupa tabel permanen. Namun, sumbernya dapat berupa tabel permanen atau tampilan terwujud yang dikonfigurasi untuk konsumsi streaming. Kedua objek harus menggunakan gaya distribusi dan kunci distribusi yang sama, jika salah satu didefinisikan. Jika objek diurutkan, kedua objek harus menggunakan gaya pengurutan yang sama dan menentukan kolom yang sama dengan kunci pengurutan.

Perintah ALTER TABLE APPEND secara otomatis dikomit segera setelah operasi selesai. Itu tidak bisa diputar kembali. Anda tidak dapat menjalankan ALTER TABLE APPEND dalam blok transaksi

(BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Hak istimewa yang diperlukan

Bergantung pada perintah ALTER TABLE APPEND, salah satu hak istimewa berikut diperlukan:

- Superuser
- Pengguna dengan hak istimewa sistem ALTER TABLE
- Pengguna dengan hak DELETE dan SELECT pada tabel sumber, dan hak istimewa INSERT pada tabel target

Sintaksis

```
ALTER TABLE target_table_name APPEND FROM [ source_table_name
 | source_materialized_view_name ]
[ IGNOREEXTRA | FILLTARGET ]
```

Menambahkan dari tampilan terwujud hanya berfungsi dalam kasus di mana tampilan terwujud Anda dikonfigurasi untuk. Streaming konsumsi ke tampilan yang terwujud

Parameter

nama_tabel_target

Nama tabel yang barisnya ditambahkan. Entah menentukan hanya nama tabel atau gunakan format schema_name.table_name untuk menggunakan skema tertentu. Tabel target harus berupa tabel permanen yang ada.

DARI source_table_name

Nama tabel yang menyediakan baris yang akan ditambahkan. Entah menentukan hanya nama tabel atau gunakan format schema_name.table_name untuk menggunakan skema tertentu. Tabel sumber harus berupa tabel permanen yang ada.

DARI source_materialized_view_name

Nama tampilan terwujud yang menyediakan baris yang akan ditambahkan. Menambahkan dari tampilan terwujud hanya berfungsi dalam kasus di mana tampilan terwujud Anda dikonfigurasi untuk. Streaming konsumsi ke tampilan yang terwujud Tampilan sumber yang terwujud harus sudah ada.

ABAIKAN EKSTRA

Kata kunci yang menentukan bahwa jika tabel sumber menyertakan kolom yang tidak ada dalam tabel target, data di kolom tambahan harus dibuang. Anda tidak dapat menggunakan IGNOREEXTRA dengan FILLTARGET.

TARGET ISI

Kata kunci yang menentukan bahwa jika tabel target menyertakan kolom yang tidak ada dalam tabel sumber, kolom harus diisi dengan nilai DEFAULT kolom, jika ada yang ditentukan, atau NULL. Anda tidak dapat menggunakan IGNOREEXTRA dengan FILLTARGET.

ALTER TABLE MENAMBAHKAN catatan penggunaan

- ALTER TABLE APPEND hanya memindahkan kolom identik dari tabel sumber ke tabel target. Urutan kolom tidak masalah.
- Jika tabel sumber atau tabel target berisi kolom tambahan, gunakan FILLTARGET atau IGNOREEXTRA sesuai dengan aturan berikut:
  - Jika tabel sumber berisi kolom yang tidak ada di tabel target, sertakan IGNOREEXTRA. Perintah mengabaikan kolom tambahan dalam tabel sumber.
  - Jika tabel target berisi kolom yang tidak ada di tabel sumber, sertakan FILLTARGET. Perintah mengisi kolom tambahan dalam tabel target dengan nilai kolom default atau nilai IDENTITY, jika didefinisikan, atau NULL.
  - Jika tabel sumber dan tabel target berisi kolom tambahan, perintah gagal. Anda tidak dapat menggunakan FILLTARGET dan IGNOREEXTRA.
- Jika kolom dengan nama yang sama tetapi atribut berbeda ada di kedua tabel, perintah gagal. Like-named kolom harus memiliki atribut berikut yang sama:
  - Jenis data
  - Ukuran kolom
  - Pengkodean kompresi
  - Tidak nol
  - Urutkan gaya
  - Urutkan kolom kunci
  - Gaya distribusi
  - Kolom kunci distribusi

- Anda tidak dapat menambahkan kolom identitas. Jika tabel sumber dan tabel target memiliki kolom identitas, perintah gagal. Jika hanya tabel sumber yang memiliki kolom identitas, sertakan parameter IGNOREEXTRA sehingga kolom identitas diabaikan. Jika hanya tabel target yang memiliki kolom identitas, sertakan parameter FILLTARGET sehingga kolom identitas diisi sesuai dengan klausa IDENTITY yang ditentukan untuk tabel. Untuk informasi selengkapnya, lihat DEFAULT.
- Anda dapat menambahkan kolom identitas default dengan pernyataan ALTER TABLE APPEND. Untuk informasi selengkapnya, lihat CREATE TABLE.
- Operasi ALTER TABLE APPEND menyimpan kunci eksklusif saat dijalankan di Amazon Redshift streaming tampilan terwujud yang terhubung ke salah satu dari berikut ini:
  - Aliran data Amazon Kinesis
  - Topik Streaming Terkelola Amazon untuk Apache Kafka
  - Aliran eksternal yang didukung, seperti topik Confluent Cloud Kafka

  Untuk informasi selengkapnya, lihat Streaming konsumsi ke tampilan yang terwujud.

ALTER TABLE APPEND contoh

Misalkan organisasi Anda memelihara tabel, SALES_MONTHLY, untuk menangkap transaksi penjualan saat ini. Anda ingin memindahkan data dari tabel transaksi ke tabel SALES, setiap bulan.

Anda dapat menggunakan perintah INSERT INTO dan TRUNCATE berikut untuk menyelesaikan tugas.

```
insert into sales (select * from sales_monthly);
truncate sales_monthly;
```

Namun, Anda dapat melakukan operasi yang sama jauh lebih efisien dengan menggunakan perintah ALTER TABLE APPEND.

Pertama, kueri tabel katalog PG_TABLE_DEF sistem untuk memverifikasi bahwa kedua tabel memiliki kolom yang sama dengan atribut kolom yang identik.

```
select trim(tablename) as table, "column", trim(type) as type,
encoding, distkey, sortkey, "notnull"
from pg_table_def where tablename like 'sales%';
```

```
table      | column     | type                          | encoding | distkey | sortkey |
 notnull
-----------+------------+-------------------------------+----------+---------+---------
+--------
sales      | salesid    | integer                       | lzo      | false   |       0 |
 true
sales      | listid     | integer                       | none     | true    |       1 |
 true
sales      | sellerid   | integer                       | none     | false   |       2 |
 true
sales      | buyerid    | integer                       | lzo      | false   |       0 |
 true
sales      | eventid    | integer                       | mostly16 | false   |       0 |
 true
sales      | dateid     | smallint                      | lzo      | false   |       0 |
 true
sales      | qtysold    | smallint                      | mostly8  | false   |       0 |
 true
sales      | pricepaid  | numeric(8,2)                  | delta32k | false   |       0 |
 false
sales      | commission | numeric(8,2)                  | delta32k | false   |       0 |
 false
sales      | saletime   | timestamp without time zone | lzo      | false   |       0 |
 false
salesmonth | salesid    | integer                       | lzo      | false   |       0 |
 true
salesmonth | listid     | integer                       | none     | true    |       1 |
 true
salesmonth | sellerid   | integer                       | none     | false   |       2 |
 true
salesmonth | buyerid    | integer                       | lzo      | false   |       0 |
 true
salesmonth | eventid    | integer                       | mostly16 | false   |       0 |
 true
salesmonth | dateid     | smallint                      | lzo      | false   |       0 |
 true
salesmonth | qtysold    | smallint                      | mostly8  | false   |       0 |
 true
salesmonth | pricepaid  | numeric(8,2)                  | delta32k | false   |       0 |
 false
salesmonth | commission | numeric(8,2)                  | delta32k | false   |       0 |
 false
```

```
salesmonth | saletime   | timestamp without time zone | lzo      | false   |       0 |
 false
```

Selanjutnya, lihat ukuran masing-masing tabel.

```
select count(*) from sales_monthly;
 count
-------
  2000
(1 row)

select count(*) from sales;
 count
-------
 412,214
(1 row)
```

Sekarang jalankan perintah ALTER TABLE APPEND berikut.

```
alter table sales append from sales_monthly;
```

Lihatlah ukuran setiap tabel lagi. Tabel SALES_MONTHLY sekarang memiliki 0 baris, dan tabel SALES telah tumbuh 2000 baris.

```
select count(*) from sales_monthly;
 count
-------
     0
(1 row)

select count(*) from sales;
 count
-------
 414214
(1 row)
```

Jika tabel sumber memiliki lebih banyak kolom daripada tabel target, tentukan parameter IGNOREEXTRA. Contoh berikut menggunakan parameter IGNOREEXTRA untuk mengabaikan kolom tambahan dalam tabel SALES_LISTING saat menambahkan ke tabel SALES.

```
alter table sales append from sales_listing ignoreextra;
```

Jika tabel target memiliki lebih banyak kolom daripada tabel sumber, tentukan parameter FILLTARGET. Contoh berikut menggunakan parameter FILLTARGET untuk mengisi kolom dalam tabel SALES_REPORT yang tidak ada di tabel SALES_MONTH.

```
alter table sales_report append from sales_month filltarget;
```

Contoh berikut menunjukkan contoh bagaimana menggunakan ALTER TABLE APPEND dengan tampilan terwujud sebagai sumber.

```
ALTER TABLE target_tbl APPEND FROM my_streaming_materialized_view;
```

Tabel dan nama tampilan terwujud dalam contoh ini adalah sampel. Menambahkan dari tampilan terwujud hanya berfungsi dalam kasus di mana tampilan terwujud Anda dikonfigurasi untuk. Streaming konsumsi ke tampilan yang terwujud Ini memindahkan semua catatan dalam tampilan terwujud sumber ke tabel target dengan skema yang sama dengan tampilan terwujud dan membiarkan tampilan terwujud utuh. Ini adalah perilaku yang sama seperti ketika sumber data adalah tabel.

UBAH TEMPLAT

Mengubah definisi template yang ada. Gunakan perintah ini untuk mengganti nama template, mengubah pemilik template, menambah atau menghapus parameter dari definisi template, atau menetapkan nilai parameter.

Hak istimewa yang diperlukan

Untuk mengubah template, Anda harus memiliki salah satu dari berikut ini:

- Hak istimewa pengguna super
- Hak istimewa ALTER TEMPLATE dan hak istimewa PENGGUNAAN pada skema yang berisi template

Sintaksis

```
ALTER TEMPLATE [database_name.][schema_name.]template_name
{
RENAME TO new_name
| OWNER TO new_owner
```

```
| ADD  parameter [AS] [value]
| DROP parameter
| SET parameter TO value1 [, parameter2 TO value2 , ...]
};
```

Parameter

database_name

(Opsional) Nama database tempat template dibuat. Jika tidak ditentukan, database saat ini digunakan.

nama skema

(Opsional) Nama skema tempat template dibuat. Jika tidak ditentukan, template dicari di jalur pencarian saat ini.

nama_templat

Nama template yang akan diubah.

GANTI NAMA MENJADI

Klausa yang mengganti nama template.

nama_baru

Nama baru template. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

PEMILIK UNTUK

Klausul yang mengubah pemilik template.

pemilik baru

Pemilik baru template.

TAMBAH parameter [AS] [nilai]

Menambahkan parameter baru ke template.

- Untuk parameter kata kunci saja (seperti CSV atau GZIP), tentukan hanya nama parameter.
- Untuk parameter yang memerlukan nilai, tentukan nama parameter diikuti oleh nilai. Anda dapat secara opsional menyertakan AS antara parameter dan nilai.

Parameter DROP

Menghapus parameter yang ditentukan dari template. Tidak dapat menghapus beberapa parameter dengan satu perintah DROP.

ATUR parameter KE nil ai1 [, parameter2 KE nilai2,...]

Memperbarui nilai parameter template yang ada. Hanya digunakan untuk parameter yang sudah memiliki nilai. Beberapa parameter dapat diperbarui dalam satu perintah.

Contoh

Contoh berikut mengganti nama template test_template menjadi demo_template.

```
ALTER TEMPLATE test_template
RENAME TO demo_template;
```

Contoh berikut memberikan kepemilikan skema demo_template kepada pengguna bob.

```
ALTER TEMPLATE demo_template
OWNER TO bob;
```

Contoh berikut menambahkan parameter CSV ke template demo_template

```
ALTER TEMPLATE demo_template
ADD CSV;
```

Contoh berikut menambahkan parameter TIMEFORMAT 'auto' ke template demo_template

```
ALTER TEMPLATE demo_template
ADD TIMEFORMAT 'auto';
```

Contoh berikut menjatuhkan parameter ENCRYPTED dari template demo_template

```
ALTER TEMPLATE demo_template
DROP ENCRYPTED;
```

Contoh berikut menetapkan DELIMITER parameter ke '|' dan TIMEFORMAT parameter ke'epochsecs':

```
ALTER TEMPLATE demo_template
SET DELIMITER TO '|', TIMEFORMAT TO 'epochsecs';
```

ALTER USER

Mengubah pengguna database.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk ALTER USER:

- Superuser
- Pengguna dengan hak istimewa ALTER USER
- Pengguna saat ini yang ingin mengubah kata sandi mereka sendiri

Sintaksis

```
ALTER USER username [ WITH ] option [, ... ]

where option is

CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }
| PASSWORD { 'password' | 'md5hash' | 'sha256hash' | DISABLE }
[ VALID UNTIL 'expiration_date' ]
| RENAME TO new_name |
| CONNECTION LIMIT { limit | UNLIMITED }
| SESSION TIMEOUT limit | RESET SESSION TIMEOUT
| SET parameter { TO | = } { value | DEFAULT }
| RESET parameter
| EXTERNALID external_id
| NOLOGIN
| LOGIN PASSWORD { 'password' | 'md5hash' | 'sha256hash' }
```

Parameter

username

Nama pengguna.

DENGAN

Kata kunci opsional.

CREATEDB | NOCREATEDB

Opsi CREATEDB memungkinkan pengguna untuk membuat database baru. NOCREATEDB adalah default.

CREATEUSER | NOCREATEUSER

Opsi CREATEUSER membuat superuser dengan semua hak database, termasuk CREATE USER. Defaultnya adalah NOCREATEUSER. Untuk informasi selengkapnya, lihat superuser.

AKSES SYSLOG {DIBATASI | TIDAK DIBATASI}

Klausa yang menentukan tingkat akses yang dimiliki pengguna ke tabel dan tampilan sistem Amazon Redshift.

Pengguna biasa yang memiliki izin SYSLOG ACCESS RESTRICTED hanya dapat melihat baris yang dihasilkan oleh pengguna tersebut dalam tabel dan tampilan sistem yang terlihat pengguna. Defaultnya adalah DIBATASI.

Pengguna biasa yang memiliki izin SYSLOG ACCESS UNLIMITED dapat melihat semua baris dalam tabel dan tampilan sistem yang terlihat pengguna, termasuk baris yang dihasilkan oleh pengguna lain. UNRESTRICTED tidak memberikan akses pengguna biasa ke tabel yang terlihat oleh pengguna super. Hanya pengguna super yang dapat melihat tabel yang terlihat oleh pengguna super.

> **Note**
>
> Memberikan pengguna akses tak terbatas ke tabel sistem memberikan visibilitas pengguna terhadap data yang dihasilkan oleh pengguna lain. Misalnya, STL_QUERY dan STL_QUERYTEXT berisi teks lengkap pernyataan INSERT, UPDATE, dan DELETE, yang mungkin berisi data sensitif buatan pengguna.

Semua baris di SVV_TRANSACTIONS terlihat oleh semua pengguna.

Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

KATA SANDI {'kata sandi' | 'md5hash' | 'sha256hash' | NONAKTIFKAN}

Menetapkan kata sandi pengguna.

Secara default, pengguna dapat mengubah kata sandi mereka sendiri, kecuali kata sandi dinonaktifkan. Untuk menonaktifkan kata sandi pengguna, tentukan NONAKTIFKAN. Ketika kata sandi pengguna dinonaktifkan, kata sandi dihapus dari sistem dan pengguna dapat masuk hanya menggunakan kredenSIAL pengguna sementara AWS Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat Menggunakan otentikasi IAM untuk menghasilkan kredenSIAL pengguna database. Hanya pengguna super yang dapat mengaktifkan atau menonaktifkan kata sandi. Anda tidak dapat menonaktifkan kata sandi pengguna super. Untuk mengaktifkan kata sandi, jalankan ALTER USER dan tentukan kata sandi.

Untuk detail tentang menggunakan parameter PASSWORD, lihatBUAT PENGGUNA.

BERLAKU HINGGA 'expiration_date'

Menentukan bahwa kata sandi memiliki tanggal kedaluwarsa. Gunakan nilai `'infinity'` untuk menghindari tanggal kedaluwarsa. Tipe data yang valid untuk parameter ini adalah stempel waktu.

Hanya pengguna super yang dapat menggunakan parameter ini.

GANTI NAMA MENJADI

Mengganti nama pengguna.

nama_baru

Nama baru pengguna. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

> **Important**
>
> Saat Anda mengganti nama pengguna, Anda juga harus mengatur ulang kata sandi pengguna. Kata sandi reset tidak harus berbeda dari kata sandi sebelumnya. Nama pengguna digunakan sebagai bagian dari enkripsi kata sandi, jadi ketika pengguna berganti nama, kata sandi dihapus. Pengguna tidak akan dapat masuk sampai kata sandi diatur ulang. Contoh:
>
> ```
> alter user newuser password 'EXAMPLENewPassword11';
> ```

BATAS KONEKSI {limit | TIDAK TERBATAS}

Jumlah maksimum koneksi database yang diizinkan untuk dibuka oleh pengguna secara bersamaan. Batas tidak diberlakukan untuk pengguna super. Gunakan kata kunci UNLIMITED

untuk mengizinkan jumlah maksimum koneksi bersamaan. Batas jumlah koneksi untuk setiap database mungkin juga berlaku. Untuk informasi selengkapnya, lihat BUAT BASIS DATA. Defaultnya adalah UNLIMITED. Untuk melihat koneksi saat ini, kueri STV_SESSION tampilan sistem.

> **Note**
>
> Jika batas koneksi pengguna dan database berlaku, slot koneksi yang tidak digunakan harus tersedia yang berada dalam kedua batas saat pengguna mencoba menyambung.

Batas BATAS WAKTU SESI | ATUR ULANG BATAS WAKTU SESI

Waktu maksimum dalam detik sesi tetap tidak aktif atau idle. Kisarannya adalah 60 detik (satu menit) hingga 1.728.000 detik (20 hari). Jika tidak ada batas waktu sesi yang ditetapkan untuk pengguna, pengaturan cluster berlaku. Untuk informasi selengkapnya, lihat Kuota dan batasan di Amazon Redshift di Panduan Manajemen Amazon Redshift.

Saat Anda mengatur batas waktu sesi, itu hanya diterapkan pada sesi baru.

Untuk melihat informasi tentang sesi pengguna aktif, termasuk waktu mulai, nama pengguna, dan batas waktu sesi, kueri tampilan STV_SESSION sistem. Untuk melihat informasi tentang riwayat sesi pengguna, kueri STL_SESSIONS tampilan. Untuk mengambil informasi tentang pengguna database, termasuk nilai batas waktu sesi, kueri tampilan. SVL_USER_INFO

SET

Menetapkan parameter konfigurasi ke nilai default baru untuk semua sesi yang dijalankan oleh pengguna yang ditentukan.

MENGATUR ULANG

Mengatur ulang parameter konfigurasi ke nilai default asli untuk pengguna yang ditentukan.

Parameter

Nama parameter yang akan diatur atau diatur ulang.

nilai

Nilai baru dari parameter.

DEFAULT

Menetapkan parameter konfigurasi ke nilai default untuk semua sesi yang dijalankan oleh pengguna yang ditentukan.

EXTERNALID_id eksternal

Pengidentifikasi untuk pengguna, yang terkait dengan penyedia identitas. Pengguna harus menonaktifkan kata sandi mereka. Untuk informasi selengkapnya, lihat Federasi penyedia identitas asli (IdP) untuk Amazon Red shift.

NOLOGIN

Mengunci pengguna yang ditentukan sehingga mereka tidak dapat terhubung ke database. Amazon Redshift menolak login untuk pengguna yang terkunci. Saat Anda mengunci pengguna, Amazon Redshift tidak mengakhiri sesi pengguna yang ada. Untuk membuka kunci pengguna, gunakan LOGIN PASSWORD.

Tentukan NOLOGIN dengan sendirinya, tanpa KATA SANDI. Jika Anda menyertakan KATA SANDI dengan NOLOGIN, Amazon Redshift mengubah kata sandi pengguna tetapi tetap mengunci pengguna, yang membuat pengguna terkunci dengan kata sandi baru. Untuk mengunci pengguna tanpa mengubah kata sandi mereka, gunakan NOLOGIN saja.

Penguncian pengguna hanya berlaku untuk pengguna basis data berbasis kata sandi. Pengguna federasi tidak dapat dikunci. Ini termasuk pengguna yang mengotentikasi melalui AWS Identity and Access Management (IAM) atau. AWS IAM Identity Center Anda juga tidak dapat mengunci pengguna admin. Untuk mengunci pengguna, Anda harus menjadi superuser atau pengguna dengan hak istimewa ALTER USER. Opsi ini hanya tersedia ketika keamanan penguncian pengguna diaktifkan untuk cluster. Untuk informasi selengkapnya, lihat TAMPILKAN PENGUNCIAN PENGGUNA dan upayan_login_gagal maksimal.

KATA SANDI MASUK {'kata sandi' | 'md5hash' | 'sha 256hash'}

Membuka kunci pengguna yang ditentukan dan menetapkan kata sandi baru. Anda harus menentukan KATA SANDI saat Anda menggunakan LOGIN. Saat Anda membuka kunci pengguna, Amazon Redshift mengatur ulang penghitung login gagal pengguna tersebut.

Untuk membuka kunci pengguna, Anda harus menjadi superuser atau pengguna dengan hak istimewa ALTER USER. Seperti halnya NOLOGIN, ini hanya berlaku untuk pengguna berbasis kata sandi. Opsi ini hanya tersedia ketika keamanan penguncian pengguna diaktifkan untuk cluster.

Catatan penggunaan

- Mencoba mengubah rdsdb — Anda tidak dapat mengubah nama pengguna. `rdsdb`
- Membuat kata sandi yang tidak dikenal — Saat menggunakan otentikasi AWS Identity and Access Management (IAM) untuk membuat kredenSIAL pengguna database, Anda mungkin ingin membuat superuser yang hanya dapat masuk menggunakan kredenSIAL sementara. Anda tidak dapat menonaktifkan kata sandi superuser, tetapi Anda dapat membuat kata sandi yang tidak dikenal menggunakan string hash MD5 yang dibuat secara acak.

  ```
  alter user iam_superuser password 'md51234567890123456780123456789012';
  ```

- Mengatur search_path — Ketika Anda mengatur search_path parameter dengan perintah ALTER USER, modifikasi akan berlaku pada login pengguna berikutnya yang ditentukan. Jika Anda ingin mengubah nilai search_path untuk pengguna dan sesi saat ini, gunakan perintah SET.
- Mengatur zona waktu — Saat Anda menggunakan SET TIMEZONE dengan perintah ALTER USER, modifikasi akan berlaku pada login pengguna berikutnya yang ditentukan.
- Bekerja dengan penyembunyian data dinamis dan kebijakan keamanan tingkat baris — Saat cluster yang disediakan atau namespace tanpa server memiliki penyembunyian data dinamis atau kebijakan keamanan tingkat baris, perintah berikut diblokir untuk pengguna biasa:

  ```
  ALTER <current_user> SET enable_case_sensitive_super_attribute/
  enable_case_sensitive_identifier/downcase_delimited_identifier
  ```

  Hanya pengguna super dan pengguna dengan hak istimewa ALTER USER yang dapat mengatur opsi konfigurasi ini. Untuk informasi tentang keamanan tingkat baris, lihat. Row-level keamanan Untuk informasi tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

Contoh

Contoh berikut memberi pengguna ADMIN hak istimewa untuk membuat database:

```
alter user admin createdb;
```

Contoh berikut menetapkan kata sandi pengguna ADMIN ke `adminPass9` dan menetapkan tanggal dan waktu kedaluwarsa untuk kata sandi:

```
alter user admin password 'adminPass9'
```

```
valid until '2017-12-31 23:59';
```

Contoh berikut mengganti nama pengguna ADMIN menjadi SYSADMIN:

```
alter user admin rename to sysadmin;
```

Contoh berikut memperbarui batas waktu sesi idle-session untuk pengguna menjadi 300 detik.

```
ALTER USER dbuser SESSION TIMEOUT 300;
```

Mengatur ulang batas waktu sesi idle-session pengguna. Saat Anda mengatur ulang, pengaturan cluster berlaku. Anda harus menjadi superuser database untuk menjalankan perintah ini. Untuk informasi selengkapnya, lihat Kuota dan batasan di Amazon Redshift dalam Panduan Manajemen Amazon Redshift.

```
ALTER USER dbuser RESET SESSION TIMEOUT;
```

Contoh berikut memperbarui ID eksternal untuk pengguna bernamabob. Namespace adalahmyco_aad. Jika namespace tidak terkait dengan penyedia identitas terdaftar, itu mengakibatkan kesalahan.

```
ALTER USER myco_aad:bob EXTERNALID "ABC123" PASSWORD DISABLE;
```

Contoh berikut menetapkan zona waktu untuk semua sesi yang dijalankan oleh pengguna database tertentu. Ini mengubah zona waktu untuk sesi berikutnya, tetapi tidak untuk sesi saat ini.

```
ALTER USER odie SET TIMEZONE TO 'Europe/Zurich';
```

Contoh berikut mengunci pengguna `data_analyst` sehingga mereka tidak dapat terhubung ke database:

```
ALTER USER data_analyst NOLOGIN;
```

Contoh berikut membuka kunci pengguna `data_analyst` dan menetapkan kata sandi baru. Membuka kunci pengguna mengatur ulang penghitung login gagal mereka:

```
ALTER USER data_analyst LOGIN PASSWORD 'NewStr0ngP@ssword';
```

Contoh berikut menetapkan jumlah maksimum koneksi database yang bob diizinkan untuk dibuka oleh pengguna.

```
ALTER USER bob CONNECTION LIMIT 10;
```

MENGANALISA

Memperbarui statistik tabel untuk digunakan oleh perencana kueri.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk ANALYZE:

- Superuser
- Pengguna dengan hak istimewa ANALYZE
- Pemilik hubungan
- Pemilik database kepada siapa tabel dibagikan

Sintaksis

```
ANALYZE [ VERBOSE ]
[ [ table_name [ ( column_name [, ...] ) ] ]
[ PREDICATE COLUMNS | ALL  COLUMNS ]
```

Parameter

BERTELE-TELE

Klausa yang mengembalikan pesan informasi kemajuan tentang operasi ANALYZE. Opsi ini berguna ketika Anda tidak menentukan tabel.

table_name

Anda dapat menganalisis tabel tertentu, termasuk tabel sementara. Anda dapat memenuhi syarat tabel dengan nama skemanya. Anda dapat secara opsional menentukan table_name untuk menganalisis satu tabel. Anda tidak dapat menentukan lebih dari satu table_name dengan pernyataan ANALYZE table _name tunggal. Jika Anda tidak menentukan nilai table_name, semua tabel dalam database yang terhubung saat ini dianalisis, termasuk tabel persisten

dalam katalog sistem. Amazon Redshift melewatkan analisis tabel jika persentase baris yang telah berubah sejak ANALYZE terakhir lebih rendah dari ambang analisis. Untuk informasi selengkapnya, lihat Analisis ambang batas.

Anda tidak perlu menganalisis tabel sistem Amazon Redshift (tabel STL dan STV).

nama_kolom

Jika Anda menentukan table_name, Anda juga dapat menentukan satu atau lebih kolom dalam tabel (sebagai daftar yang dipisahkan kolom dalam tanda kurung). Jika daftar kolom ditentukan, hanya kolom yang terdaftar yang dianalisis.

KOLOM PREDIKAT | SEMUA KOLOM

Klausul yang menunjukkan apakah ANALYZE harus menyertakan hanya kolom predikat. Tentukan KOLOM PREDIKAT untuk menganalisis hanya kolom yang telah digunakan sebagai predikat dalam kueri sebelumnya atau kemungkinan kandidat untuk digunakan sebagai predikat. Tentukan SEMUA KOLOM untuk menganalisis semua kolom. Defaultnya adalah ALL COLUMNS.

Sebuah kolom disertakan dalam himpunan kolom predikat jika salah satu dari berikut ini benar:

- Kolom telah digunakan dalam kueri sebagai bagian dari filter, kondisi gabungan, atau grup demi klausa.
- Kolom adalah kunci distribusi.
- Kolom adalah bagian dari kunci pengurutan.

Jika tidak ada kolom yang ditandai sebagai kolom predikat, misalnya karena tabel belum ditanyakan, semua kolom dianalisis bahkan ketika KOLOM PREDIKAT ditentukan. Ketika ini terjadi, Amazon Redshift mungkin merespons dengan pesan seperti Tidak ada kolom predikat yang ditemukan untuk ""*table-name*. Menganalisis semua kolom. Untuk informasi selengkapnya tentang kolom predikat, lihatMenganalisis tabel.

Catatan penggunaan

Amazon Redshift secara otomatis menjalankan ANALYZE pada tabel yang Anda buat dengan perintah berikut:

- CREATE TABLE AS
- BUAT TABEL TEMP SEBAGAI

- PILIH MENJADI

Anda tidak dapat menganalisis tabel eksternal.

Anda tidak perlu menjalankan perintah ANALYZE pada tabel ini ketika mereka pertama kali dibuat. Jika Anda memodifikasinya, Anda harus menganalisisnya dengan cara yang sama seperti tabel lainnya.

Analisis ambang batas

Untuk mengurangi waktu pemrosesan dan meningkatkan kinerja sistem secara keseluruhan, Amazon Redshift melewatkan ANALYZE untuk tabel jika persentase baris yang telah berubah sejak perintah ANALYZE terakhir dijalankan lebih rendah dari ambang analisis yang ditentukan oleh parameter. analisis_ambang_persen Secara default, `analyze_threshold_percent` adalah 10. `analyze_threshold_percent`Untuk mengubah sesi saat ini, jalankan SET perintah. Contoh berikut berubah `analyze_threshold_percent` menjadi 20 persen.

```
set analyze_threshold_percent to 20;
```

Untuk menganalisis tabel ketika hanya sejumlah kecil baris telah berubah, atur `analyze_threshold_percent` ke jumlah kecil yang sewenang-wenang. Misalnya, jika Anda mengatur `analyze_threshold_percent` ke 0,01, maka tabel dengan 100.000.000 baris tidak dilewati jika setidaknya 10.000 baris telah berubah.

```
set analyze_threshold_percent to 0.01;
```

Jika ANALYZE melewatkan tabel karena tidak memenuhi ambang analisis, Amazon Redshift mengembalikan pesan berikut.

```
ANALYZE SKIP
```

Untuk menganalisis semua tabel meskipun tidak ada baris yang berubah, atur `analyze_threshold_percent` ke 0.

Untuk melihat hasil operasi ANALYSIS, kueri tabel STL_ANALISIS sistem.

Untuk informasi selengkapnya tentang menganalisis tabel, lihatMenganalisis tabel.

Contoh

Analisis semua tabel dalam database TICKIT dan mengembalikan informasi kemajuan.

```
analyze verbose;
```

Analisis tabel LISTING saja.

```
analyze listing;
```

Analisis kolom VENUEID dan VENUENAME di tabel VENUE.

```
analyze venue(venueid, venuename);
```

Analisis hanya kolom predikat dalam tabel VENUE.

```
analyze venue predicate columns;
```

MENGANALISIS KOMPRESI

Melakukan analisis kompresi dan menghasilkan laporan dengan pengkodean kompresi yang disarankan untuk tabel yang dianalisis. Untuk setiap kolom, laporan mencakup perkiraan potensi pengurangan ruang disk dibandingkan dengan pengkodean RAW.

Sintaksis

```
ANALYZE COMPRESSION
[ [ table_name ]
[ ( column_name [, ...] ) ] ]
[COMPROWS numrows]
```

Parameter

table_name

Anda dapat menganalisis kompresi untuk tabel tertentu, termasuk tabel sementara. Anda dapat memenuhi syarat tabel dengan nama skemanya. Anda dapat secara opsional menentukan table_name untuk menganalisis satu tabel. Jika Anda tidak menentukan table_name, semua tabel dalam database yang terhubung saat ini akan dianalisis. Anda tidak dapat menentukan lebih dari satu table_name dengan pernyataan COMPRESSION ANALYZE tunggal.

nama_kolom

Jika Anda menentukan table_name, Anda juga dapat menentukan satu atau lebih kolom dalam tabel (sebagai daftar yang dipisahkan kolom dalam tanda kurung).

KERUSUHAN

Jumlah baris yang akan digunakan sebagai ukuran sampel untuk analisis kompresi. Analisis dijalankan pada baris dari setiap irisan data. Misalnya, jika Anda menentukan COMPROWS 1000000 (1.000.000) dan sistem berisi 4 irisan total, tidak lebih dari 250.000 baris per irisan dibaca dan dianalisis. Jika COMPROWS tidak ditentukan, ukuran sampel default menjadi 100.000 per irisan. Nilai COMPROWS yang lebih rendah dari default 100.000 baris per irisan secara otomatis ditingkatkan ke nilai default. Namun, analisis kompresi tidak menghasilkan rekomendasi jika jumlah data dalam tabel tidak mencukupi untuk menghasilkan sampel yang bermakna. Jika angka COMPROWS lebih besar dari jumlah baris dalam tabel, perintah ANALYZE COMPRESSION masih melanjutkan dan menjalankan analisis kompresi terhadap semua baris yang tersedia. Menggunakan COMPROWS menghasilkan kesalahan jika tabel tidak ditentukan.

mati rasa

Jumlah baris yang akan digunakan sebagai ukuran sampel untuk analisis kompresi. Rentang yang diterima untuk angka adalah angka antara 1000 dan 1000000000 (1.000.000.000).

Catatan penggunaan

ANALYZE COMPRESSION memperoleh kunci tabel eksklusif, yang mencegah pembacaan dan penulisan bersamaan terhadap tabel. Hanya jalankan perintah ANALYZE COMPRESSION saat tabel tidak aktif.

Jalankan ANALYZE COMPRESSION untuk mendapatkan rekomendasi untuk skema pengkodean kolom, berdasarkan sampel isi tabel. ANALYZE COMPRESSION adalah alat penasihat dan tidak mengubah pengkodean kolom tabel. Anda dapat menerapkan pengkodean yang disarankan dengan membuat ulang tabel atau dengan membuat tabel baru dengan skema yang sama. Membuat ulang tabel yang tidak terkompresi dengan skema pengkodean yang sesuai dapat secara signifikan mengurangi jejak pada disk. Pendekatan ini menghemat ruang disk dan meningkatkan kinerja kueri untuk I/O-bound beban kerja.

ANALISIS COMPRESSION melewatkan fase analisis aktual dan langsung mengembalikan jenis pengkodean asli pada kolom apa pun yang ditetapkan sebagai SORTKEY. Ini dilakukan karena

pemindaian terbatas rentang mungkin berkinerja buruk ketika kolom SORTKEY dikompresi jauh lebih tinggi daripada kolom lainnya.

Contoh

Contoh berikut menunjukkan pengkodean dan perkiraan pengurangan persen untuk kolom dalam tabel LISTING saja:

```
analyze compression listing;

  Table  |     Column     | Encoding | Est_reduction_pct
---------+----------------+----------+------------------
 listing | listid         | az64     | 40.96
 listing | sellerid       | az64     | 46.92
 listing | eventid        | az64     | 53.37
 listing | dateid         | raw      | 0.00
 listing | numtickets     | az64     | 65.66
 listing | priceperticket | az64     | 72.94
 listing | totalprice     | az64     | 68.05
 listing | listtime       | az64     | 49.74
```

Contoh berikut menganalisis kolom QTYSOLD, COMMISSION, dan SALETIME dalam tabel PENJUALAN.

```
analyze compression sales(qtysold, commission, saletime);

 Table |   Column   | Encoding | Est_reduction_pct
-------+------------+----------+------------------
 sales | salesid    | N/A      | 0.00
 sales | listid     | N/A      | 0.00
 sales | sellerid   | N/A      | 0.00
 sales | buyerid    | N/A      | 0.00
 sales | eventid    | N/A      | 0.00
 sales | dateid     | N/A      | 0.00
 sales | qtysold    | az64     | 83.06
 sales | pricepaid  | N/A      | 0.00
 sales | commission | az64     | 71.85
 sales | saletime   | az64     | 49.63
```

LAMPIRKAN KEBIJAKAN MASKING

Melampirkan kebijakan penyembunyian data dinamis yang ada ke kolom. Untuk informasi selengkapnya tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat melampirkan kebijakan masking.

Sintaksis

```
ATTACH MASKING POLICY
{
  policy_name ON relation_name
  | database_name.policy_name ON database_name.schema_name.relation_name
}
( { output_column_names | output_path } )
[ USING ( { input_column_names | input_path } ) ]
TO { user_name | ROLE role_name | PUBLIC }
[ PRIORITY priority ];
```

Parameter

nama_kebijakan

Nama kebijakan masking yang akan dilampirkan.

database_name

Nama database tempat kebijakan dan relasi dibuat. Kebijakan dan hubungan harus berada di database yang sama. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

schema_name

Nama skema milik relasi.

nama relasi

Nama relasi untuk melampirkan kebijakan masking.

nama_kolom keluaran

Nama-nama kolom yang akan diterapkan kebijakan masking.

jalur keluaran

Jalur lengkap objek SUPER yang akan diterapkan kebijakan masking, termasuk nama kolom. Misalnya, untuk relasi dengan kolom tipe SUPER bernama`person`, output_path mungkin. `person.name.first_name`

nama_kolom masukan

Nama-nama kolom yang akan diambil kebijakan masking sebagai masukan. Parameter ini bersifat opsional. Jika tidak ditentukan, kebijakan masking menggunakan output_column_names sebagai input.

masukan_jalur

Jalur lengkap objek SUPER yang akan diambil kebijakan masking sebagai masukan. Parameter ini bersifat opsional. Jika tidak ditentukan, kebijakan masking menggunakan output_path untuk input.

nama pengguna

Nama pengguna yang akan dilampirkan kebijakan masking. Anda tidak dapat melampirkan dua kebijakan ke kombinasi pengguna dan kolom atau peran dan kolom yang sama. Anda dapat melampirkan kebijakan ke pengguna dan kebijakan lain ke peran pengguna. Dalam hal ini, kebijakan dengan prioritas yang lebih tinggi berlaku.

Anda hanya dapat mengatur salah satu dari user_name, role_name, dan PUBLIC dalam satu perintah ATTACH MASKING POLICY.

role_name

Nama peran yang akan dilampirkan kebijakan masking. Anda tidak dapat melampirkan dua kebijakan ke column/role pasangan yang sama. Anda dapat melampirkan kebijakan ke pengguna dan kebijakan lain ke peran pengguna. Dalam hal ini, kebijakan dengan prioritas yang lebih tinggi berlaku.

Anda hanya dapat mengatur salah satu dari user_name, role_name, dan PUBLIC dalam satu perintah ATTACH MASKING POLICY.

PUBLIK

Melampirkan kebijakan masking ke semua pengguna yang mengakses tabel. Anda harus memberikan prioritas yang lebih tinggi pada kebijakan penyamaran tertentu column/user atau column/role pasangan tertentu daripada kebijakan PUBLIK agar dapat diterapkan.

Anda hanya dapat mengatur salah satu dari user_name, role_name, dan PUBLIC dalam satu perintah ATTACH MASKING POLICY.

prioritas

Prioritas kebijakan masking. Ketika beberapa kebijakan masking diterapkan pada kueri pengguna tertentu, kebijakan prioritas tertinggi berlaku.

Anda tidak dapat melampirkan dua kebijakan berbeda ke kolom yang sama dengan prioritas yang sama, bahkan jika kedua kebijakan tersebut dilampirkan ke pengguna atau peran yang berbeda. Anda dapat melampirkan kebijakan yang sama beberapa kali ke kumpulan tabel, kolom keluaran, kolom input, dan parameter prioritas yang sama, selama pengguna atau peran yang dilampirkan kebijakan berbeda setiap kali.

Anda tidak dapat menerapkan kebijakan ke kolom dengan prioritas yang sama dengan kebijakan lain yang dilampirkan ke kolom tersebut, meskipun kebijakan tersebut untuk peran yang berbeda. Ini adalah bidang opsional. Jika Anda tidak menentukan prioritas, kebijakan masking secara default akan melampirkan dengan prioritas 0.

Untuk penggunaan ATTACH MASKING POLICY di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

LAMPIRKAN KEBIJAKAN RLS

Melampirkan kebijakan keamanan tingkat baris pada tabel ke satu atau lebih pengguna atau peran.

Pengguna super dan pengguna atau peran yang memiliki `sys:secadmin` peran dapat melampirkan kebijakan.

Sintaksis

```
ATTACH RLS POLICY
{
  policy_name ON [TABLE] table_name [, ...]
  | database_name.policy_name ON [TABLE] database_name.schema_name.table_name [, ...]
}
TO { user_name | ROLE role_name | PUBLIC } [, ...]
```

Parameter

nama_kebijakan

Nama kebijakan .

database_name

Nama database tempat kebijakan dan relasi dibuat. Kebijakan dan hubungan harus berada di database yang sama. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

schema_name

Nama skema milik relasi.

table_name

Hubungan yang dilampirkan pada kebijakan keamanan tingkat baris.

UNTUK {user_name | ROLE name _name | PUBLIK} [,...]

Menentukan apakah kebijakan dilampirkan ke satu atau lebih pengguna atau peran tertentu.

Untuk penggunaan ATTACH RLS POLICY di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

Catatan penggunaan

Saat bekerja dengan pernyataan ATTACH RLS POLICY, perhatikan hal berikut:

- Tabel yang dilampirkan harus memiliki semua kolom yang tercantum dalam klausa WITH dari pernyataan pembuatan kebijakan.
- Amazon Redshift RLS mendukung melampirkan kebijakan RLS ke objek berikut:
  - Tabel
  - Tampilan
  - Late-binding pandangan
  - Tampilan terwujud
- Amazon Redshift RLS tidak mendukung melampirkan kebijakan RLS ke objek berikut:
  - Tabel katalog
  - Cross-database hubungan

  - Tabel eksternal
  - Tabel sementara
  - Tabel pencarian kebijakan
  - Tabel dasar tampilan yang terwujud
- Kebijakan RLS yang dilampirkan ke pengguna super atau pengguna dengan `sys:secadmin` izin diabaikan.

Contoh

Contoh berikut melampirkan kebijakan RLS ke tabel dan kombinasi peran yang ditentukan. Kebijakan RLS berlaku untuk setiap pengguna dengan peran `analyst` atau `dbadmin` saat mereka mengakses tabel tickit_category_redshift.

```
ATTACH RLS POLICY policy_concerts ON tickit_category_redshift TO ROLE analyst, ROLE
 dbadmin;
```

MULAI

Memulai transaksi. Identik dengan MULAI TRANSAKSI.

Transaksi adalah unit kerja logis tunggal, apakah itu terdiri dari satu perintah atau beberapa perintah. Secara umum, semua perintah dalam transaksi berjalan pada snapshot dari database yang waktu mulanya ditentukan oleh nilai yang ditetapkan untuk parameter konfigurasi `transaction_snapshot_begin` sistem.

Secara default, operasi Amazon Redshift individual (kueri, pernyataan DDL, beban) secara otomatis diikat ke database. Jika Anda ingin menangguhkan komit untuk operasi sampai pekerjaan berikutnya selesai, Anda perlu membuka transaksi dengan pernyataan BEGIN, lalu jalankan perintah yang diperlukan, lalu tutup transaksi dengan AKHIR pernyataan COMMIT atau. Jika perlu, Anda dapat menggunakan ROLLBACK pernyataan untuk menghentikan transaksi yang sedang berlangsung. Pengecualian untuk perilaku ini adalah MEMOTONG perintah, yang melakukan transaksi di mana ia dijalankan dan tidak dapat diputar kembali.

Sintaksis

```
BEGIN [ WORK | TRANSACTION ] [ ISOLATION LEVEL option ] [ READ WRITE | READ ONLY ]
```

```
START TRANSACTION [ ISOLATION LEVEL option ] [ READ WRITE | READ ONLY ]

Where option is

SERIALIZABLE
| READ UNCOMMITTED
| READ COMMITTED
| REPEATABLE READ

Note: READ UNCOMMITTED, READ COMMITTED, and REPEATABLE READ have no
operational impact and map to SERIALIZABLE in Amazon Redshift. You can see database
 isolation levels on your cluster
by querying the stv_db_isolation_level table.
```

Parameter

BEKERJA

Kata kunci opsional.

TRANSAKSI

Kata kunci opsional; KERJA dan TRANSAKSI adalah sinonim.

TINGKAT ISOLASI DAPAT DISERIALKAN

Isolasi yang dapat diserialkan didukung secara default, sehingga perilaku transaksi sama apakah sintaks ini disertakan dalam pernyataan atau tidak. Untuk informasi selengkapnya, lihat Mengelola operasi penulisan bersamaan. Tidak ada tingkat isolasi lain yang didukung.

> **Note**
>
> Standar SQL mendefinisikan empat tingkat isolasi transaksi untuk mencegah pembacaan kotor (di mana transaksi membaca data yang ditulis oleh transaksi tidak berkomitmen bersamaan), pembacaan yang tidak dapat diulang (di mana transaksi membaca ulang data yang dibaca sebelumnya dan menemukan bahwa data diubah oleh transaksi lain yang dilakukan sejak pembacaan awal), dan pembacaan phantom (di mana transaksi menjalankan kembali kueri, mengembalikan satu set baris yang memenuhi kondisi pencarian, dan kemudian menemukan bahwa kumpulan baris telah berubah karena dari transaksi lain yang baru-baru ini dilakukan):
>
> - Baca tidak terikat: Pembacaan kotor, pembacaan yang tidak dapat diulang, dan pembacaan hantu dimungkinkan.

- Baca berkomitmen: Pembacaan yang tidak dapat diulang dan pembacaan hantu dimungkinkan.
- Pembacaan yang dapat diulang: Pembacaan hantu dimungkinkan.
- Serialisasi: Mencegah pembacaan kotor, pembacaan yang tidak dapat diulang, dan pembacaan hantu.

Meskipun Anda dapat menggunakan salah satu dari empat tingkat isolasi transaksi, Amazon Redshift memproses semua tingkat isolasi sebagai dapat diserialkan.

BACA TULIS

Memberikan izin baca dan tulis transaksi.

BACA SAJA

Memberikan izin baca-saja transaksi.

Contoh

Contoh berikut memulai blok transaksi yang dapat diserialisasi:

```
begin;
```

Contoh berikut memulai blok transaksi dengan tingkat isolasi yang dapat diserialisasi dan izin baca dan tulis:

```
begin read write;
```

PANGGILAN

Menjalankan prosedur tersimpan. Perintah CALL harus menyertakan nama prosedur dan nilai argumen input. Anda harus memanggil prosedur tersimpan dengan menggunakan pernyataan CALL.

> **Note**
>
> CALL tidak dapat menjadi bagian dari kueri reguler.

Sintaksis

```
CALL sp_name ( [ argument ] [, ...] )
```

Parameter

sp_nama

Nama prosedur yang akan dijalankan.

argumen

Nilai argumen input. Parameter ini juga bisa menjadi nama fungsi, misalnya`pg_last_query_id()`. Anda tidak dapat menggunakan query sebagai argumen CALL.

Catatan penggunaan

Prosedur tersimpan Amazon Redshift mendukung panggilan bersarang dan rekursif, seperti yang dijelaskan di bawah ini. Selain itu, pastikan dukungan pengemudi Anda mutakhir, juga dijelaskan berikut.

Topik

- Panggilan bersarang
- Dukungan pengemudi

Panggilan bersarang

Prosedur tersimpan Amazon Redshift mendukung panggilan bersarang dan rekursif. Jumlah maksimum tingkat bersarang yang diizinkan adalah 16. Panggilan bersarang dapat merangkum logika bisnis ke dalam prosedur yang lebih kecil, yang dapat dibagikan oleh beberapa pemanggil.

Jika Anda memanggil prosedur bersarang yang memiliki parameter keluaran, prosedur dalam harus menentukan argumen INOUT. Dalam hal ini, prosedur bagian dalam dilewatkan dalam variabel nonkonstan. Argumen OUT tidak diperbolehkan. Perilaku ini terjadi karena variabel diperlukan untuk menahan output dari panggilan dalam.

Hubungan antara prosedur dalam dan luar dicatat di `from_sp_call` kolomSVL_STORED_PROC_CALL.

Contoh berikut menunjukkan meneruskan variabel ke panggilan prosedur bersarang melalui argumen INOUT.

```
CREATE OR REPLACE PROCEDURE inner_proc(INOUT a int, b int, INOUT c int) LANGUAGE
 plpgsql
AS $$
BEGIN
  a := b * a;
  c := b * c;
END;
$$;

CREATE OR REPLACE PROCEDURE outer_proc(multiplier int) LANGUAGE plpgsql
AS $$
DECLARE
  x int := 3;
  y int := 4;
BEGIN
  DROP TABLE IF EXISTS test_tbl;
  CREATE TEMP TABLE test_tbl(a int, b varchar(256));
  CALL inner_proc(x, multiplier, y);
  insert into test_tbl values (x, y::varchar);
END;
$$;

CALL outer_proc(5);

SELECT * from test_tbl;
 a  | b
----+----
 15 | 20
(1 row)
```

Dukungan pengemudi

Sebaiknya upgrade driver Java Database Connectivity (JDBC) dan Open Database Connectivity (ODBC) ke versi terbaru yang memiliki dukungan untuk prosedur tersimpan Amazon Redshift.

Anda mungkin dapat menggunakan driver yang ada jika alat klien Anda menggunakan operasi API driver yang melewati pernyataan CALL ke server. Parameter keluaran, jika ada, dikembalikan sebagai hasil set satu baris.

Versi terbaru dari driver Amazon Redshift JDBC dan ODBC memiliki dukungan metadata untuk penemuan prosedur tersimpan. Mereka juga memiliki `CallableStatement` dukungan untuk aplikasi Java kustom. Untuk informasi selengkapnya tentang driver, lihat Menghubungkan ke Cluster Amazon Redshift Menggunakan Alat Klien SQL di Panduan Manajemen Amazon Redshift.

Contoh berikut menunjukkan cara menggunakan operasi API yang berbeda dari driver JDBC untuk panggilan prosedur tersimpan.

```
void statement_example(Connection conn) throws SQLException {
  statement.execute("CALL sp_statement_example(1)");
}

void prepared_statement_example(Connection conn) throws SQLException {
  String sql = "CALL sp_prepared_statement_example(42, 84)";
  PreparedStatement pstmt = conn.prepareStatement(sql);
  pstmt.execute();
}

void callable_statement_example(Connection conn) throws SQLException {
  CallableStatement cstmt = conn.prepareCall("CALL sp_create_out_in(?,?)");
  cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
  cstmt.setInt(2, 42);
  cstmt.executeQuery();
  Integer out_value = cstmt.getInt(1);
}
```

Contoh

Contoh berikut memanggil nama prosedur`test_spl`.

```
call test_sp1(3,'book');
INFO:  Table "tmp_tbl" does not exist and will be skipped
INFO:  min_val = 3, f2 = book
```

Contoh berikut memanggil nama prosedur`test_spl2`.

```
call test_sp2(2,'2019');

         f2          | column2
---------------------+---------
 2019+2019+2019+2019 | 2
```

```
(1 row)
```

CANCEL (BATALKAN)

Membatalkan kueri database yang sedang berjalan.

Perintah CANCEL memerlukan ID proses atau ID sesi dari kueri yang sedang berjalan dan menampilkan pesan konfirmasi untuk memverifikasi bahwa kueri dibatalkan.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CANCEL:

- Superuser membatalkan kueri mereka sendiri
- Superuser membatalkan kueri pengguna
- Pengguna dengan hak istimewa CANCEL membatalkan kueri pengguna
- Pengguna membatalkan kueri mereka sendiri

Sintaksis

```
CANCEL process_id [ 'message' ]
```

Parameter

Proses_id

Untuk membatalkan kueri yang berjalan di cluster Amazon Redshift, gunakan `pid` (ID Proses) STV_TERBARU yang sesuai dengan kueri yang ingin Anda batalkan.

Untuk membatalkan kueri yang berjalan di grup kerja Amazon Redshift Serverless, gunakan `session_id` dari SYS_QUERY_HISTORY yang sesuai dengan kueri yang ingin Anda batalkan.

'pesan'

Pesan konfirmasi opsional yang ditampilkan saat pembatalan kueri selesai. Jika Anda tidak menentukan pesan, Amazon Redshift menampilkan pesan default sebagai verifikasi. Anda harus melampirkan pesan dalam tanda kutip tunggal.

Catatan penggunaan

Anda tidak dapat membatalkan kueri dengan menentukan ID kueri; Anda harus menentukan ID proses kueri (PID) atau ID sesi. Anda hanya dapat membatalkan kueri yang saat ini dijalankan oleh pengguna Anda. Superuser dapat membatalkan semua kueri.

Jika kueri dalam beberapa sesi menahan kunci pada tabel yang sama, Anda dapat menggunakan PG_TERMINATE_BACKEND fungsi untuk mengakhiri salah satu sesi. Melakukan hal ini memaksa transaksi yang sedang berjalan dalam sesi yang dihentikan untuk melepaskan semua kunci dan mengembalikan transaksi. Untuk melihat kunci yang saat ini dipegang, kueri tabel STV_LOCKS sistem.

Setelah kejadian internal tertentu, Amazon Redshift mungkin memulai ulang sesi aktif dan menetapkan PID baru. Jika PID telah berubah, Anda mungkin menerima pesan kesalahan berikut.

```
Session <PID> does not exist. The session PID might have changed. Check the
 stl_restarted_sessions system table for details.
```

Untuk menemukan PID baru, kueri tabel STL_RESTARTED_SESSIONS sistem dan filter pada `oldpid` kolom.

```
select oldpid, newpid from stl_restarted_sessions where oldpid = 1234;
```

Contoh

Untuk membatalkan kueri yang sedang berjalan di cluster Amazon Redshift, pertama-tama ambil ID proses untuk kueri yang ingin Anda batalkan. Untuk menentukan ID proses untuk semua kueri yang sedang berjalan, ketik perintah berikut:

```
select pid, starttime, duration,
trim(user_name) as user,
trim (query) as querytxt
from stv_recents
where status = 'Running';

pid |         starttime          | duration |   user   |    querytxt
-----+----------------------------+----------+----------+-----------------
802 | 2008-10-14 09:19:03.550885 |      132 | dwuser | select
venuename from venue where venuestate='FL', where venuecity not in
('Miami' , 'Orlando');
```

```
834 | 2008-10-14 08:33:49.473585 |  1250414 | dwuser | select *
from listing;
964 | 2008-10-14 08:30:43.290527 |   326179 | dwuser | select
sellerid from sales where qtysold in (8, 10);
```

Periksa teks kueri untuk menentukan id proses (PID) mana yang sesuai dengan kueri yang ingin Anda batalkan.

Ketik perintah berikut untuk menggunakan PID 802 untuk membatalkan kueri itu:

```
cancel 802;
```

Sesi tempat kueri berjalan menampilkan pesan berikut:

```
ERROR:  Query (168) cancelled on user's request
```

di 168 mana ID kueri (bukan ID proses yang digunakan untuk membatalkan kueri).

Atau, Anda dapat menentukan pesan konfirmasi khusus untuk ditampilkan alih-alih pesan default. Untuk menentukan pesan khusus, sertakan pesan Anda dalam tanda kutip tunggal di akhir perintah CANCEL:

```
cancel 802 'Long-running query';
```

Sesi tempat kueri berjalan menampilkan pesan berikut:

```
ERROR:  Long-running query
```

TUTUP

(Opsional) Menutup semua sumber daya bebas yang terkait dengan kursor terbuka. COMMIT,AKHIR, dan ROLLBACK secara otomatis menutup kursor, sehingga tidak perlu menggunakan perintah CLOSE untuk menutup kursor secara eksplisit.

Untuk informasi lebih lanjut, lihatMENYATAKAN,AMBIL.

Sintaksis

```
CLOSE cursor
```

Parameter

kursor

Nama kursor untuk menutup.

Contoh TUTUP

Perintah berikut menutup kursor dan melakukan komit, yang mengakhiri transaksi:

```
close movie_cursor;
commit;
```

KOMENTAR

Membuat atau mengubah komentar tentang objek database.

Sintaksis

```
COMMENT ON
{
TABLE object_name |
COLUMN object_name.column_name |
CONSTRAINT constraint_name ON table_name |
DATABASE object_name |
VIEW object_name
}
IS 'text' | NULL
```

Parameter

nama_objek

Nama objek database yang dikomentari. Anda dapat menambahkan komentar ke objek berikut:

- TABEL
- COLUMN (juga mengambil column_name).
- CONSTRAINT (juga mengambil constra int_name dan table_name).
- BASIS DATA

- MELIHAT
- SCHEMA

IS 'teks' | NULL

Teks komentar yang ingin Anda tambahkan atau ganti untuk objek yang ditentukan. String teks adalah tipe data TEXT. Lampirkan komentar dalam tanda kutip tunggal. Atur nilai ke NULL untuk menghapus teks komentar.

nama_kolom

Nama kolom yang dikomentari. Parameter KOLOM. Mengikuti tabel yang ditentukan di`object_name`.

nama_kendala

Nama kendala yang sedang dikomentari. Parameter kendala.

table_name

Nama tabel yang berisi batasan. Parameter kendala.

Catatan penggunaan

Anda harus menjadi superuser atau pemilik objek database untuk menambahkan atau memperbarui komentar.

Komentar pada database hanya dapat diterapkan ke database saat ini. Pesan peringatan ditampilkan jika Anda mencoba mengomentari database yang berbeda. Peringatan yang sama ditampilkan untuk komentar pada database yang tidak ada.

Komentar pada tabel eksternal, kolom eksternal, dan kolom tampilan pengikatan akhir tidak didukung.

Contoh

Contoh berikut menambahkan komentar ke tabel PENJUALAN.

```
COMMENT ON TABLE sales IS 'This table stores tickets sales data';
```

Contoh berikut menampilkan komentar pada tabel PENJUALAN.

```
select obj_description('public.sales'::regclass);
```

```
obj_description
------------------------------------
This table stores tickets sales data
```

Contoh berikut menghapus komentar dari tabel SALES.

```
COMMENT ON TABLE sales IS NULL;
```

Contoh berikut menambahkan komentar ke kolom EVENTID dari tabel SALES.

```
COMMENT ON COLUMN sales.eventid IS 'Foreign-key reference to the EVENT table.';
```

Contoh berikut menampilkan komentar pada kolom EVENTID (kolom nomor 5) dari tabel SALES.

```
select col_description( 'public.sales'::regclass, 5::integer );

col_description
----------------------------------------
Foreign-key reference to the EVENT table.
```

Contoh berikut menambahkan komentar deskriptif ke tabel EVENT.

```
comment on table event is 'Contains listings of individual events.';
```

Untuk melihat komentar, kueri katalog sistem PG_DESCRIPTION. Contoh berikut mengembalikan deskripsi untuk tabel EVENT.

```
select * from pg_catalog.pg_description
where objoid =
(select oid from pg_class where relname = 'event'
and relnamespace =
(select oid from pg_catalog.pg_namespace where nspname = 'public') );

objoid | classoid | objsubid | description
-------+----------+----------+----------------------------------------
116658 |     1259 |        0 | Contains listings of individual events.
```

COMMIT

Komitmen transaksi saat ini ke database. Perintah ini membuat pembaruan database dari transaksi permanen.

Sintaksis

```
COMMIT [ WORK | TRANSACTION ]
```

Parameter

BEKERJA

Kata kunci opsional. Kata kunci ini tidak didukung dalam prosedur tersimpan.

TRANSAKSI

Kata kunci opsional. KERJA dan TRANSAKSI adalah sinonim. Keduanya tidak didukung dalam prosedur tersimpan.

Untuk informasi tentang menggunakan COMMIT dalam prosedur tersimpan, lihatMengelola transaksi.

Contoh

Masing-masing contoh berikut mengkomit transaksi saat ini ke database:

```
commit;
```

```
commit work;
```

```
commit transaction;
```

MENYALIN

Client-side enkripsi untuk perintah COPY dan UNLOAD tidak akan lagi terbuka untuk pelanggan baru mulai 30 April 2025. Jika Anda menggunakan enkripsi sisi klien dengan perintah COPY dan UNLOAD dalam 12 bulan sebelum 30 April 2025, Anda dapat terus menggunakan enkripsi sisi

klien dengan perintah COPY atau UNLOAD hingga 30 April 2026. Setelah 30 April 2026, Anda tidak akan dapat menggunakan enkripsi sisi klien untuk COPY dan UNLOAD. Kami menyarankan Anda beralih menggunakan enkripsi sisi server untuk COPY dan UNLOAD sesegera mungkin. Jika Anda sudah menggunakan enkripsi sisi server untuk COPY dan UNLOAD, tidak ada perubahan dan Anda dapat terus menggunakannya tanpa mengubah kueri Anda. Untuk informasi selengkapnya tentang enkripsi untuk COPY dan UNLOAD, lihat parameter ENCRYPTED di bawah ini.

Memuat data ke dalam tabel dari file data atau dari tabel Amazon DynamoDB. File dapat ditemukan di bucket Amazon Simple Storage Service (Amazon S3), cluster Amazon EMR, atau host jarak jauh yang diakses menggunakan koneksi Secure Shell (SSH).

> **Note**
>
> Tabel eksternal Amazon Redshift Spectrum bersifat read-only. Anda tidak dapat menyalin ke tabel eksternal.

Perintah COPY menambahkan data input sebagai baris tambahan ke tabel.

Ukuran maksimum satu baris input dari sumber apa pun adalah 4 MB.

Topik

- Izin yang diperlukan
- Sintaks COPY
- Parameter yang diperlukan
- Parameter opsional
- Catatan penggunaan dan sumber daya tambahan untuk perintah COPY
- Contoh perintah COPY
- SALIN PEKERJAAN
- COPY dengan TEMPLATE
- Referensi parameter COPY
- Catatan penggunaan
- Contoh COPY

Izin yang diperlukan

Untuk menggunakan perintah COPY, Anda harus memiliki INSERT hak istimewa untuk tabel Amazon Redshift.

Sintaks COPY

```
COPY table-name
[ column-list ]
FROM data_source
authorization
[ [ FORMAT ] [ AS ] data_format ]
[ parameter [ argument ] [, ... ] ]
```

Anda dapat melakukan operasi COPY dengan sedikitnya tiga parameter: nama tabel, sumber data, dan otorisasi untuk mengakses data.

Amazon Redshift memperluas fungsionalitas perintah COPY sehingga Anda dapat memuat data dalam beberapa format data dari beberapa sumber data, mengontrol akses untuk memuat data, mengelola transformasi data, dan mengelola operasi pemuatan.

Bagian berikut menyajikan parameter perintah COPY yang diperlukan, mengelompokkan parameter opsional berdasarkan fungsi. Mereka juga menjelaskan setiap parameter dan menjelaskan bagaimana berbagai opsi bekerja bersama. Anda dapat langsung menuju deskripsi parameter dengan menggunakan daftar parameter abjad.

Parameter yang diperlukan

Perintah COPY membutuhkan tiga elemen:

- Table Name
- Data Source
- Authorization

Perintah COPY paling sederhana menggunakan format berikut.

```
COPY table-name
FROM data-source
authorization;
```

Contoh berikut membuat tabel bernama CATDEMO, dan kemudian memuat tabel dengan data sampel dari file data di Amazon S3 bernama`category_pipe.txt`.

```
create table catdemo(catid smallint, catgroup varchar(10), catname varchar(10), catdesc
 varchar(50));
```

Dalam contoh berikut, sumber data untuk perintah COPY adalah file data yang `category_pipe.txt` dinamai dalam `tickit` folder bucket Amazon S3 bernama`redshift-downloads`. Perintah COPY diotorisasi untuk mengakses bucket Amazon S3 melalui peran AWS Identity and Access Management (IAM). Jika cluster Anda memiliki peran IAM yang sudah ada dengan izin untuk mengakses Amazon S3 terlampir, Anda dapat mengganti Nama Sumber Daya Amazon (ARN) peran Anda dalam perintah COPY berikut dan menjalankannya.

```
copy catdemo
from 's3://redshift-downloads/tickit/category_pipe.txt'
iam_role 'arn:aws:iam::<aws-account-id>:role/<role-name>'
region 'us-east-1';
```

Untuk petunjuk lengkap tentang cara menggunakan perintah SALIN untuk memuat data sampel, termasuk instruksi untuk memuat data dari Wil AWS ayah lain, lihat Memuat Data Sampel dari Amazon S3 di Panduan Memulai Amazon Redshift.

nama-tabel

Nama tabel target untuk perintah COPY. Tabel harus sudah ada dalam basis data. Tabel bisa bersifat sementara atau persisten. Perintah COPY menambahkan data input baru ke baris yang ada dalam tabel.

DARI sumber data

Lokasi data sumber yang akan dimuat ke tabel target. File manifes dapat ditentukan dengan beberapa sumber data.

Repositori data yang paling umum digunakan adalah bucket Amazon S3. Anda juga dapat memuat dari file data yang terletak di cluster Amazon EMR, instans Amazon EC2, atau host jarak jauh yang dapat diakses cluster Anda menggunakan koneksi SSH, atau Anda dapat memuat langsung dari tabel DynamoDB.

- SALIN dari Amazon S3
- SALIN dari Amazon EMR

- SALIN dari host jarak jauh (SSH)
- SALIN dari Amazon DynamoDB

Otorisasi

Klausa yang menunjukkan metode yang digunakan cluster Anda untuk otentikasi dan otorisasi untuk mengakses AWS sumber daya lain. Perintah COPY memerlukan otorisasi untuk mengakses data di AWS sumber daya lain, termasuk di Amazon S3, Amazon EMR, Amazon DynamoDB, dan Amazon EC2. Anda dapat memberikan otorisasi tersebut dengan mereferensikan peran IAM yang dilampirkan ke cluster Anda atau dengan memberikan ID kunci akses dan kunci akses rahasia untuk pengguna IAM.

- Parameter otorisasi
- Role-based kontrol akses
- Key-based kontrol akses

Parameter opsional

Anda dapat secara opsional menentukan bagaimana COPY memetakan data bidang ke kolom dalam tabel target, menentukan atribut data sumber untuk mengaktifkan perintah COPY untuk membaca dan mengurai data sumber dengan benar, dan mengelola operasi yang dilakukan perintah COPY selama proses pemuatan.

- Opsi pemetaan kolom
- Parameter format data
- Parameter konversi data
- Operasi beban data

Pemetaan kolom

Secara default, COPY menyisipkan nilai bidang ke kolom tabel target dalam urutan yang sama seperti bidang yang terjadi dalam file data. Jika urutan kolom default tidak berfungsi, Anda dapat menentukan daftar kolom atau menggunakan ekspresi JSONPath untuk memetakan bidang data sumber ke kolom target.

- Column List
- JSONPaths File

Parameter format data

Anda dapat memuat data dari file teks dalam format lebar tetap, dibatasi karakter, nilai yang dipisahkan koma (CSV), atau JSON, atau dari file Avro.

Secara default, perintah COPY mengharapkan data sumber berada dalam file teks yang dibatasi karakter UTF-8 . Pembatas default adalah karakter pipa (|). Jika data sumber dalam format lain, gunakan parameter berikut untuk menentukan format data.

- FORMAT
- CSV
- DELIMITER
- FIXEDWIDTH
- SHAPEFILE
- AVRO
- JSON format for COPY
- ENCRYPTED
- BZIP2
- GZIP
- LZOP
- PARQUET
- ORC
- ZSTD

Parameter konversi data

Saat memuat tabel, COPY mencoba untuk secara implisit mengonversi string dalam data sumber ke tipe data kolom target. Jika Anda perlu menentukan konversi yang berbeda dari perilaku default, atau jika konversi default menghasilkan kesalahan, Anda dapat mengelola konversi data dengan menentukan parameter berikut.

- ACCEPTANYDATE
- ACCEPTINVCHARS
- BLANKSASNULL
- DATEFORMAT

- EMPTYASNULL
- ENCODING
- ESCAPE
- EXPLICIT_IDS
- FILLRECORD
- IGNOREBLANKLINES
- IGNOREHEADER
- NULL AS
- REMOVEQUOTES
- ROUNDEC
- TIMEFORMAT
- TRIMBLANKS
- TRUNCATECOLUMNS

Operasi beban data

Kelola perilaku default operasi beban untuk pemecahan masalah atau untuk mengurangi waktu muat dengan menentukan parameter berikut.

- COMPROWS
- COMPUPDATE
- IGNOREALLERRORS
- MAXERROR
- NOLOAD
- STATUPDATE

Catatan penggunaan dan sumber daya tambahan untuk perintah COPY

Untuk informasi selengkapnya tentang cara menggunakan perintah SALIN, lihat topik berikut:

- Catatan penggunaan
- Tutorial: Memuat data dari Amazon S3
- Praktik terbaik Amazon Redshift untuk memuat data

- Memuat tabel dengan perintah COPY
  - Memuat data dari Amazon S3
  - Memuat data dari Amazon EMR
  - Memuat data dari host jarak jauh
  - Memuat data dari tabel Amazon DynamoDB
- Memecahkan masalah beban data

Contoh perintah COPY

Untuk contoh selengkapnya yang menunjukkan cara MENYALIN dari berbagai sumber, dalam format yang berbeda, dan dengan opsi SALIN yang berbeda, lihatContoh COPY.

SALIN PEKERJAAN

Untuk informasi tentang menggunakan perintah ini, lihatBuat integrasi acara S3 untuk menyalin file secara otomatis dari bucket Amazon S3.

Mengelola perintah COPY yang memuat data ke dalam tabel. Perintah COPY JOB adalah perpanjangan dari perintah COPY dan mengotomatiskan pemuatan data dari bucket Amazon S3. Saat Anda membuat pekerjaan COPY, Amazon Redshift mendeteksi kapan file Amazon S3 baru dibuat di jalur yang ditentukan, lalu memuatnya secara otomatis tanpa campur tangan Anda. Parameter yang sama yang digunakan dalam perintah COPY asli digunakan saat memuat data. Amazon Redshift melacak file yang dimuat (berdasarkan nama file) untuk memverifikasi bahwa file tersebut dimuat hanya satu kali.

> **Note**
>
> Untuk informasi tentang perintah COPY, termasuk penggunaan, parameter, dan izin, lihatMENYALIN.

Izin yang diperlukan

Untuk menggunakan perintah COPY JOB, Anda harus memiliki salah satu izin berikut selain semua izin yang diperlukan untuk menggunakan COPY:

- Superuser

- Semua hal berikut:
  - Izin cakupan CREATE, ALTER, atau DROP yang relevan untuk COPY JOBS dalam database yang ingin Anda SALIN.
  - Izin PENGGUNAAN untuk skema yang ingin Anda SALIN, atau izin cakupan PENGGUNAAN untuk skema dalam database yang ingin Anda SALIN.
  - Izin INSERT untuk tabel yang ingin Anda SALIN, atau INSERT izin cakupan untuk tabel dalam skema atau database yang ingin Anda SALIN.

Peran IAM yang ditentukan dengan perintah COPY harus memiliki izin untuk mengakses data yang akan dimuat. Untuk informasi selengkapnya, lihat Izin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY.

Sintaks

Buat pekerjaan penyalinan. Parameter perintah COPY disimpan dengan pekerjaan salin.

Anda tidak dapat menjalankan COPY JOB CREATE dalam lingkup blok transaksi.

```
COPY copy-command JOB CREATE job-name
[AUTO ON | OFF]
```

Ubah konfigurasi pekerjaan penyalinan.

```
COPY JOB ALTER job-name
[AUTO ON | OFF]
```

Jalankan pekerjaan penyalinan. Parameter perintah COPY yang disimpan digunakan.

```
COPY JOB RUN job-name
```

Buat daftar semua pekerjaan salinan.

```
COPY JOB LIST
```

Tampilkan detail pekerjaan penyalinan.

```
COPY JOB SHOW job-name
```

Hapus pekerjaan penyalinan.

Anda tidak dapat menjalankan COPY JOB DROP dalam lingkup blok transaksi.

```
COPY JOB DROP job-name
```

Parameter

salin-perintah

Perintah COPY yang memuat data dari Amazon S3 ke Amazon Redshift. Klausa berisi parameter COPY yang menentukan bucket Amazon S3, tabel target, peran IAM, dan parameter lain yang digunakan saat memuat data. Semua parameter perintah COPY untuk pemuatan data Amazon S3 didukung kecuali:

- COPY JOB tidak menyerap file yang sudah ada sebelumnya di folder yang ditunjuk oleh perintah COPY. Hanya file yang dibuat setelah stempel waktu pembuatan COPY JOB yang dicerna.
- Anda tidak dapat menentukan perintah COPY dengan opsi MAXERROR atau IGNOREALLERRORS.
- Anda tidak dapat menentukan file manifes. COPY JOB memerlukan lokasi Amazon S3 yang ditunjuk untuk memantau file yang baru dibuat.
- Anda tidak dapat menentukan perintah COPY dengan jenis otorisasi seperti kunci Akses dan Rahasia. Hanya perintah COPY yang menggunakan `IAM_ROLE` parameter untuk otorisasi yang didukung. Untuk informasi selengkapnya, lihat Parameter otorisasi.
- COPY JOB tidak mendukung peran IAM default yang terkait dengan cluster. Anda harus menentukan `IAM_ROLE` dalam perintah COPY.

Untuk informasi selengkapnya, lihat SALIN dari Amazon S3.

nama-pekerjaan

Nama pekerjaan yang digunakan untuk mereferensikan pekerjaan COPY. Nama pekerjaan tidak dapat berisi tanda hubung (-).

[OTOMATIS HIDUP | MATI]

Klausul yang menunjukkan apakah data Amazon S3 dimuat secara otomatis ke dalam tabel Amazon Redshift.

- Saat`ON`, Amazon Redshift memantau jalur sumber Amazon S3 untuk file yang baru dibuat, dan jika ditemukan, perintah COPY dijalankan dengan parameter COPY dalam definisi pekerjaan. Ini adalah opsi default.
- Saat`OFF`, Amazon Redshift tidak menjalankan COPY JOB secara otomatis.

Catatan penggunaan

Opsi perintah COPY tidak divalidasi hingga waktu berjalan. Misalnya, sumber data Amazon S3 yang tidak valid `IAM_ROLE` atau menghasilkan kesalahan waktu proses saat COPY JOB dimulai.

Jika cluster dijeda, COPY JOBS tidak dijalankan.

Untuk menanyakan file perintah COPY yang dimuat dan memuat kesalahan, lihatSTL_LOAD_COMMIT,STL_LOAD_ERROR,STL_LOADERROR_DETAIL. Untuk informasi selengkapnya, lihat Memverifikasi bahwa data dimuat dengan benar.

COPY JOBS tidak didukung pada database Zero-ETL karena beroperasi dalam mode read-only.

Contoh

Contoh berikut menunjukkan pembuatan COPY JOB untuk memuat data dari bucket Amazon S3.

```
COPY public.target_table
FROM 's3://amzn-s3-demo-bucket/staging-folder'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyLoadRoleName'
JOB CREATE my_copy_job_name
AUTO ON;
```

COPY dengan TEMPLATE

Anda dapat menggunakan templat Redshift dengan perintah COPY untuk menyederhanakan sintaks perintah dan menjaga konsistensi di seluruh operasi pemuatan data. Alih-alih menentukan parameter pemformatan yang sama berulang kali, Anda mendefinisikannya sekali dalam template dan mereferensikan template dalam perintah COPY Anda. Saat Anda menggunakan template, perintah COPY menggabungkan parameter dari template dengan parameter apa pun yang ditentukan langsung dalam perintah. Jika parameter yang sama muncul di template dan perintah, parameter perintah diutamakan. Untuk informasi selengkapnya, lihat BUAT TEMPLAT.

Template untuk perintah COPY dapat dibuat dengan:

- Parameter format data
- Parameter kompresi file
- Parameter konversi data
- Operasi beban data

Untuk daftar lengkap parameter yang didukung, lihat MENYALIN perintah.

Izin yang diperlukan

Untuk menggunakan template dalam perintah COPY, Anda harus memiliki:

- Semua izin yang diperlukan untuk menjalankan perintah COPY (lihatIzin yang diperlukan)
- Salah satu izin template berikut:
  - Hak istimewa pengguna super
  - Hak istimewa PENGGUNAAN pada template dan hak istimewa PENGGUNAAN pada skema yang berisi template

Sintaksis

```
COPY target_table FROM 's3://...'
authorization
[ option, ...]
USING TEMPLATE [database_name.][schema_name.]template_name;
```

Parameter

database_name

(Opsional) Nama database tempat template ada. Jika tidak ditentukan, database saat ini digunakan.

nama skema

(Opsional) Nama skema tempat template ada. Jika tidak ditentukan, template dicari di jalur pencarian saat ini.

nama_templat

Nama template yang akan digunakan dalam COPY.

Catatan penggunaan

- Command-specific parameter (sumber, tujuan, otorisasi) masih harus ditentukan dalam perintah COPY.
- Template tidak dapat berisi spesifikasi file manifes untuk perintah COPY.

Contoh

Contoh berikut menunjukkan cara membuat template dan menggunakannya dalam perintah COPY:

```
CREATE TEMPLATE public.test_template FOR COPY AS
CSV DELIMITER '|' IGNOREHEADER 1 MAXERROR 100;

COPY public.target_table
FROM 's3://amzn-s3-demo-bucket/staging-folder'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyLoadRoleName'
USING TEMPLATE public.test_template;
```

Ketika parameter ada di template dan perintah, parameter perintah diutamakan. Dalam contoh ini, jika template `public.test_template` berisi `DELIMITER '|'` tetapi perintah COPY menentukan`DELIMITER ','`, pembatas koma (`,`) dari perintah akan digunakan sebagai pengganti pembatas pipa (`|`) dari template.

```
COPY public.target_table
FROM 's3://amzn-s3-demo-bucket/staging-folder'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyLoadRoleName'
DELIMITER ','
USING TEMPLATE public.test_template;
```

Referensi parameter COPY

COPY memiliki banyak parameter yang dapat digunakan dalam banyak situasi. Namun, tidak semua parameter didukung dalam setiap situasi. Misalnya, untuk memuat dari file ORC atau PARQUET ada sejumlah parameter yang didukung. Untuk informasi selengkapnya, lihat SALIN dari format data kolumnar.

Topik

- Sumber data
- Parameter otorisasi

- Opsi pemetaan kolom
- Parameter format data
- Parameter kompresi file
- Parameter konversi data
- Operasi beban data
- Daftar parameter abjad

Sumber data

Anda dapat memuat data dari file teks di bucket Amazon S3, di cluster Amazon EMR, atau pada host jarak jauh yang dapat diakses cluster Anda menggunakan koneksi SSH. Anda juga dapat memuat data langsung dari tabel DynamoDB.

Ukuran maksimum satu baris input dari sumber apa pun adalah 4 MB.

Untuk mengekspor data dari tabel ke sekumpulan file di Amazon S3, gunakan MEMBONGKAR perintah.

Topik

- SALIN dari Amazon S3
- SALIN dari Amazon EMR
- SALIN dari host jarak jauh (SSH)
- SALIN dari Amazon DynamoDB

SALIN dari Amazon S3

Untuk memuat data dari file yang terletak di satu atau beberapa bucket S3, gunakan klausa FROM untuk menunjukkan bagaimana COPY menemukan file di Amazon S3. Anda dapat menyediakan jalur objek ke file data sebagai bagian dari klausa FROM, atau Anda dapat memberikan lokasi file manifes yang berisi daftar jalur objek Amazon S3. COPY dari Amazon S3 menggunakan koneksi HTTPS. Pastikan bahwa rentang IP S3 ditambahkan ke daftar izin Anda. Untuk mempelajari selengkapnya tentang rentang IP S3 yang diperlukan, lihat Isolasi jaringan.

> **Important**
>
> Jika bucket Amazon S3 yang menyimpan file data tidak berada di Wil AWS ayah yang sama dengan cluster Anda, Anda harus menggunakan REGION parameter untuk menentukan Wilayah tempat data berada.

Topik

- Sintaksis
- Contoh
- Parameter opsional
- Parameter yang tidak didukung

Sintaksis

```
FROM { 's3://objectpath' | 's3://manifest_file' }
authorization
| MANIFEST
| ENCRYPTED
| REGION [AS] 'aws-region'
| optional-parameters
```

Contoh

Contoh berikut menggunakan jalur objek untuk memuat data dari Amazon S3.

```
copy customer
from 's3://amzn-s3-demo-bucket/customer'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Contoh berikut menggunakan file manifes untuk memuat data dari Amazon S3.

```
copy customer
from 's3://amzn-s3-demo-bucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

Parameter

FROM

Sumber data yang akan dimuat. Untuk informasi selengkapnya tentang pengkodean file Amazon S3, lihatParameter konversi data.

's3://salin_dari_s3_object path '

Menentukan jalur ke objek Amazon S3 yang berisi data—misalnya,. `'s3://amzn-s3-demo-bucket/custdata.txt'` Parameter s3://copy_from_s3_objectpath dapat mereferensikan satu file atau sekumpulan objek atau folder yang memiliki awalan kunci yang sama. Misalnya, nama `custdata.txt` adalah awalan kunci yang mengacu pada sejumlah file fisik:`custdata.txt`,`custdata.txt.1`,`custdata.txt.2`,`custdata.txt.bak`, dan sebagainya. Awalan kunci juga dapat mereferensikan sejumlah folder. Misalnya, `'s3://amzn-s3-demo-bucket/custfolder'` mengacu pada folder`custfolder`,`custfolder_1`,`custfolder_2`, dan sebagainya. Jika awalan kunci mereferensikan beberapa folder, semua file dalam folder dimuat. Jika awalan kunci cocok dengan file serta folder, seperti`custfolder.log`, COPY mencoba memuat file juga. Jika awalan kunci mungkin mengakibatkan COPY mencoba memuat file yang tidak diinginkan, gunakan file manifes. Untuk informasi lebih lanjut, lihatcopy_from_s3_manifest_file, berikut.

> **Important**
>
> Jika bucket S3 yang menyimpan file data tidak berada di Wil AWS ayah yang sama dengan cluster Anda, Anda harus menggunakan REGION parameter untuk menentukan Wilayah tempat data berada.

Untuk informasi selengkapnya, lihat Memuat data dari Amazon S3.

's3://salin_dari_s3_manifest_ file'

Menentukan kunci objek Amazon S3 untuk file manifes yang mencantumkan file data yang akan dimuat. Argumen 's3://copy_from_s3_manifest_file' harus secara ekspl isit mereferensikan satu file tunggal—misalnya,. `'s3://amzn-s3-demo-bucket/manifest.txt'` Itu tidak dapat mereferensikan awalan kunci.

Manifes adalah file teks dalam format JSON yang mencantumkan URL setiap file yang akan dimuat dari Amazon S3. URL menyertakan nama bucket dan jalur objek lengkap untuk file. File yang ditentukan dalam manifes dapat berada di bucket yang berbeda, tetapi semua bucket harus

berada di Wilayah yang sama AWS dengan cluster Amazon Redshift. Jika file terdaftar dua kali, file dimuat dua kali. Contoh berikut menunjukkan JSON untuk manifes yang memuat tiga file.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata.1","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket1/custdata.2","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket2/custdata.1","mandatory":false}
  ]
}
```

Karakter tanda kutip ganda diperlukan, dan harus berupa tanda kutip sederhana (0x22), bukan tanda kutip miring atau “pintar”. Setiap entri dalam manifes dapat secara opsional menyertakan `mandatory` flag. Jika `mandatory` disetel ke`true`, COPY berakhir jika tidak menemukan file untuk entri itu; jika tidak, COPY akan berlanjut. Nilai default-nya `mandatory` is `false`.

Saat memuat dari file data dalam format ORC atau Parquet, `meta` bidang diperlukan, seperti yang ditunjukkan pada contoh berikut.

```
{
   "entries":[
      {
         "url":"s3://amzn-s3-demo-bucket1/orc/2013-10-04-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      },
      {
         "url":"s3://amzn-s3-demo-bucket2/orc/2013-10-05-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      }
   ]
}
```

File manifes tidak boleh dienkripsi atau dikompresi, bahkan jika opsi ENCRYPTED, GZIP, LZOP, BZIP2, atau ZSTD ditentukan. COPY mengembalikan kesalahan jika file manifes yang ditentukan tidak ditemukan atau file manifes tidak terbentuk dengan benar.

Jika file manifes digunakan, parameter MANIFEST harus ditentukan dengan perintah COPY. Jika parameter MANIFEST tidak ditentukan, COPY mengasumsikan bahwa file yang ditentukan dengan FROM adalah file data.

Untuk informasi selengkapnya, lihat Memuat data dari Amazon S3.

otorisasi

Perintah COPY memerlukan otorisasi untuk mengakses data di AWS sumber daya lain, termasuk di Amazon S3, Amazon EMR, Amazon DynamoDB, dan Amazon EC2. Anda dapat memberikan otorisasi tersebut dengan mereferensikan peran AWS Identity and Access Management (IAM) yang dilampirkan ke cluster Anda (kontrol akses berbasis peran) atau dengan memberikan kredentif akses untuk pengguna (kontrol akses berbasis kunci). Untuk meningkatkan keamanan dan fleksibilitas, sebaiknya gunakan kontrol akses berbasis peran IAM. Untuk informasi selengkapnya, lihat Parameter otorisasi.

MANIFESTASI

Menentukan bahwa manifes digunakan untuk mengidentifikasi file data yang akan dimuat dari Amazon S3. Jika parameter MANIFEST digunakan, COPY memuat data dari file yang tercantum dalam manifes yang direferensikan oleh 's3://copy_from_s3_manifest_file'. Jika file manifes tidak ditemukan, atau tidak dibentuk dengan benar, COPY gagal. Untuk informasi selengkapnya, lihat Menggunakan manifes untuk menentukan file data.

TERENKRIPSI

Klausa yang menentukan bahwa file input di Amazon S3 dienkripsi menggunakan enkripsi sisi klien dengan kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Memuat file data terenkripsi dari Amazon S3. Jangan tentukan ENKRIPSI jika file input dienkripsi menggunakan enkripsi sisi server Amazon S3 (SSE-KMS atau SSE-S3). COPY membaca file terenkripsi sisi server secara otomatis.

Jika Anda menentukan parameter ENCRYPTED, Anda juga harus menentukan MASTER_SYMMETRIC_KEY parameter atau memasukkan **`master_symmetric_key`** nilai dalam the section called “Menggunakan KREDENSIAL” string.

Jika file terenkripsi dalam format terkompresi, tambahkan parameter GZIP, LZOP, BZIP2, atau ZSTD.

File manifes dan file JSONPaths tidak boleh dienkripsi, bahkan jika opsi ENCRYPTED ditentukan.

MASTER_SYMMETRIC_KEY 'root_key'

Kunci simetris root yang digunakan untuk mengenkripsi file data di Amazon S3. Jika MASTER_SYMMETRIC_KEY ditentukan, ENCRYPTED parameter juga harus ditentukan. MASTER_SYMMETRIC_KEY tidak dapat digunakan dengan parameter CREDENTIALS. Untuk informasi selengkapnya, lihat Memuat file data terenkripsi dari Amazon S3.

Jika file terenkripsi dalam format terkompresi, tambahkan parameter GZIP, LZOP, BZIP2, atau ZSTD.

WILAYAH [AS] 'aws- wilayah'

Menentukan AWS Wilayah tempat data sumber berada. REGION diperlukan untuk COPY dari bucket Amazon S3 atau tabel DynamoDB jika AWS sumber daya yang berisi data tidak berada di Wilayah yang sama dengan cluster Amazon Redshift.

Nilai untuk aws_region harus cocok dengan Wilayah yang tercantum dalam tabel wilayah Amazon Redshift dan titik akhir.

Jika parameter REGION ditentukan, semua sumber daya, termasuk file manifes atau beberapa bucket Amazon S3, harus berada di Wilayah yang ditentukan.

> **Note**
>
> Mentransfer data di seluruh Wilayah dikenakan biaya tambahan terhadap bucket Amazon S3 atau tabel DynamoDB yang berisi data. Untuk informasi selengkapnya tentang harga, lihat Transfer Data KELUAR Dari Amazon S3 Ke Wil AWS ayah Lain di halaman Harga Amazon S3 dan Transfer Data OUT di halaman Harga Amazon DynamoDB.

Secara default, COPY mengasumsikan bahwa data berada di Wilayah yang sama dengan cluster Amazon Redshift.

Parameter opsional

Anda dapat secara opsional menentukan parameter berikut dengan COPY dari Amazon S3:

- Opsi pemetaan kolom
- Parameter format data
- Parameter konversi data

- Operasi beban data

Parameter yang tidak didukung

Anda tidak dapat menggunakan parameter berikut dengan COPY dari Amazon S3:

- SSH
- RASIO BACA

SALIN dari Amazon EMR

Anda dapat menggunakan perintah COPY untuk memuat data secara paralel dari cluster Amazon EMR yang dikonfigurasi untuk menulis file teks ke Hadoop Distributed File System (HDFS) cluster dalam bentuk file dengan lebar tetap, file yang dibatasi karakter, file CSV, file, atau file Avro. JSON-formatted

Topik

- Sintaksis
- Contoh
- Parameter
- Parameter yang didukung
- Parameter yang tidak didukung

Sintaksis

```
FROM 'emr://emr_cluster_id/hdfs_filepath'
authorization
[ optional_parameters ]
```

Contoh

Contoh berikut memuat data dari cluster Amazon EMR.

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/part-*'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Parameter

FROM

Sumber data yang akan dimuat.

'emr://emr _cluster_id/hdfs_file_path '

Pengidentifikasi unik untuk cluster Amazon EMR dan jalur file HDFS yang mereferensikan file data untuk perintah COPY. Nama file data HDFS tidak boleh berisi karakter wildcard asterisk (*) dan tanda tanya (?).

> **Note**
>
> Cluster Amazon EMR harus terus berjalan sampai operasi COPY selesai. Jika salah satu file data HDFS diubah atau dihapus sebelum operasi COPY selesai, Anda mungkin memiliki hasil yang tidak terduga, atau operasi COPY mungkin gagal.

Anda dapat menggunakan karakter wildcard asterisk (*) dan tanda tanya (?) sebagai bagian dari argumen hdfs_file_path untuk menentukan beberapa file yang akan dimuat. Misalnya, `'emr://j-SAMPLE2B500FC/myoutput/part*'` mengidentifikasi file`part-0000`,`part-0001`, dan sebagainya. Jika jalur file tidak berisi karakter wildcard, itu diperlakukan sebagai string literal. Jika Anda hanya menentukan nama folder, COPY mencoba memuat semua file dalam folder.

> **Important**
>
> Jika Anda menggunakan karakter wildcard atau hanya menggunakan nama folder, verifikasi bahwa tidak ada file yang tidak diinginkan akan dimuat. Misalnya, beberapa proses mungkin menulis file log ke folder keluaran.

Untuk informasi selengkapnya, lihat Memuat data dari Amazon EMR.

otorisasi

Perintah COPY memerlukan otorisasi untuk mengakses data di AWS sumber daya lain, termasuk di Amazon S3, Amazon EMR, Amazon DynamoDB, dan Amazon EC2. Anda dapat memberikan otorisasi tersebut dengan mereferensikan peran AWS Identity and Access Management (IAM) yang dilampirkan ke cluster Anda (kontrol akses berbasis peran) atau dengan memberikan kredentif akses untuk pengguna (kontrol akses berbasis kunci). Untuk meningkatkan keamanan

dan fleksibilitas, sebaiknya gunakan kontrol akses berbasis peran IAM. Untuk informasi selengkapnya, lihat Parameter otorisasi.

Parameter yang didukung

Anda dapat secara opsional menentukan parameter berikut dengan COPY dari Amazon EMR:

- Opsi pemetaan kolom
- Parameter format data
- Parameter konversi data
- Operasi beban data

Parameter yang tidak didukung

Anda tidak dapat menggunakan parameter berikut dengan COPY dari Amazon EMR:

- TERENKRIPSI
- MANIFESTASI
- WILAYAH
- RASIO BACA
- SSH

SALIN dari host jarak jauh (SSH)

Anda dapat menggunakan perintah COPY untuk memuat data secara paralel dari satu atau beberapa host jarak jauh, seperti instans Amazon Elastic Compute Cloud (Amazon EC2) atau komputer lain. COPY terhubung ke host jarak jauh menggunakan Secure Shell (SSH) dan menjalankan perintah pada host jarak jauh untuk menghasilkan output teks. Host jarak jauh dapat berupa instance Linux EC2 atau komputer Unix atau Linux lain yang dikonfigurasi untuk menerima koneksi SSH. Amazon Redshift dapat terhubung ke beberapa host, dan dapat membuka beberapa koneksi SSH ke setiap host. Amazon Redshift mengirimkan perintah unik melalui setiap koneksi untuk menghasilkan output teks ke output standar host, yang kemudian dibaca Amazon Redshift seperti file teks.

Gunakan klausa FROM untuk menentukan kunci objek Amazon S3 untuk file manifes yang menyediakan informasi yang digunakan COPY untuk membuka koneksi SSH dan menjalankan perintah jarak jauh.

Topik

- Sintaksis
- Contoh
- Parameter
- Parameter opsional
- Parameter yang tidak didukung

> **Important**
>
> Jika bucket S3 yang menyimpan file manifes tidak berada di Wil AWS ayah yang sama dengan cluster Anda, Anda harus menggunakan parameter REGION untuk menentukan Wilayah tempat bucket berada.

Sintaksis

```
FROM 's3://'ssh_manifest_file' }
authorization
SSH
| optional-parameters
```

Contoh

Contoh berikut menggunakan file manifes untuk memuat data dari host jarak jauh menggunakan SSH.

```
copy sales
from 's3://amzn-s3-demo-bucket/ssh_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
ssh;
```

Parameter

FROM

Sumber data yang akan dimuat.

's3://salin_dari_ssh_manifest_ file'

Perintah COPY dapat terhubung ke beberapa host menggunakan SSH, dan dapat membuat beberapa koneksi SSH ke setiap host. COPY menjalankan perintah melalui setiap koneksi host, dan kemudian memuat output dari perintah secara paralel ke dalam tabel. Argumen s3://copy_from_ssh_manifest_file menentukan kunci objek Amazon S3 untuk file manifes yang menyediakan informasi yang digunakan COPY untuk membuka koneksi SSH dan menjalankan perintah jarak jauh.

Argumen s3://copy_from_ssh_manifest_file harus secara eksplisit mereferensikan satu file; itu tidak bisa menjadi awalan kunci. Berikut ini menunjukkan contoh:

```
's3://amzn-s3-demo-bucket/ssh_manifest.txt'
```

File manifes adalah file teks dalam format JSON yang digunakan Amazon Redshift untuk terhubung ke host. File manifes menentukan titik akhir host SSH dan perintah yang akan dijalankan pada host untuk mengembalikan data ke Amazon Redshift. Secara opsional, Anda dapat menyertakan kunci publik host, nama pengguna login, dan tanda wajib untuk setiap entri. Contoh berikut menunjukkan file manifes yang membuat dua koneksi SSH:

```
{
    "entries": [
     {"endpoint":"<ssh_endpoint_or_IP>",
           "command": "<remote_command>",
           "mandatory":true,
           "publickey": "<public_key>",
           "username": "<host_user_name>"},
     {"endpoint":"<ssh_endpoint_or_IP>",
           "command": "<remote_command>",
           "mandatory":true,
           "publickey": "<public_key>",
           "username": "<host_user_name>"}
     ]
}
```

File manifes berisi satu "entries" konstruksi untuk setiap koneksi SSH. Anda dapat memiliki beberapa koneksi ke satu host atau beberapa koneksi ke beberapa host. Karakter tanda kutip ganda diperlukan seperti yang ditunjukkan, baik untuk nama bidang maupun nilainya. Karakter tanda kutip harus berupa tanda kutip sederhana (0x22), bukan tanda kutip miring atau “pintar”.

Satu-satunya nilai yang tidak memerlukan karakter tanda kutip ganda adalah nilai Boolean `true` atau `false` untuk bidang. `"mandatory"`

Daftar berikut menjelaskan bidang dalam file manifes.

titik akhir

Alamat URL atau alamat IP host—misalnya,`"ec2-111-222-333.compute-1.amazonaws.com"`, atau. `"198.51.100.0"`

perintah

Perintah yang akan dijalankan oleh host untuk menghasilkan output teks atau output biner dalam format gzip, lzop, bzip2, atau zstd. Perintah dapat berupa perintah apa pun yang pengguna "host_user_name" memiliki izin untuk menjalankan. Perintah dapat sesederhana mencetak file, atau dapat menanyakan database atau meluncurkan skrip. Output (file teks, file biner gzip, file biner lzop, atau file biner bzip2) harus dalam bentuk yang dapat diserap oleh perintah Amazon Redshift COPY. Untuk informasi selengkapnya, lihat Mempersiapkan data input Anda.

kunci publik

(Opsional) Kunci publik dari host. Jika disediakan, Amazon Redshift akan menggunakan kunci publik untuk mengidentifikasi host. Jika kunci publik tidak disediakan, Amazon Redshift tidak akan mencoba identifikasi host. Misalnya, jika kunci publik host jarak jauh adalah`ssh-rsa AbcCbaxxx…Example root@amazon.com`, ketikkan teks berikut di bidang kunci publik: `"AbcCbaxxx…Example"`

wajib

(Opsional) Klausa yang menunjukkan apakah perintah COPY harus gagal jika upaya koneksi gagal. Nilai default-nya `false`. Jika Amazon Redshift tidak berhasil membuat setidaknya satu koneksi, perintah COPY gagal.

nama pengguna

(Opsional) Nama pengguna yang akan digunakan untuk masuk ke sistem host dan menjalankan perintah jarak jauh. Nama login pengguna harus sama dengan login yang digunakan untuk menambahkan kunci publik cluster Amazon Redshift ke file kunci resmi host. Nama pengguna default adalah`redshift`.

Untuk informasi selengkapnya tentang membuat file manifes, lihatProses pemuatan data.

Untuk COPY dari host jarak jauh, parameter SSH harus ditentukan dengan perintah COPY. Jika parameter SSH tidak ditentukan, COPY mengasumsikan bahwa file yang ditentukan dengan FROM adalah file data dan akan gagal.

Jika Anda menggunakan kompresi otomatis, perintah COPY melakukan dua operasi pembacaan data, yang berarti akan menjalankan perintah jarak jauh dua kali. Operasi baca pertama adalah menyediakan sampel data untuk analisis kompresi, kemudian operasi baca kedua benar-benar memuat data. Jika menjalankan perintah jarak jauh dua kali dapat menyebabkan masalah, Anda harus menonaktifkan kompresi otomatis. Untuk menonaktifkan kompresi otomatis, jalankan perintah COPY dengan parameter COMPUPDATE disetel ke OFF. Untuk informasi selengkapnya, lihat Memuat tabel dengan kompresi otomatis.

Untuk prosedur terperinci untuk menggunakan COPY dari SSH, lihatMemuat data dari host jarak jauh.

otorisasi

Perintah COPY memerlukan otorisasi untuk mengakses data di AWS sumber daya lain, termasuk di Amazon S3, Amazon EMR, Amazon DynamoDB, dan Amazon EC2. Anda dapat memberikan otorisasi tersebut dengan mereferensikan peran AWS Identity and Access Management (IAM) yang dilampirkan ke cluster Anda (kontrol akses berbasis peran) atau dengan memberikan kredentif akses untuk pengguna (kontrol akses berbasis kunci). Untuk meningkatkan keamanan dan fleksibilitas, sebaiknya gunakan kontrol akses berbasis peran IAM. Untuk informasi selengkapnya, lihat Parameter otorisasi.

SSH

Klausa yang menentukan bahwa data akan dimuat dari host jarak jauh menggunakan protokol SSH. Jika Anda menentukan SSH, Anda juga harus menyediakan file manifes menggunakan s3://copy_from_ssh_manifest_file argumen.

> **Note**
>
> Jika Anda menggunakan SSH untuk menyalin dari host menggunakan alamat IP pribadi di VPC jarak jauh, VPC harus mengaktifkan perutean VPC yang ditingkatkan. Untuk informasi selengkapnya tentang perutean VPC yang Ditingkatkan, lihat Perutean VPC yang Ditingkatkan Amazon Redshift.

Parameter opsional

Anda dapat secara opsional menentukan parameter berikut dengan COPY dari SSH:

- Opsi pemetaan kolom
- Parameter format data
- Parameter konversi data
- Operasi beban data

Parameter yang tidak didukung

Anda tidak dapat menggunakan parameter berikut dengan COPY dari SSH:

- TERENKRIPSI
- MANIFESTASI
- RASIO PEMBACAAN

SALIN dari Amazon DynamoDB

Untuk memuat data dari tabel DynamoDB yang ada, gunakan klausa FROM untuk menentukan nama tabel DynamoDB.

Topik

- Sintaksis
- Contoh
- Parameter opsional
- Parameter yang tidak didukung

> **Important**
>
> Jika tabel DynamoDB tidak berada di Wilayah yang sama dengan cluster Amazon Redshift Anda, Anda harus menggunakan parameter REGION untuk menentukan Wilayah tempat data berada.

Sintaksis

```
FROM 'dynamodb://table-name'
authorization
READRATIO ratio
| REGION [AS] 'aws_region'
| optional-parameters
```

Contoh

Contoh berikut memuat data dari tabel DynamoDB.

```
copy favoritemovies from 'dynamodb://ProductCatalog'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
readratio 50;
```

Parameter

FROM

Sumber data yang akan dimuat.

'dynamodb://nama-tabel '

Nama tabel DynamoDB yang berisi data, misalnya. `'dynamodb://ProductCatalog'` Untuk detail tentang cara atribut DynamoDB dipetakan ke kolom Amazon Redshift, lihat. Memuat data dari tabel Amazon DynamoDB

Nama tabel DynamoDB unik untuk AWS akun, yang diidentifikasi oleh kredenSIAL AWS akses.

otorisasi

Perintah COPY memerlukan otorisasi untuk mengakses data di AWS sumber daya lain, termasuk di Amazon S3, Amazon EMR, DynamoDB, dan Amazon EC2. Anda dapat memberikan otorisasi tersebut dengan mereferensikan peran AWS Identity and Access Management (IAM) yang dilampirkan ke cluster Anda (kontrol akses berbasis peran) atau dengan memberikan kredentif akses untuk pengguna (kontrol akses berbasis kunci). Untuk meningkatkan keamanan dan fleksibilitas, sebaiknya gunakan kontrol akses berbasis peran IAM. Untuk informasi selengkapnya, lihat Parameter otorisasi.

Rasio READRATIO [AS]

Persentase throughput yang disediakan tabel DynamoDB untuk digunakan untuk pemuatan data. READRATIO diperlukan untuk COPY dari DynamoDB. Ini tidak dapat digunakan dengan COPY

dari Amazon S3. Kami sangat menyarankan untuk menetapkan rasio ke nilai yang kurang dari rata-rata throughput yang disediakan yang tidak digunakan. Nilai yang valid adalah bilangan bulat 1-200.

> **Important**
>
> Menyetel READRATIO ke 100 atau lebih tinggi memungkinkan Amazon Redshift untuk menggunakan keseluruhan throughput yang disediakan tabel DynamoDB, yang secara serius menurunkan kinerja operasi baca bersamaan terhadap tabel yang sama selama sesi COPY. Lalu lintas tulis tidak terpengaruh. Nilai yang lebih tinggi dari 100 diizinkan untuk memecahkan masalah skenario langka saat Amazon Redshift gagal memenuhi throughput tabel yang disediakan. Jika Anda memuat data dari DynamoDB ke Amazon Redshift secara berkelanjutan, pertimbangkan untuk mengatur tabel DynamoDB Anda sebagai deret waktu untuk memisahkan lalu lintas langsung dari operasi COPY.

Parameter opsional

Anda dapat secara opsional menentukan parameter berikut dengan COPY dari Amazon DynamoDB:

- Opsi pemetaan kolom
- Parameter konversi data berikut didukung:
  - ACCEPTANYDATE
  - BLANKSASNULL
  - DATEFORMAT
  - EMPTYASNULL
  - ROUNDEC
  - TIMEFORMAT
  - TRIMBLANKS
  - TRUNCATECOLUMNS
- Operasi beban data

Parameter yang tidak didukung

Anda tidak dapat menggunakan parameter berikut dengan COPY dari DynamoDB:

- Semua parameter format data
- MELARIKAN DIRI
- REKAMAN ISI
- ABAIKAN BLANKLINES
- ABAIKAN HEADER
- NULL
- HAPUS KUTIPAN
- ACCEPTINVCHARS
- MANIFESTASI
- TERENKRIPSI

Parameter otorisasi

Perintah COPY memerlukan otorisasi untuk mengakses data di AWS sumber daya lain, termasuk di Amazon S3, Amazon EMR, Amazon DynamoDB, dan Amazon EC2. Anda dapat memberikan otorisasi tersebut dengan mereferensikan peran AWS Identity and Access Management (IAM) yang dilampirkan ke cluster Anda (kontrol akses berbasis peran). Anda dapat mengenkripsi data muat Anda di Amazon S3.

Topik berikut memberikan detail selengkapnya dan contoh opsi otentikasi:

- Izin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY
- Role-based kontrol akses
- Key-based kontrol akses

Gunakan salah satu dari berikut ini untuk memberikan otorisasi untuk perintah COPY:

- the section called “Menggunakan IAM_ROLE”Parameter
- Parameter the section called “Menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY”
- Klausa the section called “Menggunakan KREDENSIAL”

Menggunakan parameter IAM_ROLE

IAM_ROLE

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah COPY berjalan.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Jika Anda menentukan IAM_ROLE, Anda tidak dapat menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY, SESSION_TOKEN, atau CREDENTIALS.

Berikut ini menunjukkan sintaks untuk parameter IAM_ROLE.

```
IAM_ROLE { default | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>' }
```

Untuk informasi selengkapnya, lihat Role-based kontrol akses.

Menggunakan parameter ACCESS_KEY_ID dan SECRET_ACCESS_KEY

AKSES_KEY_ID, KUNCI AKSES_RAHASIA

Metode otorisasi ini tidak disarankan.

> **Note**
>
> Alih-alih memberikan kredentif akses sebagai teks biasa, kami sangat menyarankan menggunakan otentikasi berbasis peran dengan menentukan parameter IAM_ROLE. Untuk informasi selengkapnya, lihat Role-based kontrol akses.

SESSION_TOKEN

Token sesi untuk digunakan dengan kredenSIAL akses sementara. Ketika SESSION_TOKEN ditentukan, Anda juga harus menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY untuk memberikan kredenSIAL kunci akses sementara. Jika Anda menentukan SESSION_TOKEN Anda tidak dapat menggunakan IAM_ROLE atau CREDENTIALS. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di Panduan Pengguna IAM.

> **Note**
>
> Alih-alih membuat kredentif keamanan sementara, kami sangat menyarankan untuk menggunakan otentikasi berbasis peran. Saat Anda mengotorisasi menggunakan peran IAM,

Amazon Redshift secara otomatis membuat kredenSIAL pengguna sementara untuk setiap sesi. Untuk informasi selengkapnya, lihat Role-based kontrol akses.

Berikut ini menunjukkan sintaks untuk parameter SESSION_TOKEN dengan parameter ACCESS_KEY_ID dan SECRET_ACCESS_KEY.

```
ACCESS_KEY_ID '<access-key-id>'
SECRET_ACCESS_KEY '<secret-access-key>'
SESSION_TOKEN '<temporary-token>';
```

Jika Anda menentukan SESSION_TOKEN Anda tidak dapat menggunakan CREDENTIALS atau IAM_ROLE.

Menggunakan parameter CREDENTIALS

KREDENSIBILITAS

Klausa yang menunjukkan metode yang akan digunakan cluster Anda saat mengakses AWS sumber daya lain yang berisi file data atau file manifes. Anda tidak dapat menggunakan parameter CREDENTIALS dengan IAM_ROLE atau ACCESS_KEY_ID dan SECRET_ACCESS_KEY.

Berikut ini menunjukkan sintaks untuk parameter CREDENTIALS.

```
[WITH] CREDENTIALS [AS] 'credentials-args'
```

Note

Untuk meningkatkan fleksibilitas, sebaiknya gunakan IAM_ROLE parameter alih-alih parameter CREDENTIALS.

Secara opsional, jika ENCRYPTED parameter digunakan, string credentials-args juga menyediakan kunci enkripsi.

String credentials-args peka huruf besar-kecil dan tidak boleh mengandung spasi.

Kata kunci WITH dan AS bersifat opsional dan diabaikan.

Anda dapat menentukan role-based access control atau key-based access control. Dalam kedua kasus tersebut, peran atau pengguna IAM harus memiliki izin yang diperlukan untuk mengakses

AWS sumber daya yang ditentukan. Untuk informasi selengkapnya, lihat Izin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY.

> **Note**
>
> Untuk melindungi AWS kredensibilitas Anda dan melindungi data sensitif, kami sangat menyarankan untuk menggunakan kontrol akses berbasis peran.

Untuk menentukan kontrol akses berbasis peran, berikan string credentials-args dalam format berikut.

```
'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
```

Untuk menggunakan kredentif token sementara, Anda harus memberikan ID kunci akses sementara, kunci akses rahasia sementara, dan token sementara. String credentials-args dalam format berikut.

```
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;token=<temporary-token>'
```

Perintah COPY menggunakan kontrol akses berbasis peran dengan kredenSIAL sementara akan menyerupai pernyataan contoh berikut:

```
COPY customer FROM 's3://amzn-s3-demo-bucket/mydata'
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key-id>;token=<temporary-token>'
```

Untuk informasi selengkapnya, lihat Kredensial keamanan sementara.

Jika ENCRYPTED parameter digunakan, string credentials-args dalam format berikut, di mana *<root-key>* nilai kunci root yang digunakan untuk mengenkripsi file.

```
CREDENTIALS
'<credentials-args>;master_symmetric_key=<root-key>'
```

Perintah COPY menggunakan kontrol akses berbasis peran dengan kunci enkripsi akan menyerupai pernyataan contoh berikut:

```
COPY customer FROM 's3://amzn-s3-demo-bucket/mydata'
CREDENTIALS
'aws_iam_role=arn:aws:iam::<account-id>:role/<role-name>;master_symmetric_key=<root-key>'
```

Opsi pemetaan kolom

Secara default, COPY menyisipkan nilai ke kolom tabel target dalam urutan yang sama seperti bidang yang terjadi dalam file data. Jika urutan kolom default tidak berfungsi, Anda dapat menentukan daftar kolom atau menggunakan ekspresi JSONPath untuk memetakan bidang data sumber ke kolom target.

- Column List
- JSONPaths File

Daftar kolom

Anda dapat menentukan daftar nama kolom yang dipisahkan koma untuk memuat bidang data sumber ke kolom target tertentu. Kolom dapat berada dalam urutan apa pun dalam pernyataan COPY, tetapi saat memuat dari file datar, seperti dalam bucket Amazon S3, urutannya harus sesuai dengan urutan data sumber.

Saat memuat dari tabel Amazon DynamoDB, urutan tidak masalah. Perintah COPY mencocokkan nama atribut dalam item yang diambil dari tabel DynamoDB dengan nama kolom di tabel Amazon Redshift. Untuk informasi selengkapnya, lihat Memuat data dari tabel Amazon DynamoDB

Format untuk daftar kolom adalah sebagai berikut.

```
COPY tablename (column1 [,column2, ...])
```

Jika kolom dalam tabel target dihilangkan dari daftar kolom, maka COPY memuat DEFAULT ekspresi kolom target.

Jika kolom target tidak memiliki default, maka COPY mencoba memuat NULL.

Jika COPY mencoba menetapkan NULL ke kolom yang didefinisikan sebagai NOT NULL, perintah COPY gagal.

Jika IDENTITY kolom disertakan dalam daftar kolom, maka juga EXPLICIT_IDS harus ditentukan; jika kolom IDENTITY dihilangkan, maka EXPLICIT_IDS tidak dapat ditentukan. Jika tidak ada daftar

kolom yang ditentukan, perintah berperilaku seolah-olah daftar kolom lengkap sesuai urutan telah ditentukan, dengan kolom IDENTITY dihilangkan jika EXPLICIT_IDS juga tidak ditentukan.

Jika kolom didefinisikan dengan GENERATED BY DEFAULT AS IDENTITY, maka kolom tersebut dapat disalin. Nilai dihasilkan atau diperbarui dengan nilai yang Anda berikan. Opsi EXPLICIT_IDS tidak diperlukan. COPY tidak memperbarui tanda air identitas tinggi. Untuk informasi selengkapnya, lihat GENERATED BY DEFAULT AS IDENTITY.

File JSONPaths

Saat memuat dari file data dalam format JSON atau Avro, COPY secara otomatis memetakan elemen data dalam data sumber JSON atau Avro ke kolom di tabel target. Itu dilakukan dengan mencocokkan nama bidang dalam skema Avro dengan nama kolom di tabel target atau daftar kolom.

Dalam beberapa kasus, nama kolom dan nama bidang tidak cocok, atau Anda perlu memetakan ke tingkat yang lebih dalam dalam hierarki data. Dalam kasus ini, Anda dapat menggunakan file JSONPaths untuk secara eksplisit memetakan elemen data JSON atau Avro ke kolom.

Untuk informasi selengkapnya, lihat File JSONPaths.

Parameter format data

Secara default, perintah COPY mengharapkan data sumber menjadi teks yang dibatasi karakter UTF-8. Pembatas default adalah karakter pipa (|). Jika data sumber dalam format lain, gunakan parameter berikut untuk menentukan format data:

- FORMAT
- CSV
- DELIMITER
- FIXEDWIDTH
- SHAPEFILE
- AVRO
- JSON format for COPY
- PARQUET
- ORC

Selain format data standar, COPY mendukung format data kolumnar berikut untuk COPY dari Amazon S3:

- ORC
- PARQUET

COPY dari format kolumnar didukung dengan batasan tertentu. Untuk informasi selengkapnya, lihat SALIN dari format data kolumnar.

Parameter format data

FORMAT [SEBAGAI]

(Opsional) Mengidentifikasi kata kunci format data. Argumen FORMAT dijelaskan sebagai berikut.

CSV [QUOTE [AS] 'quote_character']

Memungkinkan penggunaan format CSV dalam data input. Untuk secara otomatis menghindari pembatas, karakter baris baru, dan pengembalian kereta, lampirkan bidang dalam karakter yang ditentukan oleh parameter QUOTE. Karakter tanda kutip default adalah tanda kutip ganda ("). Ketika karakter tanda kutip digunakan dalam bidang, hindari karakter dengan karakter tanda kutip tambahan. Misalnya, jika karakter tanda kutip adalah tanda kutip ganda, untuk memasukkan string `A "quoted" word` file input harus menyertakan string`"A ""quoted"" word"`. Saat parameter CSV digunakan, pembatas default adalah koma (,). Anda dapat menentukan pembatas yang berbeda dengan menggunakan parameter DELIMITER.

Ketika bidang tertutup dalam tanda kutip, spasi putih antara pembatas dan karakter tanda kutip diabaikan. Jika pembatas adalah karakter spasi putih, seperti tab, pembatas tidak diperlakukan sebagai spasi putih.

CSV tidak dapat digunakan dengan FIXEDWIDTH, REMOVEQUOTES, atau ESCAPE.

KUTIPAN [AS] 'quote_character'

Tidak wajib. Menentukan karakter yang akan digunakan sebagai karakter tanda kutip saat menggunakan parameter CSV. Defaultnya adalah tanda kutip ganda ("). Jika Anda menggunakan parameter QUOTE untuk menentukan karakter tanda kutip selain tanda kutip ganda, Anda tidak perlu menghindari tanda kutip ganda di dalam bidang. Parameter QUOTE hanya dapat digunakan dengan parameter CSV. Kata kunci AS adalah opsional.

PEMBATAS [AS] ['del imiter_char']

Menentukan karakter yang digunakan untuk memisahkan bidang dalam file input, seperti karakter pipa (`|`), koma (`,`), tab (`\t`), atau beberapa karakter seperti`|~|`. Non-printable karakter didukung. Karakter juga dapat direpresentasikan dalam oktal sebagai unit UTF-8 kode mereka. Untuk oktal,

gunakan format '\ ddd', di mana' d 'adalah digit oktal (0—7). Pembatas default adalah karakter pipa (|), kecuali parameter CSV digunakan, dalam hal ini pembatas default adalah koma (). , Kata kunci AS adalah opsional. DELIMITER tidak dapat digunakan dengan FIXEDWIDTH.

FIXEDWIDTH 'fixedwidth_spec'

Memuat data dari file di mana setiap lebar kolom adalah panjang tetap, bukan kolom yang dipisahkan oleh pembatas. fixedwidth_spec adalah string yang menentukan label kolom dan lebar kolom yang ditentukan pengguna. Label kolom dapat berupa string teks atau integer, tergantung pada apa yang dipilih pengguna. Label kolom tidak memiliki hubungan dengan nama kolom. Urutan label/width pasangan harus sesuai dengan urutan kolom tabel dengan tepat. FIXEDWIDTH tidak dapat digunakan dengan CSV atau DELIMITER. Di Amazon Redshift, panjang kolom CHAR dan VARCHAR dinyatakan dalam byte, jadi pastikan bahwa lebar kolom yang Anda tentukan mengakomodasi panjang biner karakter multibyte saat menyiapkan file yang akan dimuat. Untuk informasi selengkapnya, lihat Jenis karakter.

Format untuk fixedwidth_ spec ditampilkan sebagai berikut:

```
'colLabel1:colWidth1,colLabel:colWidth2, ...'
```

SHAPEFILE [SIMPLIFICATE [AUTO] ['toleransi']]

Memungkinkan penggunaan format SHAPEFILE dalam data input. Secara default, kolom pertama dari shapefile adalah kolom `GEOMETRY` atau`IDENTITY`. Semua kolom berikutnya mengikuti urutan yang ditentukan dalam shapefile.

Anda tidak dapat menggunakan SHAPEFILE dengan FIXEDWIDTH, REMOVEQUOTES, atau ESCAPE.

Untuk menggunakan `GEOGRAPHY` objek dengan`COPY FROM SHAPEFILE`, pertama-tama menelan ke dalam `GEOMETRY` kolom, dan kemudian melemparkan objek ke `GEOGRAPHY` objek.

MENYEDERHANAKAN [toleransi]

(Opsional) Menyederhanakan semua geometri selama proses penyerapan menggunakan Ramer-Douglas-Peucker algoritma dan toleransi yang diberikan.

SIMPLIHKAN AUTO [toleransi]

(Opsional) Menyederhanakan hanya geometri yang lebih besar dari ukuran geometri maksimum. Penyederhanaan ini menggunakan Ramer-Douglas-Peucker algoritma dan toleransi yang dihitung secara otomatis jika ini tidak melebihi toleransi yang ditentukan.

Algoritma ini menghitung ukuran untuk menyimpan objek dalam toleransi yang ditentukan. Nilai toleransi adalah opsional.

Untuk contoh memuat shapefile, lihat. Memuat shapefile ke Amazon Redshift

AVRO [AS] 'av ro_option'

Menentukan bahwa data sumber dalam format Avro.

Format Avro didukung untuk COPY dari layanan dan protokol ini:

- Amazon S3
- Amazon EMR
- Host jarak jauh (SSH)

Avro tidak didukung untuk COPY dari DynamoDB.

Avro adalah protokol serialisasi data. File sumber Avro menyertakan skema yang mendefinisikan struktur data. Jenis skema Avro harus`record`. COPY menerima file Avro yang dibuat menggunakan codec default yang tidak terkompresi serta codec `snappy` kompresi `deflate` dan. Untuk informasi lebih lanjut tentang Avro, kunjungi Apache Av ro.

Nilai yang valid untuk avro_option adalah sebagai berikut:

- `'auto'`
- `'auto ignorecase'`
- `'s3://jsonpaths_file'`

Nilai default-nya `'auto'`.

COPY secara otomatis memetakan elemen data dalam data sumber Avro ke kolom di tabel target. Itu dilakukan dengan mencocokkan nama bidang dalam skema Avro dengan nama kolom di tabel target. Pencocokan peka huruf besar/kecil untuk `'auto'` dan tidak peka huruf besar/kecil. `'auto ignorecase'`

Nama kolom di tabel Amazon Redshift selalu huruf kecil, jadi ketika Anda menggunakan `'auto'` opsi, nama bidang yang cocok juga harus huruf kecil. Jika nama bidang tidak semuanya huruf kecil, Anda dapat menggunakan opsi. `'auto ignorecase'` Dengan `'auto'` argumen default, COPY hanya mengenali tingkat pertama bidang, atau bidang luar, dalam struktur.

Untuk memetakan nama kolom secara eksplisit ke nama bidang Avro, Anda dapat menggunakan aFile JSONPaths.

Secara default, COPY mencoba mencocokkan semua kolom dalam tabel target dengan nama bidang Avro. Untuk memuat subset kolom, Anda dapat secara opsional menentukan daftar kolom. Jika kolom dalam tabel target dihilangkan dari daftar kolom, COPY memuat DEFAULT ekspresi kolom target. Jika kolom target tidak memiliki default, COPY mencoba memuat NULL. Jika kolom disertakan dalam daftar kolom dan COPY tidak menemukan bidang yang cocok di data Avro, COPY mencoba memuat NULL ke kolom.

Jika COPY mencoba menetapkan NULL ke kolom yang didefinisikan sebagai NOT NULL, perintah COPY gagal.

Skema Avro

File data sumber Avro mencakup skema yang mendefinisikan struktur data. COPY membaca skema yang merupakan bagian dari file data sumber Avro untuk memetakan elemen data ke kolom tabel target. Contoh berikut menunjukkan skema Avro.

```
{
    "name": "person",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "guid", "type": "string"},
        {"name": "name", "type": "string"},
        {"name": "address", "type": "string"}]
}
```

Skema Avro didefinisikan menggunakan format JSON. Objek JSON tingkat atas berisi tiga pasangan nama-nilai dengan nama, atau kunci,, `"name"`, `"type"` dan. `"fields"`

K `"fields"` unci berpasangan dengan array objek yang menentukan nama dan tipe data dari setiap bidang dalam struktur data. Secara default, COPY secara otomatis mencocokkan nama bidang dengan nama kolom. Nama kolom selalu huruf kecil, jadi nama bidang yang cocok juga harus huruf kecil, kecuali Anda menentukan opsi. `‘auto ignorecase’` Setiap nama bidang yang tidak cocok dengan nama kolom diabaikan. Pesanan tidak masalah. Pada contoh sebelumnya, COPY memetakan ke nama kolom`id`,`guid`,`name`, dan`address`.

Dengan `'auto'` argumen default, COPY hanya mencocokkan objek tingkat pertama ke kolom. Untuk memetakan ke tingkat yang lebih dalam dalam skema, atau jika nama bidang dan nama kolom tidak cocok, gunakan file JSONPaths untuk menentukan pemetaan. Untuk informasi selengkapnya, lihat File JSONPaths.

Jika nilai yang terkait dengan kunci adalah tipe data Avro yang kompleks seperti byte, array, record, map, atau link, COPY memuat nilai sebagai string. Di sini, string adalah representasi JSON dari data. COPY memuat tipe data Avro enum sebagai string, di mana konten adalah nama jenisnya. Sebagai contoh, lihat SALIN dari format JSON.

Ukuran maksimum header file Avro, yang mencakup skema dan metadata file, adalah 1 MB.

Ukuran maksimum blok data Avro tunggal adalah 4 MB. Ini berbeda dari ukuran baris maksimum. Jika ukuran maksimum blok data Avro tunggal terlampaui, bahkan jika ukuran baris yang dihasilkan kurang dari batas ukuran baris 4 MB, perintah COPY gagal.

Dalam menghitung ukuran baris, Amazon Redshift secara internal menghitung karakter pipa (|) dua kali. Jika data input Anda berisi sejumlah besar karakter pipa, dimungkinkan untuk ukuran baris melebihi 4 MB meskipun blok data kurang dari 4 MB.

JSON [AS] 'json_option'

Sumber data dalam format JSON.

Format JSON didukung untuk COPY dari layanan dan protokol ini:

- Amazon S3
- SALIN dari Amazon EMR
- COPY dari SSH

JSON tidak didukung untuk COPY dari DynamoDB.

Nilai yang valid untuk json_option adalah sebagai berikut:

- `'auto'`
- `'auto ignorecase'`
- `'s3://jsonpaths_file'`
- `'noshred'`

Nilai default-nya `'auto'`. Amazon Redshift tidak merobek atribut struktur JSON menjadi beberapa kolom saat memuat dokumen JSON.

Secara default, COPY mencoba mencocokkan semua kolom di tabel target dengan kunci nama bidang JSON. Untuk memuat subset kolom, Anda dapat secara opsional menentukan daftar kolom. Jika kunci nama bidang JSON tidak semuanya huruf kecil, Anda dapat menggunakan `'auto ignorecase'` opsi atau a File JSONPaths untuk secara eksplisit memetakan nama kolom ke kunci nama bidang JSON.

Jika kolom dalam tabel target dihilangkan dari daftar kolom, maka COPY memuat DEFAULT ekspresi kolom target. Jika kolom target tidak memiliki default, COPY mencoba memuat NULL. Jika kolom disertakan dalam daftar kolom dan COPY tidak menemukan bidang yang cocok dalam data JSON, COPY mencoba memuat NULL ke kolom.

Jika COPY mencoba menetapkan NULL ke kolom yang didefinisikan sebagai NOT NULL, perintah COPY gagal.

COPY memetakan elemen data dalam data sumber JSON ke kolom di tabel target. Ia melakukannya dengan mencoco kkan kunci objek, atau nama, dalam pasangan nama-nilai sumber dengan nama kolom dalam tabel target.

Lihat detail berikut tentang setiap nilai json_option:

'otomatis'

Dengan opsi ini, pencocokan peka huruf besar/kecil. Nama kolom di tabel Amazon Redshift selalu huruf kecil, jadi saat Anda menggunakan `'auto'` opsi, nama bidang JSON yang cocok juga harus huruf kecil.

'mengabaikan otomatis'

Dengan opsi ini, pencocokan tidak peka huruf besar/kecil. Nama kolom dalam tabel Amazon Redshift selalu huruf kecil, jadi ketika Anda menggunakan `'auto ignorecase'` opsi, nama bidang JSON yang sesuai dapat berupa huruf kecil, huruf besar, atau huruf campuran.

's3://file js onpaths_'

Dengan opsi ini, COPY menggunakan file JSONPaths bernama untuk memetakan elemen data dalam data sumber JSON ke kolom di tabel target. *`s3://jsonpaths_file`*Argumen harus berupa kunci objek Amazon S3 yang secara eksplisit mereferensikan satu file. Contohnya adalah `'s3://amzn-s3-demo-bucket/jsonpaths.txt'`. Argumen tidak bisa menjadi awalan kunci. Untuk informasi selengkapnya tentang menggunakan file JSONPaths, lihat. the section called “File JSONPaths”

Dalam beberapa kasus, file yang ditentukan oleh `jsonpaths_file` memiliki awalan yang sama dengan jalur yang ditentukan oleh `copy_from_s3_objectpath` untuk file data. Jika demikian, COPY membaca file JSONPaths sebagai file data dan mengembalikan kesalahan. Misalnya, misalkan file data Anda menggunakan jalur objek `s3://amzn-s3-demo-bucket/my_data.json` dan file JSONPaths Anda adalah. `s3://amzn-s3-demo-bucket/my_data.jsonpaths` Dalam hal ini, COPY mencoba memuat `my_data.jsonpaths` sebagai file data.

'noshre'

Dengan opsi ini, Amazon Redshift tidak merobek atribut struktur JSON menjadi beberapa kolom saat memuat dokumen JSON.

File data JSON

File data JSON berisi satu set objek atau array. COPY memuat setiap objek JSON atau array ke dalam satu baris di tabel target. Setiap objek atau array yang sesuai dengan baris harus merupakan struktur tingkat root yang berdiri sendiri; artinya, itu tidak boleh menjadi anggota dari struktur JSON lain.

Ob jek JSON dimulai dan diakhiri dengan kurung kurung ({}) dan berisi kumpulan pasangan nama-nilai yang tidak teratur. Setiap nama dan nilai yang dipasangkan dipisahkan oleh titik dua, dan pasangan dipisahkan oleh koma. Secara default, kunci objek, atau nama, dalam pasangan nama-nilai harus cocok dengan nama kolom yang sesuai dalam tabel. Nama kolom dalam tabel Amazon Redshift selalu huruf kecil, jadi kunci nama bidang JSON yang cocok juga harus huruf kecil. Jika nama kolom dan kunci JSON Anda tidak cocok, gunakan a the section called “File JSONPaths” untuk secara eksplisit memetakan kolom ke kunci.

Urutan dalam objek JSON tidak masalah. Setiap nama yang tidak cocok dengan nama kolom diabaikan. Berikut ini menunjukkan struktur objek JSON sederhana.

```
{
  "column1": "value1",
  "column2": value2,
  "notacolumn" : "ignore this value"
}
```

Sebuah array JSON dimulai dan diakhiri dengan tanda kurung ([]), dan berisi kumpulan nilai yang dipisahkan oleh koma. Jika file data Anda menggunakan array, Anda harus menentukan file JSONPaths untuk mencocokkan nilai ke kolom. Berikut ini menunjukkan struktur array JSON sederhana.

```
["value1", value2]
```

JSON harus dibentuk dengan baik. Misalnya, objek atau array tidak dapat dipisahkan dengan koma atau karakter lain kecuali spasi putih. String harus tertutup dalam karakter tanda kutip ganda. Karakter kutipan harus berupa tanda kutip sederhana (0x22), bukan tanda kutip miring atau “pintar”.

Ukuran maksimum objek atau array JSON tunggal, termasuk kurung kurung atau tanda kurung, adalah 4 MB. Ini berbeda dari ukuran baris maksimum. Jika ukuran maksimum objek atau array JSON tunggal terlampaui, bahkan jika ukuran baris yang dihasilkan kurang dari batas ukuran baris 4 MB, perintah COPY gagal.

Dalam menghitung ukuran baris, Amazon Redshift secara internal menghitung karakter pipa (|) dua kali. Jika data input Anda berisi sejumlah besar karakter pipa, dimungkinkan untuk ukuran baris melebihi 4 MB meskipun ukuran objek kurang dari 4 MB.

COPY di `\n` muat sebagai karakter baris baru dan memu `\t` at sebagai karakter tab. Untuk memuat garis miring terbalik, lepaskan dengan backslash (). `\\`

COPY mencari sumber JSON yang ditentukan untuk objek atau array JSON yang terbentuk dengan baik dan valid. Jika COPY menemukan karakter bukan spasi putih sebelum menemukan struktur JSON yang dapat digunakan, atau di antara objek atau array JSON yang valid, COPY mengembalikan kesalahan untuk setiap instance. Kesalahan ini diperhitungkan dalam hitungan kesalahan MAXERROR. Ketika jumlah kesalahan sama atau melebihi MAXERROR, COPY gagal.

Untuk setiap kesalahan, Amazon Redshift merekam baris dalam tabel sistem STL_LOAD_ERRORS. Kolom LINE_NUMBER mencatat baris terakhir dari objek JSON yang menyebabkan kesalahan.

Jika IGNOREHEADER ditentukan, COPY mengabaikan jumlah baris yang ditentukan dalam data JSON. Karakter baris baru dalam data JSON selalu dihitung untuk perhitungan IGNOREHEADER.

COPY memuat string kosong sebagai bidang kosong secara default. Jika EMPTYASNULL ditentukan, COPY memuat string kosong untuk bidang CHAR dan VARCHAR sebagai NULL. String kosong untuk tipe data lain, seperti INT, selalu dimuat dengan NULL.

Opsi berikut tidak didukung dengan JSON:

- CSV
- PEMBATAS
- MELARIKAN DIRI
- REKAMAN ISI
- LEBAR TETAP
- ABAIKAN BLANKLINES
- NULL SEBAGAI
- RASIO PEMBACAAN

- HAPUS KUTIPAN

Untuk informasi selengkapnya, lihat SALIN dari format JSON. Untuk informasi lebih lanjut tentang struktur data JSON, kunjungi www.js on.org.

File JSONPaths

Jika Anda memuat dari JSON-formatted atau data sumber Avro, secara default COPY memetakan elemen data tingkat pertama dalam data sumber ke kolom di tabel target. Ia melakukannya dengan mencocokkan setiap nama, atau kunci objek, dalam pasangan nama-nilai dengan nama kolom di tabel target.

Jika nama kolom dan kunci objek tidak cocok, atau untuk memetakan ke tingkat yang lebih dalam dalam hierarki data, Anda dapat menggunakan file JSONPaths untuk secara eksplisit memetakan elemen data JSON atau Avro ke kolom. File JSONPaths memetakan elemen data JSON ke kolom dengan mencocokkan urutan kolom dalam tabel target atau daftar kolom.

File JSONPaths harus berisi hanya satu objek JSON (bukan array). Objek JSON adalah pasangan nama-nilai. K unci objek, yang merupakan nama dalam pasangan nama-nilai, harus. `"jsonpaths"` Nilai dalam pasangan nama-nilai adalah array ekspresi JSONPath. Setiap ekspresi JSONPath mereferensikan elemen tunggal dalam hierarki data JSON atau skema Avro, mirip dengan bagaimana ekspresi XPath mengacu pada elemen dalam dokumen XML. Untuk informasi selengkapnya, lihat Ekspresi JSONPath.

Untuk menggunakan file JSONPaths, tambahkan kata kunci JSON atau AVRO ke perintah COPY. Tentukan nama bucket S3 dan jalur objek dari file JSONPaths menggunakan format berikut.

```
COPY tablename
FROM 'data_source'
CREDENTIALS 'credentials-args'
FORMAT AS { AVRO | JSON } 's3://jsonpaths_file';
```

`s3://jsonpaths_file`Nilai harus berupa kunci objek Amazon S3 yang secara eksplisit mereferensikan satu file, seperti`'s3://amzn-s3-demo-bucket/jsonpaths.txt'`. Itu tidak bisa menjadi awalan kunci.

Dalam beberapa kasus, jika Anda memuat dari Amazon S3, file yang ditentukan oleh `jsonpaths_file` memiliki awalan yang sama dengan jalur yang ditentukan oleh `copy_from_s3_objectpath` untuk file data. Jika demikian, COPY membaca file JSONPaths sebagai file data dan mengembalikan kesalahan. Misalnya, misalkan file data Anda menggunakan

jalur objek `s3://amzn-s3-demo-bucket/my_data.json` dan file JSONPaths Anda adalah. `s3://amzn-s3-demo-bucket/my_data.jsonpaths` Dalam hal ini, COPY mencoba memuat `my_data.jsonpaths` sebagai file data.

Jika nama kunci adalah string apa pun selain`"jsonpaths"`, perintah COPY tidak mengembalikan kesalahan, tetapi mengabaikan jsonpaths_file dan menggunakan argumen sebagai gantinya. `'auto'`

Jika salah satu dari hal berikut terjadi, perintah COPY gagal:

- JSON salah format.
- Ada lebih dari satu objek JSON.
- Karakter apa pun kecuali ruang putih ada di luar objek.
- Elemen array adalah string kosong atau bukan string.

MAXERROR tidak berlaku untuk file JSONPaths.

File JSONPaths tidak boleh dienkripsi, bahkan jika ENCRYPTED opsi ditentukan.

Untuk informasi selengkapnya, lihat SALIN dari format JSON.

Ekspresi JSONPath

File JSONPaths menggunakan ekspresi JSONPath untuk memetakan bidang data ke kolom target. Setiap ekspresi JSONPath sesuai dengan satu kolom dalam tabel target Amazon Redshift. Urutan elemen array JSONPath harus sesuai dengan urutan kolom dalam tabel target atau daftar kolom, jika daftar kolom digunakan.

Karakter tanda kutip ganda diperlukan seperti yang ditunjukkan, baik untuk nama bidang maupun nilainya. Karakter tanda kutip harus berupa tanda kutip sederhana (0x22), bukan tanda kutip miring atau “pintar”.

Jika elemen objek yang direferensikan oleh ekspresi JSONPath tidak ditemukan dalam data JSON, COPY mencoba memuat nilai NULL. Jika objek yang direferensikan cacat, COPY mengembalikan kesalahan pemuatan.

Jika elemen array yang direferensikan oleh ekspresi JSONPath tidak ditemukan dalam data JSON atau Avro, COPY gagal dengan kesalahan berikut: H `Invalid JSONPath format: Not an array or index out of range.` apus elemen array dari JSONPaths yang tidak ada di data sumber dan verifikasi bahwa array dalam data sumber terbentuk dengan baik.

Ekspresi JSONPath dapat menggunakan notasi tanda kurung atau notasi titik, tetapi Anda tidak dapat mencampur notasi. Contoh berikut menunjukkan ekspresi JSONPath menggunakan notasi tanda kurung.

```
{
    "jsonpaths": [
        "$['venuename']",
        "$['venuecity']",
        "$['venuestate']",
        "$['venueseats']"
    ]
}
```

Contoh berikut menunjukkan ekspresi JSONPath menggunakan notasi titik.

```
{
    "jsonpaths": [
        "$.venuename",
        "$.venuecity",
        "$.venuestate",
        "$.venueseats"
    ]
}
```

Dalam konteks sintaks Amazon Redshift COPY, ekspresi JSONPath harus menentukan jalur eksplisit ke elemen nama tunggal dalam struktur data hierarkis JSON atau Avro. Amazon Redshift tidak mendukung elemen JSONPath apa pun, seperti karakter wildcard atau ekspresi filter, yang mungkin berubah menjadi jalur ambigu atau beberapa elemen nama.

Untuk informasi selengkapnya, lihat SALIN dari format JSON.

Menggunakan JSONPaths dengan Data Avro

Contoh berikut menunjukkan skema Avro dengan beberapa level.

```
{
    "name": "person",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "guid", "type": "string"},
        {"name": "isActive", "type": "boolean"},
```

```
        {"name": "age", "type": "int"},
        {"name": "name", "type": "string"},
        {"name": "address", "type": "string"},
        {"name": "latitude", "type": "double"},
        {"name": "longitude", "type": "double"},
        {
            "name": "tags",
            "type": {
                        "type" : "array",
                        "name" : "inner_tags",
                        "items" : "string"
                    }
        },
        {
            "name": "friends",
            "type": {
                        "type" : "array",
                        "name" : "inner_friends",
                        "items" : {
                                    "name" : "friends_record",
                                    "type" : "record",
                                    "fields" : [
                                                {"name" : "id", "type" : "int"},
                                                {"name" : "name", "type" : "string"}
                                               ]
                                   }
                    }
        },
        {"name": "randomArrayItem", "type": "string"}
    ]
}
```

Contoh berikut menunjukkan file JSONPaths yang menggunakan AvroPath ekspresi untuk mereferensikan skema sebelumnya.

```
{
    "jsonpaths": [
        "$.id",
        "$.guid",
        "$.address",
        "$.friends[0].id"
    ]
}
```

Contoh JSONPaths mencakup elemen-elemen berikut:

jsonpath

Nama objek JSON yang berisi AvroPath ekspresi.

[…]

Kurung melingkupi array JSON yang berisi elemen jalur.

$

Tanda dolar mengacu pada elemen root dalam skema Avro, yang merupakan `"fields"` array.

“$.id”,

Target AvroPath ekspresi. Dalam hal ini, target adalah elemen dalam `"fields"` array dengan nama`"id"`. Ekspresi dipisahkan dengan koma.

“$.teman [0] .id”

Kurung menunjukkan indeks array. Ekspresi JSONPath menggunakan pengindeksan berbasis nol, jadi ekspresi ini mereferensikan elemen pertama dalam `"friends"` array dengan nama. `"id"`

Sintaks skema Avro memerlukan penggunaan bidang dalam untuk menentukan struktur tipe data record dan array. Bidang dalam diabaikan oleh AvroPath ekspresi. Misalnya, bidang men `"friends"` definisikan sebuah array bernama`"inner_friends"`, yang pada gilirannya mendefinisikan rekaman bernama`"friends_record"`. Eksp AvroPath resi untuk mereferensi `"id"` kan bidang dapat mengabaikan bidang tambahan untuk mereferensikan bidang target secara langsung. Eksp AvroPath resi berikut mereferensikan dua bidang yang termasuk dalam `"friends"` array.

```
"$.friends[0].id"
"$.friends[0].name"
```

Parameter format data kolumnar

Selain format data standar, COPY mendukung format data kolumnar berikut untuk COPY dari Amazon S3. COPY dari format kolumnar didukung dengan batasan tertentu. Untuk informasi selengkapnya, lihat SALIN dari format data kolumnar.

ORC

Memuat data dari file yang menggunakan format file Optimized Row Columnar (ORC).

PARQUET

Memuat data dari file yang menggunakan format file Parquet.

Parameter kompresi file

Anda dapat memuat dari file data terkompresi dengan menentukan parameter berikut.

Parameter kompresi file

BZIP2

Nilai yang menentukan bahwa file input atau file dalam format bzip2 terkompresi (file.bz2). Operasi COPY membaca setiap file terkompresi dan membuka kompres data saat dimuat.

GZIP

Nilai yang menentukan bahwa file input atau file dalam format gzip terkompresi (file.gz). Operasi COPY membaca setiap file terkompresi dan membuka kompres data saat dimuat.

LZOP

Nilai yang menentukan bahwa file input atau file dalam format lzop terkompresi (file.lzo). Operasi COPY membaca setiap file terkompresi dan membuka kompres data saat dimuat.

> **Note**
>
> COPY tidak mendukung file yang dikompresi menggunakan opsi lzop --filter.

ZSTD

Nilai yang menentukan bahwa file input atau file dalam format Zstandard terkompresi (file.zst). Operasi COPY membaca setiap file terkompresi dan membuka kompres data saat dimuat.

> **Note**
>
> ZSTD hanya didukung dengan COPY dari Amazon S3.

Parameter konversi data

Saat memuat tabel, COPY mencoba untuk secara implisit mengonversi string dalam data sumber ke tipe data kolom target. Jika Anda perlu menentukan konversi yang berbeda dari perilaku default, atau jika konversi default menghasilkan kesalahan, Anda dapat mengelola konversi data dengan menentukan parameter berikut. Untuk informasi selengkapnya tentang sintaks parameter ini, lihat sintaks COPY.

- ACCEPTANYDATE
- ACCEPTINVCHARS
- BLANKSASNULL
- DATEFORMAT
- EMPTYASNULL
- ENCODING
- ESCAPE
- EXPLICIT_IDS
- FILLRECORD
- IGNOREBLANKLINES
- IGNOREHEADER
- NULL AS
- REMOVEQUOTES
- ROUNDEC
- TIMEFORMAT
- TRIMBLANKS
- TRUNCATECOLUMNS

Parameter konversi data

TANGGAL YANG DITERIMA

Memungkinkan format tanggal apa pun, termasuk format yang tidak valid seperti`00/00/00 00:00:00`, untuk dimuat tanpa menghasilkan kesalahan. Parameter ini hanya berlaku untuk kolom TIMESTAMP dan DATE. Selalu gunakan ACCEPTANYDATE dengan parameter DATEFORMAT. Jika format tanggal untuk data tidak sesuai dengan spesifikasi DATEFORMAT, Amazon Redshift menyisipkan nilai NULL ke bidang tersebut.

ACCEPTINVCHARS [AS] ['replacement_char']

Memungkinkan pemuatan data ke kolom VARCHAR meskipun data berisi UTF-8 karakter yang tidak valid. Ketika ACCEPTINVCHARS ditentukan, COPY mengganti setiap karakter yang tidak valid dengan string dengan panjang yang sama yang terdiri dari UTF-8 karakter yang ditentukan oleh replacement_char. Misalnya, jika karakter pengganti adalah '^', karakter tiga byte yang tidak valid akan diganti dengan '^^^'.

Karakter pengganti dapat berupa karakter ASCII apa pun kecuali NULL. Defaultnya adalah tanda tanya (?). Untuk informasi tentang UTF-8 karakter yang tidak valid, lihatKesalahan pemuatan karakter multibyte.

COPY mengembalikan jumlah baris yang berisi UTF-8 karakter tidak valid, dan menambahkan entri ke tabel STL_PENGGANTIAN sistem untuk setiap baris yang terpengaruh, hingga maksimum 100 baris untuk setiap irisan node. Karakter tambahan yang tidak valid UTF-8 juga diganti, tetapi peristiwa pengganti tersebut tidak direkam.

Jika ACCEPTINVCHARS tidak ditentukan, COPY mengembalikan kesalahan setiap kali menemukan karakter yang tidak valid. UTF-8

ACCEPTINVCHARS hanya valid untuk kolom VARCHAR.

BLANKSASNULL

Memuat bidang kosong, yang hanya terdiri dari karakter spasi putih, sebagai NULL. Opsi ini hanya berlaku untuk kolom CHAR dan VARCHAR. Bidang kosong untuk tipe data lain, seperti INT, selalu dimuat dengan NULL. Misalnya, string yang berisi tiga karakter spasi secara berurutan (dan tidak ada karakter lain) dimuat sebagai NULL. Perilaku default, tanpa opsi ini, adalah memuat karakter spasi apa adanya.

DATEFORMAT [AS] {'dateformat_string' | 'otomatis'}

Jika tidak ada DATEFORMAT yang ditentukan, format defaultnya adalah `'YYYY-MM-DD'`. Misalnya, format valid alternatif adalah `'MM-DD-YYYY'`.

Jika perintah COPY tidak mengenali format nilai tanggal atau waktu Anda, atau jika nilai tanggal atau waktu Anda menggunakan format yang berbeda, gunakan `'auto'` argumen dengan parameter DATEFORMAT atau TIMEFORMAT. `'auto'`Argumen mengenali beberapa format yang tidak didukung saat menggunakan string DATEFORMAT dan TIMEFORMAT. Kata kunci `'auto'` 'peka huruf besar/kecil. Untuk informasi selengkapnya, lihat Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT.

Format tanggal dapat mencakup informasi waktu (jam, menit, detik), tetapi informasi ini diabaikan. Kata kunci AS adalah opsional. Untuk informasi selengkapnya, lihat String DATEFORMAT dan TIMEFORMAT.

EMPTYASNULL

Menunjukkan bahwa Amazon Redshift harus memuat bidang CHAR dan VARCHAR kosong sebagai NULL. Bidang kosong untuk tipe data lain, seperti INT, selalu dimuat dengan NULL. Bidang kosong terjadi ketika data berisi dua pembatas secara berurutan tanpa karakter di antara pembatas. EMPTYASNULL dan NULL AS "(string kosong) menghasilkan perilaku yang sama.

PENGKODEAN [AS] file_encoding

Menentukan jenis pengkodean data beban. Perintah COPY mengubah data dari pengkodean yang ditentukan menjadi UTF-8 selama pemuatan.

Nilai yang valid untuk file_encoding adalah sebagai berikut:

- UTF8
- UTF16
- UTF16LE
- UTF16BE
- ISO88591

Nilai default-nya UTF8.

Nama file sumber harus menggunakan UTF-8 encoding.

File-file berikut harus menggunakan UTF-8 pengkodean, bahkan jika pengkodean yang berbeda ditentukan untuk data beban:

- File manifes
- File JSONPaths

String argumen yang disediakan dengan parameter berikut harus menggunakan UTF-8:

- FIXEDWIDTH 'fixedwidth_spec'
- ACCEPTINVCHARS 'pengganti_char'
- FORMAT TANGGAL 'dateformat_string'
- FORMAT WAKTU 'timeformat_string'
- NULL SEBAGAI 'null_string'

Fixed-width file data harus menggunakan UTF-8 encoding. Lebar bidang didasarkan pada jumlah karakter, bukan jumlah byte.

Semua data beban harus menggunakan pengkodean yang ditentukan. Jika COPY menemukan pengkodean yang berbeda, itu melewatkan file dan mengembalikan kesalahan.

Jika Anda menentukanUTF16, maka data Anda harus memiliki tanda urutan byte (BOM). Jika Anda tahu apakah UTF-16 data Anda little-endian (LE) atau big-endian (BE), Anda dapat menggunakan UTF16LE atauUTF16BE, terlepas dari keberadaan BOM.

Untuk menggunakan ISO-8859-1 pengkodean tentukanISO88591. Untuk informasi lebih lanjut, lihat ISO/IEC 8859-1 di Wikipedia.

MELARIKAN DIRI

Ketika parameter ini ditentukan, karakter garis miring terbalik (\) dalam data input diperlakukan sebagai karakter escape. Karakter yang segera mengikuti karakter garis miring terbalik dimuat ke dalam tabel sebagai bagian dari nilai kolom saat ini, bahkan jika itu adalah karakter yang biasanya melayani tujuan khusus. Misalnya, Anda dapat menggunakan parameter ini untuk menghindari karakter pembatas, tanda kutip, karakter baris baru yang disematkan, atau karakter escape itu sendiri ketika salah satu karakter ini merupakan bagian yang sah dari nilai kolom.

Jika Anda menentukan parameter ESCAPE dalam kombinasi dengan parameter REMOVEQUOTES, Anda dapat menghindari dan mempertahankan tanda kutip (' atau ") yang mungkin dihapus. String null default\N,, berfungsi apa adanya, tetapi juga dapat lolos dalam data input sebagai\\N. Selama Anda tidak menentukan string null alternatif dengan parameter NULL AS, \N dan \\N menghasilkan hasil yang sama.

> **Note**
>
> Karakter kontrol 0x00 (NUL) tidak dapat dilepaskan dan harus dihapus dari data input atau dikonversi. Karakter ini diperlakukan sebagai penanda akhir catatan (EOR), menyebabkan sisa catatan terpotong.

Anda tidak dapat menggunakan parameter ESCAPE untuk pemuatan FIXEDWIDTH, dan Anda tidak dapat menentukan karakter escape itu sendiri; karakter escape selalu merupakan karakter garis miring terbalik. Juga, Anda harus memastikan bahwa data input berisi karakter escape di tempat yang sesuai.

Berikut adalah beberapa contoh data input dan data yang dimuat yang dihasilkan ketika parameter ESCAPE ditentukan. Hasil untuk baris 4 mengasumsikan bahwa parameter REMOVEQUOTES juga ditentukan. Data input terdiri dari dua bidang yang dibatasi pipa:

```
1|The quick brown fox\[newline]
jumped over the lazy dog.
2| A\\B\\C
3| A \| B \| C
4| 'A Midsummer Night\'s Dream'
```

Data yang dimuat ke kolom 2 terlihat seperti ini:

```
The quick brown fox
jumped over the lazy dog.
A\B\C
A|B|C
A Midsummer Night's Dream
```

> **Note**
>
> Menerapkan karakter escape ke data input untuk beban adalah tanggung jawab pengguna. Satu pengecualian untuk persyaratan ini adalah ketika Anda memuat ulang data yang sebelumnya dibongkar dengan parameter ESCAPE. Dalam hal ini, data sudah akan berisi karakter pelarian yang diperlukan.

Parameter ESCAPE tidak menafsirkan oktal, hex, Unicode, atau notasi urutan escape lainnya. Misalnya, jika data sumber Anda berisi nilai umpan garis oktal (`\012`) dan Anda mencoba memuat data ini dengan parameter ESCAPE, Amazon Redshift memuat nilai `012` ke dalam tabel dan tidak menafsirkan nilai ini sebagai umpan baris yang sedang lolos.

Untuk menghindari karakter baris baru dalam data yang berasal dari platform Microsoft Windows, Anda mungkin perlu menggunakan dua karakter escape: satu untuk pengembalian kereta dan satu untuk umpan baris. Atau, Anda dapat menghapus pengembalian kereta sebelum memuat file (misalnya, dengan menggunakan utilitas dos2unix).

EXPLICIT_ID

Gunakan EXPLICIT_IDS dengan tabel yang memiliki kolom IDENTITY jika Anda ingin mengganti nilai yang dibuat secara otomatis dengan nilai eksplisit dari file data sumber untuk tabel. Jika

perintah menyertakan daftar kolom, daftar itu harus menyertakan kolom IDENTITY untuk menggunakan parameter ini. Format data untuk nilai EXPLICIT_IDS harus sesuai dengan format IDENTITY yang ditentukan oleh definisi CREATE TABLE.

Saat Anda menjalankan perintah COPY terhadap tabel dengan opsi EXPLICIT_IDS, Amazon Redshift tidak memeriksa keunikan kolom IDENTITY dalam tabel.

Jika kolom didefinisikan dengan GENERATED BY DEFAULT AS IDENTITY, maka kolom tersebut dapat disalin. Nilai dihasilkan atau diperbarui dengan nilai yang Anda berikan. Opsi EXPLICIT_IDS tidak diperlukan. COPY tidak memperbarui tanda air identitas tinggi.

Untuk contoh perintah COPY menggunakan EXPLICIT_IDS, lihat. Muat VENUE dengan nilai eksplisit untuk kolom IDENTITY

REKAMAN ISI

Memungkinkan file data dimuat ketika kolom yang berdekatan hilang di akhir beberapa catatan. Kolom yang hilang dimuat sebagai NULL. Untuk format teks dan CSV, jika kolom yang hilang adalah kolom VARCHAR, string panjang nol dimuat alih-alih NULL. Untuk memuat NULL ke kolom VARCHAR dari teks dan CSV, tentukan kata kunci EMPTYASNULL. Substitusi NULL hanya berfungsi jika definisi kolom mengizinkan NULL.

Misalnya, jika definisi tabel berisi empat kolom CHAR yang dapat dibatalkan, dan catatan berisi nilai-nilai`apple, orange, banana, mango`, perintah COPY dapat memuat dan mengisi catatan yang hanya berisi nilai-nilai. `apple, orange` Nilai CHAR yang hilang akan dimuat sebagai nilai NULL.

ABAIKAN BLANKLINES

Mengabaikan baris kosong yang hanya berisi umpan baris dalam file data dan tidak mencoba memuatnya.

IGNOREHEADER [AS] number_rows

Memperlakukan number_rows yang ditentukan sebagai header file dan tidak memuatnya. Gunakan IGNOREHEADER untuk melewati header file di semua file dalam pemuatan paralel.

NULL SEBAGAI 'null_string'

Memuat bidang yang cocok dengan null_string sebagai NULL, di mana null_ string dapat berupa string apa pun. Jika data Anda menyertakan terminator null, juga disebut sebagai NUL (UTF-8 0000) atau biner nol (0x000), COPY memperlakukannya sebagai karakter lainnya.

Misalnya, rekaman yang berisi '1' || NUL || '2' disalin sebagai string dengan panjang 3 byte. Jika bidang hanya berisi NUL, Anda dapat menggunakan NULL AS untuk mengganti terminator null dengan NULL dengan menentukan `'\0'` atau `'\000'` —misalnya, `NULL AS '\0'` atau. `NULL AS '\000'` Jika bidang berisi string yang diakhiri dengan NUL dan NULL AS ditentukan, string dimasukkan dengan NUL di akhir. Jangan gunakan '\ n' (baris baru) untuk nilai n ull_string. Amazon Redshift menyimpan '\ n' untuk digunakan sebagai pembatas baris. Null_string default adalah '. `'\N`

> **Note**
>
> Jika Anda mencoba memuat nol ke kolom yang didefinisikan sebagai NOT NULL, perintah COPY akan gagal.

HAPUS KUTIPAN

Hapus tanda kutip yang mengelilingi string pada data yang akan masuk. Semua karakter dalam tanda kutip, termasuk delimiter, dipertahankan. Jika string memiliki tanda kutip tunggal atau ganda awal tetapi tidak ada tanda akhir yang sesuai, perintah COPY gagal memuat baris itu dan mengembalikan kesalahan. Tabel berikut menunjukkan beberapa contoh sederhana dari string yang berisi tanda kutip dan nilai yang dimuat yang dihasilkan.

String Input	Nilai Dimuat dengan Opsi REMOVEQUOTES
“Pembatas adalah karakter pipa (|)”	Pembatas adalah karakter pipa (|)
'Hitam'	Hitam
“Putih”	Putih
Biru'	Biru'
'Biru	Nilai tidak dimuat: kondisi kesalahan
“Biru	Nilai tidak dimuat: kondisi kesalahan
" 'Hitam' "	" Hitam "
' '	<white space>

ROUNDEC

Membulatkan nilai numerik ketika skala nilai input lebih besar dari skala kolom. Secara default, COPY memotong nilai bila diperlukan agar sesuai dengan skala kolom. Misalnya, jika nilai `20.259` dimuat ke kolom DECIMAL (8,2), COPY memotong nilai secara `20.25` default. Jika ROUNDEC ditentukan, COPY membulatkan nilai ke`20.26`. Perintah INSERT selalu membulatkan nilai bila diperlukan untuk mencocokkan skala kolom, sehingga perintah COPY dengan parameter ROUNDEC berperilaku sama dengan perintah INSERT.

TIMEFORMAT [AS] {'timeformat_string' | 'otomatis' | 'epochsecs' | 'epochmillisecs'}

Menentukan format waktu. Jika tidak ada TIMEFORMAT yang ditentukan, format defaultnya adalah `YYYY-MM-DD HH:MI:SS` untuk kolom TIMESTAMP atau `YYYY-MM-DD HH:MI:SSOF` untuk kolom TIMESTAMPTZ, dimana offset dari Coordin `OF` ated Universal Time (UTC). Anda tidak dapat menyertakan penentu zona waktu di time format_string. Untuk memuat data TIMESTAMPTZ yang dalam format yang berbeda dari format default, tentukan 'otomatis'; untuk informasi lebih lanjut, lihat. Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT Untuk informasi selengkapnya tentang timeformat_string, lihat. String DATEFORMAT dan TIMEFORMAT

`'auto'`Argumen mengenali beberapa format yang tidak didukung saat menggunakan string DATEFORMAT dan TIMEFORMAT. Jika perintah COPY tidak mengenali format nilai tanggal atau waktu Anda, atau jika nilai tanggal dan waktu Anda menggunakan format yang berbeda satu sama lain, gunakan `'auto'` argumen dengan parameter DATEFORMAT atau TIMEFORMAT. Untuk informasi selengkapnya, lihat Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT.

Jika data sumber Anda direpresentasikan sebagai zaman zaman, yaitu jumlah detik atau milidetik sejak 1 Januari 1970, 00:00:00 UTC, tentukan atau. `'epochsecs'` `'epochmillisecs'`

Kata `'epochmillisecs'` kunci`'auto'`,`'epochsecs'`, dan peka huruf besar.

Kata kunci AS adalah opsional.

TRIMBLANK

Menghapus karakter spasi putih di belakang dari string VARCHAR. Parameter ini hanya berlaku untuk kolom dengan tipe data VARCHAR.

TRUNCATECOLUMN

Potong data dalam kolom sesuai dengan jumlah karakter sehingga cocok dengan spesifikasi kolom. Berlaku hanya pada kolom dengan tipe data VARCHAR atau CHAR, dan baris berukuran 4 MB atau kurang.

Operasi beban data

Kelola perilaku default operasi beban untuk pemecahan masalah atau untuk mengurangi waktu muat dengan menentukan parameter berikut.

- COMPROWS
- COMPUPDATE
- IGNOREALLERRORS
- MAXERROR
- NOLOAD
- STATUPDATE

Parameter

COMPROWS numrows

Menentukan jumlah baris yang akan digunakan sebagai ukuran sampel untuk analisis kompresi. Analisis dijalankan pada baris dari setiap irisan data. Misalnya, jika Anda menentukan `COMPROWS 1000000` (1.000.000) dan sistem berisi empat irisan total, tidak lebih dari 250.000 baris untuk setiap irisan dibaca dan dianalisis.

Jika COMPROWS tidak ditentukan, ukuran sampel default menjadi 100.000 untuk setiap irisan. Nilai COMPROWS yang lebih rendah dari default 100.000 baris untuk setiap irisan secara otomatis ditingkatkan ke nilai default. Namun, kompresi otomatis tidak akan terjadi jika jumlah data yang dimuat tidak cukup untuk menghasilkan sampel yang berarti.

Jika nomor COMPROWS lebih besar dari jumlah baris dalam file input, perintah COPY masih melanjutkan dan menjalankan analisis kompresi pada semua baris yang tersedia. Rentang yang diterima untuk argumen ini adalah angka antara 1000 dan 2147483647 (2.147.483.647).

COMPUPDATE [PRESET | {AKTIF | BENAR} | {OFF | FALSE}],

Mengontrol apakah pengkodean kompresi diterapkan secara otomatis selama COPY.

Ketika COMPUPDATE adalah PRESET, perintah COPY memilih pengkodean kompresi untuk setiap kolom jika tabel target kosong; bahkan jika kolom sudah memiliki pengkodean selain RAW. Pengkodean kolom yang ditentukan saat ini dapat diganti. Pengkodean untuk setiap kolom didasarkan pada tipe data kolom. Tidak ada data yang diambil sampelnya. Amazon Redshift secara otomatis menetapkan pengkodean kompresi sebagai berikut:

- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, atau DOUBLE PRECISION diberi kompresi RAW.
- Kolom yang didefinisikan sebagai SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai CHAR atau VARCHAR diberi kompresi LZO.

Ketika COMPUPDATE dihilangkan, perintah COPY memilih pengkodean kompresi untuk setiap kolom hanya jika tabel target kosong dan Anda belum menentukan pengkodean (selain RAW) untuk salah satu kolom. Pengkodean untuk setiap kolom ditentukan oleh Amazon Redshift. Tidak ada data yang diambil sampelnya.

Ketika COMPUPDATE ON (atau TRUE), atau COMPUPDATE ditentukan tanpa opsi, perintah COPY menerapkan kompresi otomatis jika tabel kosong; bahkan jika kolom tabel sudah memiliki pengkodean selain RAW. Pengkodean kolom yang ditentukan saat ini dapat diganti. Pengkodean untuk setiap kolom didasarkan pada analisis data sampel. Untuk informasi selengkapnya, lihat Memuat tabel dengan kompresi otomatis.

Ketika COMPUPDATE OFF (atau FALSE), kompresi otomatis dinonaktifkan. Pengkodean kolom tidak berubah.

Untuk informasi tentang tabel sistem untuk menganalisis kompresi, lihatSTL_ANALISE_KOMPRESI.

ABAIKKAN ALLERRORS

Anda dapat menentukan opsi ini untuk mengabaikan semua kesalahan yang terjadi selama operasi pemuatan.

Anda tidak dapat menentukan opsi IGNOREALLERRORS jika Anda menentukan opsi MAXERROR. Anda tidak dapat menentukan opsi IGNOREALLERRORS untuk format kolom termasuk ORC dan Parquet.

MAXERROR [AS] error_count

Jika beban mengembalikan jumlah kesalahan error_count atau lebih besar, beban gagal. Jika beban mengembalikan lebih sedikit kesalahan, ia melanjutkan dan mengembalikan pesan INFO yang menyatakan jumlah baris yang tidak dapat dimuat. Gunakan parameter ini untuk memungkinkan pemuatan berlanjut ketika baris tertentu gagal dimuat ke dalam tabel karena kesalahan pemformatan atau inkonsistensi lain dalam data.

Tetapkan nilai ini ke `0` atau `1` jika Anda ingin pemuatan gagal segera setelah kesalahan pertama terjadi. Kata kunci AS adalah opsional. Nilai default MAXERROR adalah `0` dan batasnya adalah`100000`.

Jumlah kesalahan aktual yang dilaporkan mungkin lebih besar dari MAXERROR yang ditentukan karena sifat paralel Amazon Redshift. Jika ada node di cluster Amazon Redshift yang mendeteksi bahwa MAXERROR telah terlampaui, setiap node melaporkan semua kesalahan yang ditemuinya.

NOLOAD

Memeriksa validitas file data tanpa benar-benar memuat data. Gunakan parameter NOLOAD untuk memastikan bahwa file data Anda dimuat tanpa kesalahan sebelum menjalankan pemuatan data yang sebenarnya. Menjalankan COPY dengan parameter NOLOAD jauh lebih cepat daripada memuat data karena hanya mengurai file.

STATUPDATE [{ON | TRUE} | {OFF | FALSE}]

Mengatur komputasi otomatis dan penyegaran statistik pengoptimal di akhir perintah COPY yang berhasil. Secara default, jika parameter STATUPDATE tidak digunakan, statistik diperbarui secara otomatis jika tabel awalnya kosong.

Setiap kali menelan data ke dalam tabel yang tidak kosong secara signifikan mengubah ukuran tabel, kami sarankan memperbarui statistik baik dengan menjalankan MENGANALISA perintah atau dengan menggunakan argumen STATUPDATE ON.

Dengan STATUPDATE AKTIF (atau TRUE), statistik diperbarui secara otomatis terlepas dari apakah tabel awalnya kosong. Jika STATUPDATE digunakan, pengguna saat ini harus menjadi pemilik tabel atau superuser. Jika STATUPDATE tidak ditentukan, hanya izin INSERT yang diperlukan.

Dengan STATUPDATE OFF (atau FALSE), statistik tidak pernah diperbarui.

Untuk informasi tambahan, lihat Menganalisis tabel.

Daftar parameter abjad

Daftar berikut menyediakan tautan ke setiap deskripsi parameter perintah COPY, diurutkan berdasarkan abjad.

- ACCEPTANYDATE
- ACCEPTINVCHARS
- the section called “AKSES_KEY_ID, KUNCI AKSES_RAHASIA”
- AVRO
- BLANKSASNULL
- BZIP2
- COMPROWS
- COMPUPDATE
- the section called “KREDENSIBILITAS”
- CSV
- DATEFORMAT
- DELIMITER
- EMPTYASNULL
- ENCODING
- ENCRYPTED
- ESCAPE
- EXPLICIT_IDS
- FILLRECORD
- FIXEDWIDTH
- FORMAT
- FROM
- GZIP
- the section called “IAM_ROLE”

- IGNOREALLERRORS
- IGNOREBLANKLINES
- IGNOREHEADER
- JSON format for COPY
- LZOP
- MANIFEST
- MASTER_SYMMETRIC_KEY
- MAXERROR
- NOLOAD
- NULL AS
- READRATIO
- REGION
- REMOVEQUOTES
- ROUNDEC
- the section called “SESSION_TOKEN”
- SHAPEFILE
- SSH
- STATUPDATE
- TIMEFORMAT
- TRIMBLANKS
- TRUNCATECOLUMNS
- ZSTD

Catatan penggunaan

Topik

- Izin untuk mengakses lainnya AWS Sumber daya
- Menggunakan COPY dengan alias titik akses Amazon S3
- Memuat data multibyte dari Amazon S3

- Memuat kolom tipe data GEOMETRY atau GEOGRAPHY
- Memuat tipe data HLLSKETCH
- Memuat kolom tipe data VARBYTE
- Kesalahan saat membaca banyak file
- SALIN dari format JSON
- SALIN dari format data kolumnar
- String DATEFORMAT dan TIMEFORMAT
- Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT

Izin untuk mengakses lainnya AWS Sumber daya

Untuk memindahkan data antara cluster Anda dan AWS sumber daya lain, seperti Amazon S3, Amazon DynamoDB, Amazon EMR, atau Amazon EC2, cluster Anda harus memiliki izin untuk mengakses sumber daya dan melakukan tindakan yang diperlukan. Misalnya, untuk memuat data dari Amazon S3, COPY harus memiliki akses LIST ke bucket dan akses GET untuk objek bucket. Untuk informasi tentang izin minimum, lihatIzin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY.

Untuk mendapatkan otorisasi untuk mengakses sumber daya, cluster Anda harus diautentikasi. Anda dapat memilih salah satu metode otentikasi berikut:

- Role-based kontrol aksesUntuk kontrol akses berbasis peran, Anda menentukan peran AWS Identity and Access Management (IAM) yang digunakan cluster Anda untuk otentikasi dan otorisasi. Untuk melindungi AWS kredensi dan data sensitif Anda, kami sangat menyarankan untuk menggunakan otentikasi berbasis peran.
- Key-based kontrol akses— Untuk kontrol akses berbasis kunci, Anda memberikan kreden AWS SIAL akses (ID kunci akses dan kunci akses rahasia) untuk pengguna sebagai teks biasa.

Role-based kontrol akses

Dengan kontrol akses berbasis peran, cluster Anda sementara mengambil peran IAM atas nama Anda. Kemudian, berdasarkan otorisasi yang diberikan kepada peran, cluster Anda dapat mengakses AWS sumber daya yang diperlukan.

Membuat peran IAM mirip dengan memberikan izin kepada pengguna, karena itu adalah AWS identitas dengan kebijakan izin yang menentukan apa yang dapat dan tidak dapat dilakukan identitas.

AWS Namun, alih-alih dikaitkan secara unik dengan satu pengguna, peran dapat diasumsikan oleh entitas mana pun yang membutuhkannya. Selain itu, peran tidak memiliki kredenSIAL (kata sandi atau kunci akses) yang terkait dengannya. Sebaliknya, jika peran dikaitkan dengan cluster, kunci akses dibuat secara dinamis dan diberikan ke cluster.

Sebaiknya gunakan kontrol akses berbasis peran karena memberikan kontrol akses yang lebih aman dan berbutir halus terhadap AWS sumber daya dan data pengguna yang sensitif, selain melindungi kredensibilitas Anda. AWS

Role-based otentikasi memberikan manfaat berikut:

- Anda dapat menggunakan alat IAM AWS standar untuk menentukan peran IAM dan mengaitkan peran dengan beberapa cluster. Saat Anda mengubah kebijakan akses untuk peran, perubahan diterapkan secara otomatis ke semua cluster yang menggunakan peran tersebut.
- Anda dapat menentukan kebijakan IAM berbutir halus yang memberikan izin untuk cluster tertentu dan pengguna database untuk mengakses AWS sumber daya dan tindakan tertentu.
- Cluster Anda memperoleh kredentif sesi sementara pada waktu berjalan dan menyegarkan kredenSIAL sesuai kebutuhan hingga operasi selesai. Jika Anda menggunakan kredenSIAL sementara berbasis kunci, operasi gagal jika kredenSIAL sementara kedaluwarsa sebelum selesai.
- ID kunci akses dan ID kunci akses rahasia Anda tidak disimpan atau ditransmisikan dalam kode SQL Anda.

Untuk menggunakan kontrol akses berbasis peran, Anda harus terlebih dahulu membuat peran IAM menggunakan jenis peran layanan Amazon Redshift, lalu melampirkan peran tersebut ke cluster Anda. Peran harus memiliki, minimal, izin yang tercantum dilzin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY. Untuk langkah-langkah membuat peran IAM dan melampirkannya ke cluster Anda, lihat Mengotor isasi Amazon Redshift untuk Mengakses Layanan Lain A AWS tas Nama Anda di Panduan Manajemen Amazon Redshift.

Anda dapat menambahkan peran ke cluster atau melihat peran yang terkait dengan cluster menggunakan Amazon Redshift Management Console, CLI, atau API. Untuk informasi selengkapnya, lihat Mengaitkan Peran IAM Dengan Cluster di Panduan Manajemen Amazon Redshift.

Saat Anda membuat peran IAM, IAM mengembalikan Nama Sumber Daya Amazon (ARN) untuk peran tersebut. Untuk menentukan peran IAM, berikan peran ARN dengan the section called “Menggunakan IAM_ROLE” parameter atau the section called “Menggunakan KREDENSIAL” parameter.

Misalnya, peran berikut dilampirkan ke cluster.

```
"IamRoleArn": "arn:aws:iam::0123456789012:role/MyRedshiftRole"
```

Contoh perintah COPY berikut menggunakan parameter IAM_ROLE dengan ARN pada contoh sebelumnya untuk otentikasi dan akses ke Amazon S3.

```
copy customer from 's3://amzn-s3-demo-bucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Contoh perintah COPY berikut menggunakan parameter CREDENTIALS untuk menentukan peran IAM.

```
copy customer from 's3://amzn-s3-demo-bucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Selain itu, superuser dapat memberikan hak istimewa ASSUMEROLE kepada pengguna database dan grup untuk menyediakan akses ke peran untuk operasi COPY. Untuk informasi, lihat HIBAH.

Key-based kontrol akses

Dengan kontrol akses berbasis kunci, Anda memberikan ID kunci akses dan kunci akses rahasia untuk pengguna IAM yang berwenang mengakses AWS sumber daya yang berisi data. Anda dapat menggunakan the section called “Menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY” parameter bersama-sama atau the section called “Menggunakan KREDENSIAL” parameter.

> **Note**
>
> Kami sangat menyarankan menggunakan peran IAM untuk otentikasi alih-alih memberikan ID kunci akses teks biasa dan kunci akses rahasia. Jika Anda memilih kontrol akses berbasis kunci, jangan pernah menggunakan kredentif AWS akun (root) Anda. Selalu buat pengguna IAM dan berikan ID kunci akses dan kunci akses rahasia pengguna tersebut. Untuk langkah-langkah membuat pengguna IAM, lihat Membuat Pengguna IAM di Akun Anda AWS .

Untuk mengotentikasi menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY, ganti *<access-key-id>* dan *<secret-access-key>* dengan ID kunci akses pengguna yang berwenang dan kunci akses rahasia lengkap seperti yang ditunjukkan berikut.

```
ACCESS_KEY_ID '<access-key-id>'
SECRET_ACCESS_KEY '<secret-access-key>';
```

Untuk mengotentikasi menggunakan parameter CREDENTIALS, ganti *<access-key-id>* dan *<secret-access-key>* dengan ID kunci akses pengguna yang berwenang dan kunci akses rahasia lengkap seperti yang ditunjukkan berikut ini.

```
CREDENTIALS
'aws_access_key_id=<access-key-id>;aws_secret_access_key=<secret-access-key>';
```

Pengguna IAM harus memiliki, minimal, izin yang tercantum diIzin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY.

Kredensial keamanan sementara

Jika Anda menggunakan kontrol akses berbasis kunci, Anda dapat membatasi lebih lanjut akses yang dimiliki pengguna ke data Anda dengan menggunakan kredentif keamanan sementara. Role-basedotentikasi secara otomatis menggunakan kredenSIAL sementara.

> **Note**
>
> Kami sangat menyarankan menggunakan al role-based access control ih-alih membuat kredenSIAL sementara dan memberikan ID kunci akses dan kunci akses rahasia sebagai teks biasa. Role-based kontrol akses secara otomatis menggunakan kredenSIAL sementara.

KredenSIAL keamanan sementara memberikan keamanan yang ditingkatkan karena memiliki rentang hidup yang pendek dan tidak dapat digunakan kembali setelah kedaluwarsa. ID kunci akses dan kunci akses rahasia yang dihasilkan dengan token tidak dapat digunakan tanpa token, dan pengguna yang memiliki kredenSIAL keamanan sementara ini dapat mengakses sumber daya Anda hanya sampai kredenSIAL kedaluwarsa.

Untuk memberi pengguna akses sementara ke sumber daya Anda, Anda memang AWS Security Token Service gil operasi API (AWS STS). Operasi AWS STS API mengembalikan kredentif keamanan sementara yang terdiri dari token keamanan, ID kunci akses, dan kunci akses rahasia. Anda menerbitkan kredentif keamanan sementara kepada pengguna yang memerlukan akses sementara ke sumber daya Anda. Pengguna ini bisa menjadi pengguna IAM yang sudah ada, atau mereka bisa bukan AWS pengguna. Untuk informasi selengkapnya tentang membuat kredenSIAL

keamanan sementara, lihat Menggunakan KredenSIAL Keamanan Sementara di Panduan Pengguna IAM.

Anda dapat menggunakan the section called “Menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY” parameter bersama dengan the section called “SESSION_TOKEN” parameter atau the section called “Menggunakan KREDENSIAL” parameter. Anda juga harus memberikan ID kunci akses dan kunci akses rahasia yang disediakan dengan token.

Untuk mengotentikasi menggunakan ACCESS_KEY_ID, SECRET_ACCESS_KEY, dan SESSION_TOKEN, ganti,, dan seperti yang ditunjukkan berikut. *<temporary-access-key-id>* *<temporary-secret-access-key>* *<temporary-token>*

```
ACCESS_KEY_ID '<temporary-access-key-id>'
SECRET_ACCESS_KEY '<temporary-secret-access-key>'
SESSION_TOKEN '<temporary-token>';
```

Untuk mengotentikasi menggunakan CREDENTIALS, sertakan `session_token=`*`<temporary-token>`* dalam string kredenSIAL seperti yang ditunjukkan berikut.

```
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;session_token=<temporary-token>';
```

Contoh berikut menunjukkan perintah COPY dengan kredenSIAL keamanan sementara.

```
copy table-name
from 's3://objectpath'
access_key_id '<temporary-access-key-id>'
secret_access_key '<temporary-secret-access-key>'
session_token '<temporary-token>';
```

Contoh berikut memuat tabel LISTING dengan kredenSIAL sementara dan enkripsi file.

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listings_pipe.txt'
access_key_id '<temporary-access-key-id>'
secret_access_key '<temporary-secret-access-key>'
session_token '<temporary-token>'
master_symmetric_key '<root-key>'
```

```
encrypted;
```

Contoh berikut memuat tabel LISTING menggunakan parameter CREDENTIALS dengan kredenSIAL sementara dan enkripsi file.

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listings_pipe.txt'
credentials
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;session_token=<temporary-token>;master_symmetric_key=<root-key>'
encrypted;
```

> **Important**
>
> KredenSIAL keamanan sementara harus valid untuk seluruh durasi operasi COPY atau UNLOAD. Jika kredenSIAL keamanan sementara kedaluwarsa selama operasi, perintah gagal dan transaksi dibatalkan. Misalnya, jika kredentif keamanan sementara kedaluwarsa setelah 15 menit dan operasi COPY memerlukan satu jam, operasi COPY gagal sebelum selesai. Jika Anda menggunakan akses berbasis peran, kredentif keamanan sementara akan diperbarui secara otomatis hingga operasi selesai.

Izin IAM untuk COPY, UNLOAD, dan CREATE LIBRARY

Peran IAM atau pengguna yang direferensikan oleh parameter CREDENTIALS harus memiliki, minimal, izin berikut:

- Untuk SALIN dari Amazon S3, izin untuk DAFTAR bucket Amazon S3 dan DAPATKAN objek Amazon S3 yang sedang dimuat, dan file manifes, jika digunakan.
- Untuk COPY dari Amazon S3, Amazon EMR, dan host jarak jauh (SSH) dengan JSON-formatted data, izin untuk LIST dan GET file JSONPaths di Amazon S3, jika digunakan.
- Untuk SALIN dari DynamoDB, izin untuk MEMINDAI dan MENDESKRIPSI tabel DynamoDB yang sedang dimuat.
- Untuk COPY dari cluster Amazon EMR, izin untuk `ListInstances` tindakan pada cluster Amazon EMR.
- Untuk UNLOAD ke Amazon S3, izin GET, LIST, dan PUT untuk bucket Amazon S3 tempat file data dibongkar.

- Untuk MEMBUAT PERPUSTAKAAN dari Amazon S3, izin untuk DAFTAR bucket Amazon S3 dan DAPATKAN objek Amazon S3 yang diimpor.

> **Note**
>
> Jika Anda menerima pesan galat`S3ServiceException: Access Denied`, saat menjalankan perintah COPY, UNLOAD, atau CREATE LIBRARY, cluster Anda tidak memiliki izin akses yang tepat untuk Amazon S3.

Anda dapat mengelola izin IAM dengan melampirkan kebijakan IAM ke peran IAM yang dilampirkan ke cluster Anda, ke pengguna, atau ke grup tempat pengguna Anda berada. Misalnya, kebijakan terkel `AmazonS3ReadOnlyAccess` ola memberikan izin LIST dan GET ke sumber daya Amazon S3. Untuk informasi selengkapnya tentang kebijakan IAM, lihat M engelola Kebijakan IAM di Panduan Pengguna IAM.

Menggunakan COPY dengan alias titik akses Amazon S3

COPY mendukung alias titik akses Amazon S3. Untuk informasi selengkapnya, lihat Menggunakan alias gaya bucket untuk titik akses Anda di Panduan Pengguna Layanan Penyimpanan S ederhana Amazon.

Memuat data multibyte dari Amazon S3

Jika data Anda menyertakan karakter multibyte non-ASCII (seperti karakter Mandarin atau Cyrillic), Anda harus memuat data ke kolom VARCHAR. Tipe data VARCHAR mendukung UTF-8 karakter empat byte, tetapi tipe data CHAR hanya menerima karakter ASCII byte tunggal. Anda tidak dapat memuat karakter lima byte atau lebih lama ke tabel Amazon Redshift. Untuk informasi selengkapnya, lihat Karakter multibyte.

Memuat kolom tipe data GEOMETRY atau GEOGRAPHY

Anda dapat MENYALIN ke `GEOMETRY` atau `GEOGRAPHY` kolom dari data dalam file teks yang dibatasi karakter, seperti file CSV. Data harus dalam bentuk heksadesimal dari format biner terkenal (baik WKB atau EWKB) atau format teks terkenal (baik WKT atau EWKT) dan sesuai dengan ukuran maksimum satu baris input ke perintah COPY. Untuk informasi selengkapnya, lihat MENYALIN.

Untuk informasi tentang cara memuat dari shapefile, lihat. Memuat shapefile ke Amazon Redshift

Untuk informasi selengkapnya tentang tipe GEOGRAPHY data GEOMETRY atau, lihatMengkueri data spasial di Amazon Redshift.

Memuat tipe data HLLSKETCH

Anda dapat menyalin sketsa HLL hanya dalam format jarang atau padat yang didukung oleh Amazon Redshift. Untuk menggunakan perintah COPY pada HyperLogLog sketsa, gunakan format Base64 untuk HyperLogLog sketsa padat dan format JSON untuk sketsa jarang HyperLogLog . Untuk informasi selengkapnya, lihat HyperLogLog fungsi.

Contoh berikut mengimpor data dari file CSV ke dalam tabel menggunakan CREATE TABLE dan COPY. Pertama, contoh membuat tabel t1 menggunakan CREATE TABLE.

```
CREATE TABLE t1 (sketch hllsketch, a bigint);
```

Kemudian menggunakan COPY untuk mengimpor data dari file CSV ke dalam tabelt1.

```
COPY t1 FROM s3://amzn-s3-demo-bucket/unload/' IAM_ROLE
 'arn:aws:iam::0123456789012:role/MyRedshiftRole' NULL AS 'null' CSV;
```

Memuat kolom tipe data VARBYTE

Anda dapat memuat data dari file dalam format CSV, Parquet, dan ORC. Untuk CSV, data dimuat dari file dalam representasi heksadesimal dari data VARBYTE. Anda tidak dapat memuat data VARBYTE dengan FIXEDWIDTH opsi. REMOVEQUOTESOpsi ADDQUOTES atau COPY tidak didukung. Kolom VARBYTE tidak dapat digunakan sebagai kolom partisi.

Kesalahan saat membaca banyak file

Perintah COPY bersifat atom dan transaksional. Dengan kata lain, bahkan ketika perintah COPY membaca data dari beberapa file, seluruh proses diperlakukan sebagai satu transaksi. Jika COPY mengalami kesalahan saat membaca file, file akan dicoba lagi secara otomatis sampai waktu proses habis (lihatstatement_timeout) atau jika data tidak dapat diunduh dari Amazon S3 untuk jangka waktu yang lama (antara 15 dan 30 menit), memastikan bahwa setiap file dimuat hanya sekali. Jika perintah COPY gagal, seluruh transaksi dibatalkan dan semua perubahan dibatalkan. Untuk informasi selengkapnya tentang penanganan kesalahan beban, lihatMemecahkan masalah beban data.

Setelah perintah COPY berhasil dimulai, itu tidak gagal jika sesi berakhir, misalnya ketika klien terputus. Namun, jika perintah COPY berada dalam blok transaksi BEGIN... END yang tidak selesai

karena sesi berakhir, seluruh transaksi, termasuk COPY, akan diputar kembali. Untuk informasi lebih lanjut tentang transaksi, lihatMULAI.

SALIN dari format JSON

Struktur data JSON terdiri dari sekumpulan objek atau array. Ob jek JSON dimulai dan diakhiri dengan kurung kurung, dan berisi kumpulan pasangan nama-nilai yang tidak berurutan. Setiap nama dan nilai dipisahkan oleh titik dua, dan pasangan dipisahkan dengan koma. Nama adalah string dalam tanda kutip ganda. Karakter tanda kutip harus berupa tanda kutip sederhana (0x22), bukan tanda kutip miring atau “pintar”.

Sebuah array JSON dimulai dan diakhiri dengan tanda kurung, dan berisi kumpulan nilai yang dipisahkan oleh koma. Nilai dapat berupa string dalam tanda kutip ganda, angka, Boolean benar atau salah, null, objek JSON, atau array.

Objek dan array JSON dapat bersarang, memungkinkan struktur data hierarkis. Contoh berikut menunjukkan struktur data JSON dengan dua objek yang valid.

```
{
    "id": 1006410,
    "title": "Amazon Redshift Database Developer Guide"
}
{
    "id": 100540,
    "name": "Amazon Simple Storage Service User Guide"
}
```

Berikut ini menunjukkan data yang sama dengan dua array JSON.

```
[
    1006410,
    "Amazon Redshift Database Developer Guide"
]
[
    100540,
    "Amazon Simple Storage Service User Guide"
]
```

Opsi COPY untuk JSON

Anda dapat menentukan opsi berikut saat menggunakan COPY dengan data format JSON:

- `'auto'` — COPY secara otomatis memuat bidang dari file JSON.
- `'auto ignorecase'`— COPY secara otomatis memuat bidang dari file JSON sambil mengabaikan kasus nama bidang.
- `s3://jsonpaths_file`- COPY menggunakan file JSONPaths untuk mengurai data sumber JSON. File JSONPaths adalah file teks yang berisi objek JSON tunggal dengan nama yang `"jsonpaths"` dipasangkan dengan array ekspresi JSONPath. Jika namanya adalah string selain`"jsonpaths"`, COPY menggunakan `'auto'` argumen alih-alih menggunakan file JSONPaths.

Untuk contoh yang menunjukkan cara memuat data menggunakan`'auto'`,`'auto ignorecase'`, atau file JSONPaths, dan menggunakan objek atau array JSON, lihat. Salin dari contoh JSON

Opsi JSONPath

Dalam sintaks Amazon Redshift COPY, ekspresi JSONPath menentukan jalur eksplisit ke elemen nama tunggal dalam struktur data hierarkis JSON, menggunakan notasi tanda kurung atau notasi titik. Amazon Redshift tidak mendukung elemen JSONPath apa pun, seperti karakter wildcard atau ekspresi filter, yang mungkin berubah menjadi jalur ambigu atau beberapa elemen nama. Akibatnya, Amazon Redshift tidak dapat mengurai struktur data multi-level yang kompleks.

Berikut ini adalah contoh file JSONPaths dengan ekspresi JSONPath menggunakan notasi tanda kurung. Tanda dolar ($) mewakili struktur tingkat akar.

```
{
    "jsonpaths": [
       "$['id']",
       "$['store']['book']['title']",
 "$['location'][0]"
    ]
}
```

Dalam contoh sebelumnya, `$['location'][0]` referensi elemen pertama dalam array. JSON menggunakan pengindeksan array berbasis nol. Indeks array harus bilangan bulat positif (lebih besar dari atau sama dengan nol).

Contoh berikut menunjukkan file JSONPaths sebelumnya menggunakan notasi titik.

```
{
```

```
    "jsonpaths": [
       "$.id",
       "$.store.book.title",
 "$.location[0]"
    ]
}
```

Anda tidak dapat mencampur notasi tanda kurung dan notasi titik dalam `jsonpaths` array. Kurung dapat digunakan dalam notasi tanda kurung dan notasi titik untuk mereferensikan elemen array.

Saat menggunakan notasi titik, ekspresi JSONPath tidak dapat berisi karakter berikut:

- Tanda kutip lurus tunggal (')
- Periode, atau titik (.)
- Kurung ([]) kecuali digunakan untuk mereferensikan elemen array

Jika nilai dalam pasangan nama-nilai yang direferensikan oleh ekspresi JSONPath adalah objek atau array, seluruh objek atau array dimuat sebagai string, termasuk kurung kurung atau tanda kurung. Misalnya, misalkan data JSON Anda berisi objek berikut.

```
{
    "id": 0,
    "guid": "84512477-fa49-456b-b407-581d0d851c3c",
    "isActive": true,
    "tags": [
        "nisi",
        "culpa",
        "ad",
        "amet",
        "voluptate",
        "reprehenderit",
        "veniam"
    ],
    "friends": [
        {
            "id": 0,
            "name": "Martha Rivera"
        },
        {
            "id": 1,
            "name": "Renaldo"
```

```
        }
    ]
}
```

Ekspresi JSONPath `$['tags']` kemudian mengembalikan nilai berikut.

```
"["nisi","culpa","ad","amet","voluptate","reprehenderit","veniam"]"
```

Ekspresi JSONPath `$['friends'][1]` kemudian mengembalikan nilai berikut.

```
"{"id": 1,"name": "Renaldo"}"
```

Setiap ekspresi JSONPath dalam `jsonpaths` array sesuai dengan satu kolom di tabel target Amazon Redshift. Urutan elemen `jsonpaths` array harus sesuai dengan urutan kolom dalam tabel target atau daftar kolom, jika daftar kolom digunakan.

Untuk contoh yang menunjukkan cara memuat data menggunakan `'auto'` argumen atau file JSONPaths, dan menggunakan objek JSON atau array, lihat. Salin dari contoh JSON

Untuk informasi tentang cara menyalin beberapa file JSON, lihatMenggunakan manifes untuk menentukan file data.

Karakter melarikan diri di JSON

COPY di \n muat sebagai karakter baris baru dan memu \t at sebagai karakter tab. Untuk memuat garis miring terbalik, lepaskan dengan backslash (). \\

Misalnya, Anda memiliki JSON berikut dalam file bernama `escape.json` dalam buckets`3://amzn-s3-demo-bucket/json/`.

```
{
  "backslash": "This is a backslash: \\",
  "newline": "This sentence\n is on two lines.",
  "tab": "This sentence \t contains a tab."
}
```

Jalankan perintah berikut untuk membuat tabel ESCAPES dan memuat JSON.

```
create table escapes (backslash varchar(25), newline varchar(35), tab varchar(35));
```

```
copy escapes from 's3://amzn-s3-demo-bucket/json/escape.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as json 'auto';
```

Kueri tabel ESCAPES untuk melihat hasilnya.

```
select * from escapes;

       backslash        |      newline      |               tab
------------------------+-------------------+-----------------------------------
 This is a backslash: \ | This sentence     | This sentence    contains a tab.
                        :  is on two lines.
(1 row)
```

Kehilangan presisi numerik

Anda mungkin kehilangan presisi saat memuat angka dari file data dalam format JSON ke kolom yang didefinisikan sebagai tipe data numerik. Beberapa nilai float point tidak direpresentasikan secara tepat dalam sistem komputer. Akibatnya, data yang Anda salin dari file JSON mungkin tidak dibulatkan seperti yang Anda harapkan. Untuk menghindari hilangnya presisi, sebaiknya gunakan salah satu alternatif berikut:

- Mewakili angka sebagai string dengan melampirkan nilai dalam karakter kutipan ganda.
- Gunakan ROUNDEC untuk membulatkan angka alih-alih memotong.
- Alih-alih menggunakan file JSON atau Avro, gunakan file teks CSV, dibatasi karakter, atau lebar tetap.

SALIN dari format data kolumnar

COPY dapat memuat data dari Amazon S3 dalam format kolom berikut:

- ORC
- Parquet

Untuk contoh penggunaan COPY dari format data kolumnar, lihatContoh COPY.

COPY mendukung data berformat kolom dengan pertimbangan berikut:

- Bucket Amazon S3 harus berada di Wil AWS ayah yang sama dengan database Amazon Redshift.
- Untuk mengakses data Amazon S3 Anda melalui titik akhir VPC, atur akses menggunakan kebijakan IAM dan peran IAM seperti yang dijelaskan dalam Menggunakan Amazon Redshift Spectrum dengan Perutean VPC yang Ditingkatkan di Panduan Manajemen Amazon Redshift.
- COPY tidak secara otomatis menerapkan pengkodean kompresi.
- Hanya parameter COPY berikut yang didukung:
  - ACCEPTINVCHARS saat menyalin dari file ORC atau Parquet.
  - CATATAN PENGISIAN
  - DARI
  - IAM_PERAN
  - KREDENSIAL
  - PEMBARUAN STATUS
  - MANIFES
  - ID EKSPLISIT
- Jika COPY mengalami kesalahan saat memuat, perintah gagal. ACCEPTANYDATE dan MAXERROR tidak didukung untuk tipe data kolumnar.
- Pesan kesalahan dikirim ke klien SQL. Beberapa kesalahan dicatat di STL_LOAD_ERRORS dan STL_ERROR.
- COPY menyisipkan nilai ke kolom tabel target dalam urutan yang sama seperti kolom yang terjadi dalam file data kolumnar. Jumlah kolom dalam tabel target dan jumlah kolom dalam file data harus cocok.
- Jika file yang Anda tentukan untuk operasi COPY menyertakan salah satu ekstensi berikut, Amazon Redshift mendekompresi data tanpa perlu menambahkan parameter apa pun:
  - `.gz`
  - `.snappy`
  - `.bz2`
- COPY dari format file Parquet dan ORC menggunakan Redshift Spectrum dan akses bucket. Untuk menggunakan COPY untuk format ini, pastikan tidak ada kebijakan IAM yang memblokir penggunaan URL Amazon S3 yang telah ditandatangani sebelumnya. URL yang telah ditandatangani sebelumnya yang dihasilkan oleh Amazon Redshift berlaku selama 1 jam sehingga Amazon Redshift memiliki cukup waktu untuk memuat semua file dari bucket Amazon S3. URL unik yang ditandatangani dibuat untuk setiap file yang dipindai oleh COPY dari format data kolumnar. Untuk kebijakan bucket yang menyertakan `s3:signatureAge` tindakan, pastikan

untuk menyetel nilai ke setidaknya 3.600.000 milidetik. Untuk informasi selengkapnya, lihat Menggunakan Amazon Redshift Spectrum dengan perutean VPC yang disempurnakan.

- Parameter REGION tidak didukung dengan COPY dari format data kolumnar. Bahkan jika bucket Amazon S3 dan database Anda berada di tempat yang sama Wilayah AWS, Anda dapat mengalami kesalahan, seperti, argumen REGION tidak didukung untuk COPY berbasis PARQUET.
- COPY dari format kolumnar sekarang mendukung penskalaan konkurensi. Untuk mengaktifkan penskalaan konkurensi, lihat Mengon figurasi antrian penskalaan konkurensi.

String DATEFORMAT dan TIMEFORMAT

Perintah COPY menggunakan opsi DATEFORMAT dan TIMEFORMAT untuk mengurai nilai tanggal dan waktu dalam data sumber Anda. DATEFORMAT dan TIMEFORMAT adalah string yang diformat yang harus sesuai dengan format nilai tanggal dan waktu data sumber Anda. Misalnya, perintah COPY memuat data sumber dengan nilai tanggal `Jan-01-1999` harus menyertakan string DATEFORMAT berikut:

```
COPY ...
            DATEFORMAT AS 'MON-DD-YYYY'
```

Untuk informasi selengkapnya tentang mengelola konversi data COPY, lihat Parameter konversi data.

String DATEFORMAT dan TIMEFORMAT dapat berisi pemisah datetime (seperti '-', '/', atau ':'), serta format datepart dan timepart dalam tabel berikut.

> **Note**
>
> Jika Anda tidak dapat mencocokkan format nilai tanggal atau waktu dengan dateparts dan timeparts berikut, atau jika Anda memiliki nilai tanggal dan waktu yang menggunakan format yang berbeda satu sama lain, gunakan `'auto'` argumen dengan parameter DATEFORMAT atau TIMEFORMAT. `'auto'`Argumen mengenali beberapa format yang tidak didukung saat menggunakan string DATEFORMAT atau TIMEFORMAT. Untuk informasi selengkapnya, lihat Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT.

Part tanggal atau bagian waktu	Arti
YY	Tahun tanpa abad
YYYY	Tahun dengan abad
MM	Bulan sebagai angka
MON	Bulan sebagai nama (nama singkatan atau nama lengkap)
DD	Hari dalam sebulan sebagai angka
HH atau HH24	Jam (jam 24 jam) **Note** Dalam string format DATETIME untuk fungsi SQL, HH sama dengan HH12. Namun, dalam string DATEFORMAT dan TIMEFORMAT untuk COPY, HH sama dengan HH24.
HH12	Jam (jam 12 jam)
MI	Menit
SS	Detik
Pagi atau Sore	Indikator meridian (untuk jam 12 jam)

Format tanggal default adalah YYYY-MM-DD. Stempel waktu default tanpa format zona waktu (TIMESTAMP) adalah. YYYY-MM-DD HH:MI:SS Stempel waktu default dengan format zona waktu (TIMESTAMPTZ) adalah YYYY-MM-DD HH:MI:SSOF, di mana OF adalah offset dari UTC (misalnya, - 8:00. Anda tidak dapat menyertakan penentu zona waktu (TZ, tz, atau OF) dalam timeformat_string. Bidang detik (SS) juga mendukung detik fraksional hingga tingkat detail mikrodetik. Untuk memuat data TIMESTAMPTZ yang dalam format yang berbeda dari format default, tentukan 'auto'.

Berikut ini adalah beberapa contoh tanggal atau waktu yang dapat Anda temui di data sumber Anda, dan string DATEFORMAT atau TIMEFORMAT yang sesuai untuk mereka.

Contoh tanggal atau waktu data sumber	Sintaks DATEFORMAT atau TIMEFORMAT
03/31/2003	DATEFORMAT SEBAGAI ' MM/DD/YYYY
31 Maret 2003	FORMAT TANGGAL SEBAGAI 'MON DD, YYYY'
03.31.2003 18:45:05 03.31.2003 18:45:05.123 456	FORMAT WAKTU SEBAGAI ' MM.DD.YYYY HH:MI:SS

Contoh

Untuk contoh penggunaan TIMEFORMAT, lihatMemuat stempel waktu atau stempel tanggal.

Menggunakan pengenalan otomatis dengan DATEFORMAT dan TIMEFORMAT

Jika Anda menentukan `'auto'` sebagai argumen untuk parameter DATEFORMAT atau TIMEFORMAT, Amazon Redshift akan secara otomatis mengenali dan mengonversi format tanggal atau format waktu dalam data sumber Anda. Bagian berikut menunjukkan satu contoh.

```
copy favoritemovies from 'dynamodb://ProductCatalog'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
dateformat 'auto';
```

Ketika digunakan dengan `'auto'` argumen untuk DATEFORMAT dan TIMEFORMAT, COPY mengenali dan mengonversi format tanggal dan waktu yang tercantum dalam tabel di. String DATEFORMAT dan TIMEFORMAT Selain itu, `'auto'` argumen mengenali format berikut yang tidak didukung saat menggunakan string DATEFORMAT dan TIMEFORMAT.

Format	Contoh String Input yang Valid
ISO 8601	2019-02-11 T05:09:12.195Z

Format	Contoh String Input yang Valid
Julian	J2451187
SM	Jan-08-95 SM
YYYYMMDD HHMISS	19960108 404809
YYMMDD HHMISS	960108 040809
YYYY.DDD	1996.008
YYYY-MM-DD HH:MI:SS.SSS	1996-01-08 04:05:06.789
DD Senin HH:MI:SS YYYY TZ	17 Des 07:37:16 1997 PST
MM/DD/YYYY HH:MI:SS.SS TZ	12/17/1997 07:37:16.00 PST
YYYY-MM-DD HH:MI:SS+/-TZ	1997-12-17 07:37:16-08
DD.MM.YYYY HH:MI:SS TZ	12.17.1997 07:37:16.00 PST

Pengenalan otomatis tidak mendukung epochsecs dan epochmillisecs.

Untuk menguji apakah nilai tanggal atau stempel waktu akan dikonversi secara otomatis, gunakan fungsi CAST untuk mencoba mengonversi string ke nilai tanggal atau stempel waktu. Misalnya, perintah berikut menguji nilai stempel waktu: `'J2345678 04:05:06.789'`

```
create table formattest (test char(21));
insert into formattest values('J2345678 04:05:06.789');
select test, cast(test as timestamp) as timestamp, cast(test as date) as date from
 formattest;

        test          |      timestamp      | date
----------------------+---------------------+------------
```

```
J2345678 04:05:06.789   1710-02-23 04:05:06 1710-02-23
```

Jika data sumber untuk kolom DATE menyertakan informasi waktu, komponen waktu dipotong. Jika data sumber untuk kolom TIMESTAMP menghilangkan informasi waktu, 00:00:00 digunakan untuk komponen waktu.

Contoh COPY

> **Note**
>
> Contoh-contoh ini berisi jeda baris untuk keterbacaan. Jangan sertakan jeda baris atau spasi dalam string kredensi als-args Anda.

Topik

- Muat FAVORITEMOVIES dari tabel DynamoDB
- Muat LISTING dari bucket Amazon S3
- Muat LISTING dari cluster Amazon EMR
- Menggunakan manifes untuk menentukan file data
- Muat LISTING dari file yang dibatasi pipa (pembatas default)
- Muat LISTING menggunakan data kolumnar dalam format Parket
- Muat LISTING menggunakan data kolumnar dalam format ORC
- Muat EVENT dengan opsi
- Muat VENUE dari file data lebar tetap
- Muat KATEGORI dari file CSV
- Muat VENUE dengan nilai eksplisit untuk kolom IDENTITY
- Muat WAKTU dari file GZIP yang dibatasi pipa
- Memuat stempel waktu atau stempel tanggal
- Muat data dari file dengan nilai default
- SALIN data dengan opsi ESCAPE
- Salin dari contoh JSON
- Salin dari contoh Avro

- Mempersiapkan file untuk COPY dengan opsi ESCAPE
- Memuat shapefile ke Amazon Redshift
- Perintah COPY dengan opsi NOLOAD
- Perintah COPY dengan pembatas multibyte dan opsi ENCODING

Muat FAVORITEMOVIES dari tabel DynamoDB

SDK AWS menyertakan contoh sederhana untuk membuat tabel DynamoDB yang disebut Movies. (Untuk contoh ini, lihat Mem ulai dengan DynamoDB.) Contoh berikut memuat tabel Amazon Redshift MOVIES dengan data dari tabel DynamoDB. Tabel Amazon Redshift harus sudah ada di database.

```
copy favoritemovies from 'dynamodb://Movies'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
readratio 50;
```

Muat LISTING dari bucket Amazon S3

Contoh berikut memuat LISTING dari bucket Amazon S3. Perintah COPY memuat semua file dalam `/data/listing/` folder.

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listing/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Muat LISTING dari cluster Amazon EMR

Contoh berikut memuat tabel SALES dengan data yang dibatasi tab dari file terkompresi lzop di cluster Amazon EMR. COPY memuat setiap file dalam `myoutput/` folder yang dimulai dengan`part-`.

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/part-*'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '\t' lzop;
```

Contoh berikut memuat tabel SALES dengan data berformat JSON di cluster Amazon EMR. COPY memuat setiap file dalam `myoutput/json/` folder.

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/json/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
JSON 's3://amzn-s3-demo-bucket/jsonpaths.txt';
```

Menggunakan manifes untuk menentukan file data

Anda dapat menggunakan manifes untuk memastikan bahwa perintah COPY memuat semua file yang diperlukan, dan hanya file yang diperlukan, dari Amazon S3. Anda juga dapat menggunakan manifes ketika Anda perlu memuat beberapa file dari bucket yang berbeda atau file yang tidak berbagi awalan yang sama.

Misalnya, Anda perlu memuat tiga file berikut:`custdata1.txt`,`custdata2.txt`, dan`custdata3.txt`. Anda dapat menggunakan perintah berikut untuk memuat semua file `amzn-s3-demo-bucket` yang dimulai `custdata` dengan menentukan awalan:

```
copy category
from 's3://amzn-s3-demo-bucket/custdata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Jika hanya dua file yang ada karena kesalahan, COPY hanya memuat dua file tersebut dan selesai dengan sukses, menghasilkan pemuatan data yang tidak lengkap. Jika bucket juga berisi file yang tidak diinginkan yang kebetulan menggunakan awalan yang sama, seperti file bernama `custdata.backup` misalnya, COPY memuat file itu juga, mengakibatkan data yang tidak diinginkan dimuat.

Untuk memastikan bahwa semua file yang diperlukan dimuat dan untuk mencegah file yang tidak diinginkan dimuat, Anda dapat menggunakan file manifes. Manifes adalah file JSON-formatted teks yang mencantumkan file yang akan diproses oleh perintah COPY. Misalnya, manifes berikut memuat tiga file dalam contoh sebelumnya.

```
{
   "entries":[
      {
         "url":"s3://amzn-s3-demo-bucket/custdata.1",
         "mandatory":true
      },
      {
         "url":"s3://amzn-s3-demo-bucket/custdata.2",
         "mandatory":true
```

```
      },
      {
         "url":"s3://amzn-s3-demo-bucket/custdata.3",
         "mandatory":true
      }
   ]
}
```

`mandatory`Bendera opsional menunjukkan apakah COPY harus dihentikan jika file tidak ada. Nilai default-nya `false`. Terlepas dari pengaturan wajib, COPY berakhir jika tidak ada file yang ditemukan. Dalam contoh ini, COPY mengembalikan kesalahan jika salah satu file tidak ditemukan. File yang tidak diinginkan yang mungkin telah diambil jika Anda hanya menentukan awalan kunci, seperti`custdata.backup`, diabaikan, karena file tersebut tidak ada di manifes.

Saat memuat dari file data dalam format ORC atau Parquet, `meta` bidang diperlukan, seperti yang ditunjukkan pada contoh berikut.

```
{
   "entries":[
      {
         "url":"s3://amzn-s3-demo-bucket1/orc/2013-10-04-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      },
      {
         "url":"s3://amzn-s3-demo-bucket2/orc/2013-10-05-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      }
   ]
}
```

Contoh berikut menggunakan manifes bernama`cust.manifest`.

```
copy customer
from 's3://amzn-s3-demo-bucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as orc
```

```
manifest;
```

Anda dapat menggunakan manifes untuk memuat file dari bucket atau file yang berbeda yang tidak memiliki awalan yang sama. Contoh berikut menunjukkan JSON untuk memuat data dengan file yang namanya dimulai dengan cap tanggal.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/2013-10-04-custdata.txt","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket/2013-10-05-custdata.txt","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket/2013-10-06-custdata.txt","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket/2013-10-07-custdata.txt","mandatory":true}
  ]
}
```

Manifes dapat mencantumkan file yang berada di bucket yang berbeda, selama bucket berada di Wil AWS ayah yang sama dengan cluster.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata1.txt","mandatory":false},
    {"url":"s3://amzn-s3-demo-bucket2/custdata1.txt","mandatory":false},
    {"url":"s3://amzn-s3-demo-bucket2/custdata2.txt","mandatory":false}
  ]
}
```

Muat LISTING dari file yang dibatasi pipa (pembatas default)

Contoh berikut adalah kasus yang sangat sederhana di mana tidak ada opsi yang ditentukan dan file input berisi pembatas default, karakter pipa ('|').

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listings_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Muat LISTING menggunakan data kolumnar dalam format Parket

Contoh berikut memuat data dari folder di Amazon S3 bernama parket.

```
copy listing
```

```
from 's3://amzn-s3-demo-bucket/data/listings/parquet/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as parquet;
```

Muat LISTING menggunakan data kolumnar dalam format ORC

Contoh berikut memuat data dari folder di Amazon S3 bernama`orc`.

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listings/orc/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as orc;
```

Muat EVENT dengan opsi

Contoh berikut memuat data yang dibatasi pipa ke dalam tabel EVENT dan menerapkan aturan berikut:

- Jika pasangan tanda kutip digunakan untuk mengelilingi string karakter apa pun, mereka dihapus.
- Kedua string kosong dan string yang berisi blanko dimuat sebagai nilai NULL.
- Beban gagal jika lebih dari 5 kesalahan dikembalikan.
- Nilai stempel waktu harus sesuai dengan format yang ditentukan; misalnya, stempel waktu yang valid adalah. `2008-09-26 05:43:12`

```
copy event
from 's3://amzn-s3-demo-bucket/data/allevents_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
removequotes
emptyasnull
blanksasnull
maxerror 5
delimiter '|'
timeformat 'YYYY-MM-DD HH:MI:SS';
```

Muat VENUE dari file data lebar tetap

```
copy venue
from 's3://amzn-s3-demo-bucket/data/venue_fw.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
fixedwidth 'venueid:3,venuename:25,venuecity:12,venuestate:2,venueseats:6';
```

Contoh sebelumnya mengasumsikan file data diformat dengan cara yang sama seperti data sampel yang ditampilkan. Dalam contoh berikut, spasi bertindak sebagai placeholder sehingga semua kolom memiliki lebar yang sama seperti yang tercantum dalam spesifikasi:

```
1  Toyota Park              Bridgeview  IL0
2  Columbus Crew Stadium    Columbus    OH0
3  RFK Stadium              Washington  DC0
4  CommunityAmerica BallparkKansas City KS0
5  Gillette Stadium         Foxborough  MA68756
```

Muat KATEGORI dari file CSV

Misalkan Anda ingin memuat KATEGORI dengan nilai-nilai yang ditunjukkan pada tabel berikut.

Catid	kelompok kucing	nama kucing	catdesc
12	Menunjukkan	Musikal	Teater musikal
13	Menunjukkan	Bermain	Semua teater “non-musikal”
14	Menunjukkan	Opera	Semua opera, cahaya, dan opera “rock”
15	Konser	Klasik	Semua konser simfoni, konser, dan paduan suara

Contoh berikut menunjukkan isi file teks dengan nilai bidang yang dipisahkan oleh koma.

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,All "non-musical" theatre
14,Shows,Opera,All opera, light, and "rock" opera
15,Concerts,Classical,All symphony, concerto, and choir concerts
```

Jika Anda memuat file menggunakan parameter DELIMITER untuk menentukan input yang dibatasi koma, perintah COPY gagal karena beberapa bidang input berisi koma. Anda dapat menghindari masalah itu dengan menggunakan parameter CSV dan melampirkan bidang yang berisi koma

dalam karakter tanda kutip. Jika karakter tanda kutip muncul dalam string yang dikutip, Anda harus menghindarinya dengan menggandakan karakter tanda kutip. Karakter tanda kutip default adalah tanda kutip ganda, jadi Anda harus menghindari setiap tanda kutip ganda dengan tanda kutip ganda tambahan. File input baru Anda terlihat seperti ini.

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,"All ""non-musical"" theatre"
14,Shows,Opera,"All opera, light, and ""rock"" opera"
15,Concerts,Classical,"All symphony, concerto, and choir concerts"
```

Dengan asumsi nama file adalah`category_csv.txt`, Anda dapat memuat file dengan menggunakan perintah COPY berikut:

```
copy category
from 's3://amzn-s3-demo-bucket/data/category_csv.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
csv;
```

Atau, untuk menghindari kebutuhan untuk menghindari tanda kutip ganda dalam input Anda, Anda dapat menentukan karakter tanda kutip yang berbeda dengan menggunakan parameter QUOTE AS. Misalnya, versi berikut `category_csv.txt` menggunakan '%' sebagai karakter tanda kutip.

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,%All "non-musical" theatre%
14,Shows,Opera,%All opera, light, and "rock" opera%
15,Concerts,Classical,%All symphony, concerto, and choir concerts%
```

Perintah COPY berikut menggunakan QUOTE AS untuk memuat`category_csv.txt`:

```
copy category
from 's3://amzn-s3-demo-bucket/data/category_csv.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
csv quote as '%';
```

Muat VENUE dengan nilai eksplisit untuk kolom IDENTITY

Contoh berikut mengasumsikan bahwa ketika tabel VENUE dibuat bahwa setidaknya satu kolom (seperti `venueid` kolom) ditentukan untuk menjadi kolom IDENTITY. Perintah ini mengesampingkan perilaku IDENTITY default dari nilai pembuatan otomatis untuk kolom IDENTITY dan sebagai

gantinya memuat nilai eksplisit dari file venue.txt. Amazon Redshift tidak memeriksa apakah nilai IDENTITY duplikat dimuat ke dalam tabel saat menggunakan opsi EXLICIT_IDS.

```
copy venue
from 's3://amzn-s3-demo-bucket/data/venue.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
explicit_ids;
```

Muat WAKTU dari file GZIP yang dibatasi pipa

Contoh berikut memuat tabel TIME dari file GZIP yang dibatasi pipa:

```
copy time
from 's3://amzn-s3-demo-bucket/data/timerows.gz'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
gzip
delimiter '|';
```

Memuat stempel waktu atau stempel tanggal

Contoh berikut memuat data dengan stempel waktu yang diformat.

> **Note**
>
> TIMEFORMAT juga `HH:MI:SS` dapat mendukung detik pecahan di `SS` luar tingkat detail mikrodetik. File yang `time.txt` digunakan dalam contoh ini berisi satu baris,`2009-01-12 14:15:57.119568`.

```
copy timestamp1
from 's3://amzn-s3-demo-bucket/data/time.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
timeformat 'YYYY-MM-DD HH:MI:SS';
```

Hasil salinan ini adalah sebagai berikut:

```
select * from timestamp1;
c1
----------------------------
2009-01-12 14:15:57.119568
```

```
(1 row)
```

Muat data dari file dengan nilai default

Contoh berikut menggunakan variasi tabel VENUE dalam database TICKIT. Pertimbangkan tabel VENUE_NEW yang didefinisikan dengan pernyataan berikut:

```
create table venue_new(
venueid smallint not null,
venuename varchar(100) not null,
venuecity varchar(30),
venuestate char(2),
venueseats integer not null default '1000');
```

Pertimbangkan file data venue_noseats.txt yang tidak berisi nilai untuk kolom VENUESEATS, seperti yang ditunjukkan pada contoh berikut:

```
1|Toyota Park|Bridgeview|IL|
2|Columbus Crew Stadium|Columbus|OH|
3|RFK Stadium|Washington|DC|
4|CommunityAmerica Ballpark|Kansas City|KS|
5|Gillette Stadium|Foxborough|MA|
6|New York Giants Stadium|East Rutherford|NJ|
7|BMO Field|Toronto|ON|
8|The Home Depot Center|Carson|CA|
9|Dick's Sporting Goods Park|Commerce City|CO|
10|Pizza Hut Park|Frisco|TX|
```

Pernyataan COPY berikut akan berhasil memuat tabel dari file dan menerapkan nilai DEFAULT ('1000') ke kolom yang dihilangkan:

```
copy venue_new(venueid, venuename, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_noseats.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

Sekarang lihat tabel yang dimuat:

```
select * from venue_new order by venueid;
venueid |         venuename          |    venuecity    | venuestate | venueseats
```

```
---------+--------------------------+-----------------+------------+------------
1 | Toyota Park                | Bridgeview      | IL         |       1000
2 | Columbus Crew Stadium      | Columbus        | OH         |       1000
3 | RFK Stadium                | Washington      | DC         |       1000
4 | CommunityAmerica Ballpark  | Kansas City     | KS         |       1000
5 | Gillette Stadium           | Foxborough      | MA         |       1000
6 | New York Giants Stadium    | East Rutherford | NJ         |       1000
7 | BMO Field                  | Toronto         | ON         |       1000
8 | The Home Depot Center      | Carson          | CA         |       1000
9 | Dick's Sporting Goods Park | Commerce City   | CO         |       1000
10 | Pizza Hut Park             | Frisco          | TX         |       1000
(10 rows)
```

Untuk contoh berikut, selain mengasumsikan bahwa tidak ada data VENUESEATS yang disertakan dalam file, asumsikan juga bahwa tidak ada data VENUENAME yang disertakan:

```
1||Bridgeview|IL|
2||Columbus|OH|
3||Washington|DC|
4||Kansas City|KS|
5||Foxborough|MA|
6||East Rutherford|NJ|
7||Toronto|ON|
8||Carson|CA|
9||Commerce City|CO|
10||Frisco|TX|
```

Menggunakan definisi tabel yang sama, pernyataan COPY berikut gagal karena tidak ada nilai DEFAULT yang ditentukan untuk VENUENAME, dan VENUENAME adalah kolom NOT NULL:

```
copy venue(venueid, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

Sekarang pertimbangkan variasi tabel VENUE yang menggunakan kolom IDENTITY:

```
create table venue_identity(
venueid int identity(1,1),
venuename varchar(100) not null,
venuecity varchar(30),
venuestate char(2),
```

```
venueseats integer not null default '1000');
```

Seperti contoh sebelumnya, asumsikan bahwa kolom VENUESEATS tidak memiliki nilai yang sesuai dalam file sumber. Pernyataan COPY berikut berhasil memuat tabel, termasuk nilai data IDENTITY yang telah ditentukan sebelumnya alih-alih membuat nilai-nilai tersebut secara otomatis:

```
copy venue(venueid, venuename, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' explicit_ids;
```

Pernyataan ini gagal karena tidak menyertakan kolom IDENTITY (VENUEID hilang dari daftar kolom) namun menyertakan parameter EXPLICIT_IDS:

```
copy venue(venuename, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' explicit_ids;
```

Pernyataan ini gagal karena tidak menyertakan parameter EXPLICIT_IDS:

```
copy venue(venueid, venuename, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

SALIN data dengan opsi ESCAPE

Contoh berikut menunjukkan cara memuat karakter yang cocok dengan karakter pembatas (dalam hal ini, karakter pipa). Dalam file input, pastikan bahwa semua karakter pipa (|) yang ingin Anda muat dilepaskan dengan karakter garis miring terbalik (\). Kemudian muat file dengan parameter ESCAPE.

```
$ more redshiftinfo.txt
1|public\|event\|dwuser
2|public\|sales\|dwuser

create table redshiftinfo(infoid int,tableinfo varchar(50));

copy redshiftinfo from 's3://amzn-s3-demo-bucket/data/redshiftinfo.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
delimiter '|' escape;

select * from redshiftinfo order by 1;
infoid |       tableinfo
-------+--------------------
1      | public|event|dwuser
2      | public|sales|dwuser
(2 rows)
```

Tanpa parameter ESCAPE, perintah COPY ini gagal dengan `Extra column(s) found` kesalahan.

> **Important**
>
> Jika Anda memuat data menggunakan COPY dengan parameter ESCAPE, Anda juga harus menentukan parameter ESCAPE dengan perintah UNLOAD untuk menghasilkan file output timbal balik. Demikian pula, jika Anda UNLOAD menggunakan parameter ESCAPE, Anda perlu menggunakan ESCAPE ketika Anda MENYALIN data yang sama.

Salin dari contoh JSON

Dalam contoh berikut, Anda memuat tabel CATEGORY dengan data berikut.

CATID	KELOMPOK KUCING	NAMA KUCING	CATDESC
1	Olahraga	MLB	Liga Utama Baseball
2	Olahraga	NHL	Liga Hoki Nasional
3	Olahraga	NFL	Liga Sepak Bola Nasional
4	Olahraga	NBA	Asosiasi Bola Basket Nasional
5	Konser	Klasik	Semua konser simfoni, konser, dan paduan suara

Topik

- Muat dari data JSON menggunakan opsi 'otomatis'

- Muat dari data JSON menggunakan opsi 'auto ignorecase'
- Muat dari data JSON menggunakan file JSONPaths
- Muat dari array JSON menggunakan file JSONPaths

Muat dari data JSON menggunakan opsi 'otomatis'

Untuk memuat dari data JSON menggunakan `'auto'` opsi, data JSON harus terdiri dari sekumpulan objek. Nama kunci harus cocok dengan nama kolom, tetapi urutannya tidak masalah. Berikut ini menunjukkan isi dari file bernama`category_object_auto.json`.

```
{
    "catdesc": "Major League Baseball",
    "catid": 1,
    "catgroup": "Sports",
    "catname": "MLB"
}
{
    "catgroup": "Sports",
    "catid": 2,
    "catname": "NHL",
    "catdesc": "National Hockey League"
}
{
    "catid": 3,
    "catname": "NFL",
    "catgroup": "Sports",
    "catdesc": "National Football League"
}
{
    "bogus": "Bogus Sports LLC",
    "catid": 4,
    "catgroup": "Sports",
    "catname": "NBA",
    "catdesc": "National Basketball Association"
}
{
    "catid": 5,
    "catgroup": "Shows",
    "catname": "Musicals",
    "catdesc": "All symphony, concerto, and choir concerts"
}
```

Untuk memuat dari file data JSON dalam contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_auto.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 'auto';
```

Muat dari data JSON menggunakan opsi 'auto ignorecase'

Untuk memuat dari data JSON menggunakan `'auto ignorecase'` opsi, data JSON harus terdiri dari sekumpulan objek. Kasus nama kunci tidak harus cocok dengan nama kolom dan urutannya tidak masalah. Berikut ini menunjukkan isi dari file bernama`category_object_auto-ignorecase.json`.

```
{
    "CatDesc": "Major League Baseball",
    "CatID": 1,
    "CatGroup": "Sports",
    "CatName": "MLB"
}
{
    "CatGroup": "Sports",
    "CatID": 2,
    "CatName": "NHL",
    "CatDesc": "National Hockey League"
}
{
    "CatID": 3,
    "CatName": "NFL",
    "CatGroup": "Sports",
    "CatDesc": "National Football League"
}
{
    "bogus": "Bogus Sports LLC",
    "CatID": 4,
    "CatGroup": "Sports",
    "CatName": "NBA",
    "CatDesc": "National Basketball Association"
}
{
    "CatID": 5,
    "CatGroup": "Shows",
```

```
    "CatName": "Musicals",
    "CatDesc": "All symphony, concerto, and choir concerts"
}
```

Untuk memuat dari file data JSON dalam contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_auto ignorecase.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 'auto ignorecase';
```

Muat dari data JSON menggunakan file JSONPaths

Jika objek data JSON tidak sesuai langsung dengan nama kolom, Anda dapat menggunakan file JSONPaths untuk memetakan elemen JSON ke kolom. Urutan tidak masalah dalam data sumber JSON, tetapi urutan ekspresi file JSONPaths harus sesuai dengan urutan kolom. Misalkan Anda memiliki file data berikut, bernama`category_object_paths.json`.

```
{
    "one": 1,
    "two": "Sports",
    "three": "MLB",
    "four": "Major League Baseball"
}
{
    "three": "NHL",
    "four": "National Hockey League",
    "one": 2,
    "two": "Sports"
}
{
    "two": "Sports",
    "three": "NFL",
    "one": 3,
    "four": "National Football League"
}
{
    "one": 4,
    "two": "Sports",
    "three": "NBA",
    "four": "National Basketball Association"
}
```

```
{
    "one": 6,
    "two": "Shows",
    "three": "Musicals",
    "four": "All symphony, concerto, and choir concerts"
}
```

File JSONPaths berikut, bernama`category_jsonpath.json`, memetakan data sumber ke kolom tabel.

```
{
    "jsonpaths": [
        "$['one']",
        "$['two']",
        "$['three']",
        "$['four']"
    ]
}
```

Untuk memuat dari file data JSON dalam contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_paths.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 's3://amzn-s3-demo-bucket/category_jsonpath.json';
```

Muat dari array JSON menggunakan file JSONPaths

Untuk memuat dari data JSON yang terdiri dari satu set array, Anda harus menggunakan file JSONPaths untuk memetakan elemen array ke kolom. Misalkan Anda memiliki file data berikut, bernama`category_array_data.json`.

```
[1,"Sports","MLB","Major League Baseball"]
[2,"Sports","NHL","National Hockey League"]
[3,"Sports","NFL","National Football League"]
[4,"Sports","NBA","National Basketball Association"]
[5,"Concerts","Classical","All symphony, concerto, and choir concerts"]
```

File JSONPaths berikut, bernama`category_array_jsonpath.json`, memetakan data sumber ke kolom tabel.

```
{
    "jsonpaths": [
        "$[0]",
        "$[1]",
        "$[2]",
        "$[3]"
    ]
}
```

Untuk memuat dari file data JSON dalam contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_array_data.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 's3://amzn-s3-demo-bucket/category_array_jsonpath.json';
```

Salin dari contoh Avro

Dalam contoh berikut, Anda memuat tabel CATEGORY dengan data berikut.

CATID	KELOMPOK KUCING	NAMA KUCING	CATDESC
1	Olahraga	MLB	Liga Utama Baseball
2	Olahraga	NHL	Liga Hoki Nasional
3	Olahraga	NFL	Liga Sepak Bola Nasional
4	Olahraga	NBA	Asosiasi Bola Basket Nasional
5	Konser	Klasik	Semua konser simfoni, konser, dan paduan suara

Topik

- Muat dari data Avro menggunakan opsi 'otomatis'
- Muat dari data Avro menggunakan opsi 'auto ignorecase'
- Muat dari data Avro menggunakan file JSONPaths

Muat dari data Avro menggunakan opsi 'otomatis'

Untuk memuat dari data Avro menggunakan `'auto'` argumen, nama bidang dalam skema Avro harus cocok dengan nama kolom. Saat menggunakan `'auto'` argumen, urutan tidak masalah. Berikut ini menunjukkan skema untuk file bernama`category_auto.avro`.

```
{
    "name": "category",
    "type": "record",
    "fields": [
        {"name": "catid", "type": "int"},
        {"name": "catdesc", "type": "string"},
        {"name": "catname", "type": "string"},
        {"name": "catgroup", "type": "string"},
}
```

Data dalam file Avro dalam format biner, jadi tidak dapat dibaca manusia. Berikut ini menunjukkan representasi JSON dari data dalam `category_auto.avro` file.

```
{
   "catid": 1,
   "catdesc": "Major League Baseball",
   "catname": "MLB",
   "catgroup": "Sports"
}
{
   "catid": 2,
   "catdesc": "National Hockey League",
   "catname": "NHL",
   "catgroup": "Sports"
}
{
   "catid": 3,
   "catdesc": "National Basketball Association",
   "catname": "NBA",
   "catgroup": "Sports"
}
{
   "catid": 4,
   "catdesc": "All symphony, concerto, and choir concerts",
   "catname": "Classical",
   "catgroup": "Concerts"
```

```
}
```

Untuk memuat dari file data Avro pada contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_auto.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as avro 'auto';
```

Muat dari data Avro menggunakan opsi 'auto ignorecase'

Untuk memuat dari data Avro menggunakan `'auto ignorecase'` argumen, kasus nama bidang dalam skema Avro tidak harus cocok dengan kasus nama kolom. Saat menggunakan `'auto ignorecase'` argumen, urutan tidak masalah. Berikut ini menunjukkan skema untuk file bernama`category_auto-ignorecase.avro`.

```
{
    "name": "category",
    "type": "record",
    "fields": [
        {"name": "CatID", "type": "int"},
        {"name": "CatDesc", "type": "string"},
        {"name": "CatName", "type": "string"},
        {"name": "CatGroup", "type": "string"},
}
```

Data dalam file Avro dalam format biner, jadi tidak dapat dibaca manusia. Berikut ini menunjukkan representasi JSON dari data dalam `category_auto-ignorecase.avro` file.

```
{
   "CatID": 1,
   "CatDesc": "Major League Baseball",
   "CatName": "MLB",
   "CatGroup": "Sports"
}
{
   "CatID": 2,
   "CatDesc": "National Hockey League",
   "CatName": "NHL",
   "CatGroup": "Sports"
}
{
```

```
   "CatID": 3,
   "CatDesc": "National Basketball Association",
   "CatName": "NBA",
   "CatGroup": "Sports"
}
{
   "CatID": 4,
   "CatDesc": "All symphony, concerto, and choir concerts",
   "CatName": "Classical",
   "CatGroup": "Concerts"
}
```

Untuk memuat dari file data Avro pada contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_auto-ignorecase.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as avro 'auto ignorecase';
```

Muat dari data Avro menggunakan file JSONPaths

Jika nama bidang dalam skema Avro tidak sesuai langsung dengan nama kolom, Anda dapat menggunakan file JSONPaths untuk memetakan elemen skema ke kolom. Urutan ekspresi file JSONPaths harus sesuai dengan urutan kolom.

Misalkan Anda memiliki file data bernama `category_paths.avro` yang berisi data yang sama seperti pada contoh sebelumnya, tetapi dengan skema berikut.

```
{
    "name": "category",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "desc", "type": "string"},
        {"name": "name", "type": "string"},
        {"name": "group", "type": "string"},
        {"name": "region", "type": "string"}
     ]
}
```

File JSONPaths berikut, bernama`category_path.avropath`, memetakan data sumber ke kolom tabel.

```
{
    "jsonpaths": [
        "$['id']",
        "$['group']",
        "$['name']",
        "$['desc']"
    ]
}
```

Untuk memuat dari file data Avro pada contoh sebelumnya, jalankan perintah COPY berikut.

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_paths.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format avro 's3://amzn-s3-demo-bucket/category_path.avropath ';
```

Mempersiapkan file untuk COPY dengan opsi ESCAPE

Contoh berikut menjelaskan cara menyiapkan data untuk “melarikan diri” dari karakter baris baru sebelum mengimpor data ke tabel Amazon Redshift menggunakan perintah COPY dengan parameter ESCAPE. Tanpa menyiapkan data untuk membatasi karakter baris baru, Amazon Redshift mengembalikan kesalahan pemuatan saat Anda menjalankan perintah COPY, karena karakter baris baru biasanya digunakan sebagai pemisah catatan.

Misalnya, pertimbangkan file atau kolom di tabel eksternal yang ingin Anda salin ke tabel Amazon Redshift. Jika file atau kolom berisi XML-formatted konten atau data serupa, Anda perlu memastikan bahwa semua karakter baris baru (\ n) yang merupakan bagian dari konten dihilangkan dengan karakter garis miring terbalik (\).

File atau tabel yang berisi karakter baris baru yang disematkan memberikan pola yang relatif mudah untuk dicocokkan. Setiap karakter baris baru yang disematkan kemungkinan besar selalu mengikuti > karakter dengan potensi beberapa karakter spasi putih (' 'atau tab) di antaranya, seperti yang dapat Anda lihat dalam contoh berikut dari file teks bernama`nlTest1.txt`.

```
$ cat nlTest1.txt
<xml start>
<newline characters provide>
<line breaks at the end of each>
<line in content>
</xml>|1000
```

```
<xml>
</xml>|2000
```

Dengan contoh berikut, Anda dapat menjalankan utilitas pengolahan teks untuk melakukan pra-proses file sumber dan memasukkan karakter escape jika diperlukan. (|Karakter dimaksudkan untuk digunakan sebagai pembatas untuk memisahkan data kolom saat disalin ke tabel Amazon Redshift.)

```
$ sed -e ':a;N;$!ba;s/>[[:space:]]*\n/>\\\n/g' nlTest1.txt > nlTest2.txt
```

Demikian pula, Anda dapat menggunakan Perl untuk melakukan operasi serupa:

```
cat nlTest1.txt | perl -p -e 's/>\s*\n/>\\\n/g' > nlTest2.txt
```

Untuk mengakomodasi pemuatan data dari `nlTest2.txt` file ke Amazon Redshift, contoh ini membuat tabel dua kolom di Amazon Redshift. Kolom pertama c1, adalah kolom karakter yang menyimpan XML-formatted konten dari `nlTest2.txt` file. Kolom kedua c2 menyimpan nilai integer yang dimuat dari file yang sama.

Setelah menjalankan `sed` perintah, Anda dapat memuat data dari `nlTest2.txt` file dengan benar ke tabel Amazon Redshift menggunakan parameter ESCAPE.

> **Note**
>
> Ketika Anda menyertakan parameter ESCAPE dengan perintah COPY, itu lolos dari sejumlah karakter khusus yang menyertakan karakter garis miring terbalik (termasuk baris baru).

```
copy t2 from 's3://amzn-s3-demo-bucket/data/nlTest2.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
escape
delimiter as '|';

select * from t2 order by 2;

c1           |  c2
-------------+------
<xml start>
<newline characters provide>
<line breaks at the end of each>
```

```
<line in content>
</xml>
| 1000
<xml>
</xml>       | 2000
(2 rows)
```

Anda dapat menyiapkan file data yang diekspor dari database eksternal dengan cara yang sama. Misalnya, dengan database Oracle, Anda dapat menggunakan fungsi REPLACE pada setiap kolom yang terpengaruh dalam tabel yang ingin Anda salin ke Amazon Redshift.

```
SELECT c1, REPLACE(c2, \n',\\n' ) as c2 from my_table_with_xml
```

Selain itu, banyak alat ekspor dan ekstrak, transformasi, muat (ETL) database yang secara rutin memproses sejumlah besar data menyediakan opsi untuk menentukan karakter escape dan pembatas.

Memuat shapefile ke Amazon Redshift

Contoh berikut menunjukkan cara memuat shapefile Esri menggunakan COPY. Untuk informasi selengkapnya tentang memuat shapefiles, lihat. Memuat shapefile ke Amazon Redshift

Memuat shapefile

Langkah-langkah berikut menunjukkan cara menelan OpenStreetMap data dari Amazon S3 menggunakan perintah COPY. Contoh ini mengasumsikan bahwa arsip shapefile Norwegia dari situs unduhan Geofabrik telah diunggah ke bucket Amazon S3 pribadi di Wilayah Anda. AWS File `.shp`, `.shx`, dan `.dbf` harus berbagi awalan dan nama file Amazon S3 yang sama.

Menelan data tanpa penyederhanaan

Perintah berikut membuat tabel dan menelan data yang dapat masuk dalam ukuran geometri maksimum tanpa penyederhanaan apa pun. Buka perangkat `gis_osm_natural_free_1.shp` lunak GIS pilihan Anda dan periksa kolom di lapisan ini. Secara default, kolom IDENTITY atau GEOMETRY adalah yang pertama. Ketika kolom GEOMETRY pertama, Anda dapat membuat tabel seperti yang ditunjukkan berikut.

```
CREATE TABLE norway_natural (
   wkb_geometry GEOMETRY,
   osm_id BIGINT,
```

```
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

Atau, ketika kolom IDENTITY pertama, Anda dapat membuat tabel seperti yang ditunjukkan berikut.

```
CREATE TABLE norway_natural_with_id (
   fid INT IDENTITY(1,1),
   wkb_geometry GEOMETRY,
   osm_id BIGINT,
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

Sekarang Anda dapat menelan data menggunakan COPY.

```
COPY norway_natural FROM 's3://bucket_name/shapefiles/norway/
gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural' completed, 83891 record(s) loaded successfully
```

Atau Anda dapat menelan data seperti yang ditunjukkan berikut.

```
COPY norway_natural_with_id FROM 's3://bucket_name/shapefiles/norway/
gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural_with_id' completed, 83891 record(s) loaded
 successfully.
```

Menelan data dengan penyederhanaan

Perintah berikut membuat tabel dan mencoba menelan data yang tidak dapat muat dalam ukuran geometri maksimum tanpa penyederhanaan apa pun. Periksa `gis_osm_water_a_free_1.shp` shapefile dan buat tabel yang sesuai seperti yang ditunjukkan berikut.

```
CREATE TABLE norway_water (
   wkb_geometry GEOMETRY,
   osm_id BIGINT,
```

```
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

Ketika perintah COPY berjalan, itu menghasilkan kesalahan.

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
ERROR:  Load into table 'norway_water' failed.  Check 'stl_load_errors' system table
 for details.
```

Kueri STL_LOAD_ERRORS menunjukkan bahwa geometri terlalu besar.

```
SELECT line_number, btrim(colname), btrim(err_reason) FROM stl_load_errors WHERE query
 = pg_last_copy_id();
 line_number |    btrim     |                                  btrim
-------------+--------------
+-------------------------------------------------------------------------
     1184705 | wkb_geometry | Geometry size: 1513736 is larger than maximum supported
 size: 1048447
```

Untuk mengatasi hal ini, SIMPLIFY AUTO parameter ditambahkan ke perintah COPY untuk menyederhanakan geometri.

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
SIMPLIFY AUTO
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';

INFO:  Load into table 'norway_water' completed, 1989196 record(s) loaded successfully.
```

Untuk melihat baris dan geometri yang disederhanakan, kueri. SVL_SPATIAL_SIMPLIFY

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
 final_tolerance
-------+-------------+-------------------+--------------+------------+------------
+----------------------
```

```
    20 |     1184704 |               -1 |      1513736 | t          |    1008808 |
 1.276386653895e-05
    20 |     1664115 |               -1 |      1233456 | t          |    1023584 |
 6.11707814796635e-06
```

Menggunakan SIMPLIF Y AUTO max_tolerance dengan toleransi yang lebih rendah dari yang dihitung secara otomatis mungkin menghasilkan kesalahan konsumsi. Dalam hal ini, gunakan MAXERROR untuk mengabaikan kesalahan.

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
SIMPLIFY AUTO 1.1E-05
MAXERROR 2
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';

INFO:  Load into table 'norway_water' completed, 1989195 record(s) loaded successfully.
INFO:  Load into table 'norway_water' completed, 1 record(s) could not be loaded.
 Check 'stl_load_errors' system table for details.
```

Kueri SVL_SPATIAL_SIMPLIFY lagi untuk mengidentifikasi catatan yang tidak berhasil dimuat COPY.

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
 final_tolerance
-------+-------------+-------------------+--------------+------------+------------
+-----------------
    29 |     1184704 |           1.1e-05 |      1513736 | f          |          0 |
            0
    29 |     1664115 |           1.1e-05 |      1233456 | t          |     794432 |
     1.1e-05
```

Dalam contoh ini, catatan pertama tidak berhasil masuk, jadi `simplified` kolom menunjukkan false. Catatan kedua dimuat dalam toleransi yang diberikan. Namun, ukuran akhir lebih besar daripada menggunakan toleransi yang dihitung secara otomatis tanpa menentukan toleransi maksimum.

Memuat dari shapefile terkompresi

Amazon Redshift COPY mendukung pengambilan data dari shapefile terkompresi. Semua komponen shapefile harus memiliki awalan Amazon S3 yang sama dan akhiran kompresi yang

sama. Sebagai contoh, misalkan Anda ingin memuat data dari contoh sebelumnya. Dalam hal ini, file`gis_osm_water_a_free_1.shp.gz`,`gis_osm_water_a_free_1.dbf.gz`, dan `gis_osm_water_a_free_1.shx.gz` harus berbagi direktori Amazon S3 yang sama. Perintah COPY memerlukan opsi GZIP, dan klausa FROM harus menentukan file terkompresi yang benar, seperti yang ditunjukkan berikut.

```
COPY norway_natural FROM 's3://bucket_name/shapefiles/norway/compressed/
gis_osm_natural_free_1.shp.gz'
FORMAT SHAPEFILE
GZIP
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO:  Load into table 'norway_natural' completed, 83891 record(s) loaded successfully.
```

Memuat data ke dalam tabel dengan urutan kolom yang berbeda

Jika Anda memiliki tabel yang tidak memiliki kolom `GEOMETRY` pertama, Anda dapat menggunakan pemetaan kolom untuk memetakan kolom ke tabel target. Misalnya, buat tabel dengan `osm_id` ditentukan sebagai kolom pertama.

```
CREATE TABLE norway_natural_order (
   osm_id BIGINT,
   wkb_geometry GEOMETRY,
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

Kemudian menelan shapefile menggunakan pemetaan kolom.

```
COPY norway_natural_order(wkb_geometry, osm_id, code, fclass, name)
FROM 's3://bucket_name/shapefiles/norway/gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO:  Load into table 'norway_natural_order' completed, 83891 record(s) loaded
 successfully.
```

Memuat data ke dalam tabel dengan kolom geografi

Jika Anda memiliki tabel yang memiliki `GEOGRAPHY` kolom, pertama-tama Anda memasukkan ke dalam `GEOMETRY` kolom dan kemudian melemparkan objek ke `GEOGRAPHY` objek. Misalnya, setelah

Anda menyalin shapefile Anda ke GEOMETRY kolom, ubah tabel untuk menambahkan kolom tipe GEOGRAPHY data.

```
ALTER TABLE norway_natural ADD COLUMN wkb_geography GEOGRAPHY;
```

Kemudian ubah geometri menjadi geografi.

```
UPDATE norway_natural SET wkb_geography = wkb_geometry::geography;
```

Secara opsional, Anda dapat menjatu GEOMETRY hkan kolom.

```
ALTER TABLE norway_natural DROP COLUMN wkb_geometry;
```

Perintah COPY dengan opsi NOLOAD

Untuk memvalidasi file data sebelum Anda benar-benar memuat data, gunakan opsi NOLOAD dengan perintah COPY. Amazon Redshift mengurai file input dan menampilkan kesalahan apa pun yang terjadi. Contoh berikut menggunakan opsi NOLOAD dan tidak ada baris yang benar-benar dimuat ke dalam tabel.

```
COPY public.zipcode1
FROM 's3://amzn-s3-demo-bucket/mydata/zipcode.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
NOLOAD
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/myRedshiftRole';

Warnings:
Load into table 'zipcode1' completed, 0 record(s) loaded successfully.
```

Perintah COPY dengan pembatas multibyte dan opsi ENCODING

Contoh berikut memuat LATIN1 dari file Amazon S3 yang berisi data multibyte. Perintah COPY menentukan pembatas dalam bentuk oktal \302\246\303\254 untuk memisahkan bidang dalam file input yang dikodekan sebagai. ISO-8859-1 Untuk menentukan pembatas yang sama di UTF-8, tentukan. DELIMITER '¦ì'

```
COPY latin1
FROM 's3://amzn-s3-demo-bucket/multibyte/myfile'
IAM_ROLE 'arn:aws:iam::123456789012:role/myRedshiftRole'
DELIMITER '\302\246\303\254'
ENCODING ISO88591
```

BUAT BASIS DATA

Membuat database baru.

Untuk membuat database, Anda harus menjadi superuser atau memiliki hak istimewa CREATEDB. Untuk membuat database yang terkait dengan integrasi Zero-ETL, Anda harus menjadi superuser atau memiliki hak istimewa CREATEDB dan CREATEUSER.

Anda tidak dapat menjalankan CREATE DATABASE dalam blok transaksi (BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Sintaksis

```
CREATE DATABASE database_name
[ { [
      FROM INTEGRATION '<integration_id>'[ DATABASE '<source_database>' ]
      [ SET ]
      [ ACCEPTINVCHARS [=] { TRUE | FALSE }]
      [ QUERY_ALL_STATES [=] { TRUE | FALSE }]
      [ REFRESH_INTERVAL <interval> ]
      [ TRUNCATECOLUMNS [=] { TRUE | FALSE } ]
      [ HISTORY_MODE [=] {TRUE | FALSE} ]
      [ AUTO_REMEDIATION [=] { TRUE | FALSE } ]
    ]
    [ WITH ]
    [ OWNER [=] db_owner ]
    [ CONNECTION LIMIT { limit | UNLIMITED } ]
    [ COLLATE { CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI } ]
    [ ISOLATION LEVEL { SNAPSHOT | SERIALIZABLE } ]
  }
  | { FROM { { ARN '<arn>' } { WITH DATA CATALOG SCHEMA '<schema>' | WITH NO DATA
 CATALOG SCHEMA } } }
  | { IAM_ROLE  {default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' } }
  | { [ WITH PERMISSIONS ] FROM DATASHARE datashare_name OF [ ACCOUNT account_id ]
 NAMESPACE namespace_guid }
```

```
]
```

Parameter

database_name

Nama database baru. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

DARI INTEGRASI '<integration_id>' [DATABASE '<source_database>']

Menentukan apakah akan membuat database menggunakan pengidentifikasi integrasi Zero-ETL. Anda dapat mengambil `integration_id` dari tampilan sistem SVV_INTEGRATION. Untuk integrasi Aurora PostgreSQL Zero-ETL, Anda juga perlu menentukan `source_database` nama, yang juga dapat diambil dari SVV_INTEGRATION.

Sebagai contoh, lihat Buat database untuk menerima hasil integrasi Zero-ETL. Untuk informasi selengkapnya tentang membuat database dengan integrasi Zero-ETL, lihat Membuat database tujuan di Amazon Redshift di Panduan Manajemen Amazon Redshift.

SET

Kata kunci opsional.

ACCEPTINVCHARS [=] {BENAR | SALAH}

Klausa ACCEPTINVCHARS menetapkan apakah tabel integrasi Zero-ETL berlanjut dengan penyerapan ketika karakter tidak valid terdeteksi untuk tipe data VARCHAR. Ketika karakter tidak valid ditemukan, karakter yang tidak valid diganti dengan ? karakter default.

QUERY_ALL_STATES [=] {BENAR | SALAH}

Klausa QUERY_ALL_STATES menetapkan apakah tabel integrasi Zero-ETL dapat ditanyakan di semua status (,,, dan). `Synced` `Failed` `ResyncRequired` `ResyncInitiated` Secara default, tabel integrasi Zero-ETL hanya dapat ditanyakan dalam status. `Synced`

REFRESH_INTERVAL <interval>

Klausa REFRESH_INTERVAL menetapkan perkiraan interval waktu, dalam detik, yang ditunggu Amazon Redshift setelah siklus refresh selesai untuk memulai siklus berikutnya. Setiap siklus menyegarkan data dari sumber integrasi Zero-ETL ke database target, menerapkan semua perubahan yang terakumulasi sejak akhir siklus sebelumnya. Nilai 0 memulai siklus berikutnya segera setelah siklus sebelumnya selesai (replikasi hampir real-time). Nilai ruang yang lebih tinggi

berputar lebih jauh, mengurangi overhead penyegaran dengan mengorbankan kesegaran data langsung. Amazon Redshift menunggu di antara siklus hanya ketika konsumsi telah menyusul; ketika perubahan menumpuk lebih cepat daripada yang dapat diterapkan, siklus berjalan tanpa menunggu.

`interval`Dapat diatur ke 0-432.000 detik (5 hari) untuk integrasi Zero-ETL yang tipe sumbernya adalah Aurora MySQL, Aurora PostgreSQL, atau mesin RDS yang didukung, dan defaultnya adalah 0. Untuk integrasi Amazon DynamoDB Zero-ETL, `interval` dapat diatur ke 900—432.000 detik (15 menit—5 hari), dan defaultnya adalah 900 detik (15 menit); 0 tidak didukung.

> **Note**
>
> Untuk integrasi penulisan intensif selain DynamoDB yang menghasilkan volume perubahan yang tinggi, atur REFRESH_INTERVAL ke nilai kecil bukan nol (misalnya, 60-120 detik). Mengelompokkan lebih banyak perubahan ke dalam setiap siklus mengurangi keseluruhan overhead komputasi replikasi.

TRUNCATECOLUMNS [=] {BENAR | SALAH}

Klausa TRUNCATECOLUMNS menetapkan apakah tabel integrasi Zero-ETL berlanjut dengan penyerapan ketika nilai untuk kolom VARCHAR atau atribut kolom SUPER berada di luar batas. KetikaTRUE, nilai-nilai dipotong agar sesuai dengan kolom dan nilai atribut JSON yang meluap dipotong agar sesuai dengan kolom SUPER.

HISTORY_MODE [=] {BENAR | SALAH}

Klausa yang menentukan apakah Amazon Redshift akan menyetel mode riwayat untuk semua tabel baru dalam database yang ditentukan. Opsi ini hanya berlaku untuk database yang dibuat untuk integrasi Zero-ETL.

Klausa HISTORY_MODE dapat diatur ke TRUE atau. FALSE Nilai default-nya FALSE. Untuk informasi tentang HISTORY_MODE, lihat Mode Riwayat di Panduan Manajemen Amazon Redshift.

AUTO_REMEDIASI [=] {BENAR | SALAH}

Menentukan apakah Amazon Redshift secara otomatis menyinkronkan ulang tabel yang terpengaruh oleh baris duplikat. Saat disetel keTRUE, Amazon Redshift menandai tabel yang terpengaruh untuk disinkronisasi ulang tanpa memerlukan intervensi manual. Nilai default-nya FALSE.

Anda dapat memantau tabel mana yang telah ditandai dengan menanyakan. SVV_INTEGRASI_TABLE_STATE Pengaturan saat ini terlihat di `auto_remediation` kolomSVV_INTEGRASI.

DENGAN

Kata kunci opsional.

PEMILIK [=] db_owner

Menentukan nama pengguna pemilik database.

BATAS KONEKSI {limit | TIDAK TERBATAS}

Jumlah maksimum koneksi database pengguna diizinkan untuk membuka secara bersamaan. Batas tidak diberlakukan untuk pengguna super. Gunakan kata kunci UNLIMITED untuk mengizinkan jumlah maksimum koneksi bersamaan. Batas jumlah koneksi untuk setiap pengguna mungkin juga berlaku. Untuk informasi selengkapnya, lihat BUAT PENGGUNA. Defaultnya adalah UNLIMITED. Untuk melihat koneksi saat ini, kueri STV_SESSION tampilan sistem.

> **Note**
>
> Jika batas koneksi pengguna dan database berlaku, slot koneksi yang tidak digunakan harus tersedia yang berada dalam kedua batas saat pengguna mencoba menyambung.

SUSUN {CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI}

Klausa yang menentukan apakah pencarian atau perbandingan string peka huruf besar atau tidak peka huruf besar/kecil. Defaultnya peka huruf besar/kecil.

COLLATE tidak didukung saat Anda membuat database dari datashare.

CASE_SENSITIVE dan CS dapat dipertukarkan dan menghasilkan hasil yang sama. Demikian pula, CASE_INSENSITIVE dan CI dapat dipertukarkan dan menghasilkan hasil yang sama.

Untuk memeriksa kolasi database saat ini, gunakan DB_COLLATION fungsi.

TINGKAT ISOLASI {SNAPSHOT | DAPAT DISERIALKAN}

Klausa yang menentukan tingkat isolasi yang digunakan ketika query dijalankan terhadap database. Untuk informasi lebih lanjut tentang tingkat isolasi, lihatTingkat isolasi di Amazon Redshift.

- Isolasi SNAPSHOT — Menyediakan tingkat isolasi dengan perlindungan terhadap konflik pembaruan dan penghapusan. Ini adalah default untuk database yang dibuat di cluster yang disediakan atau namespace tanpa server.
- Isolasi SERIALIZABLE - Menyediakan serialisasi penuh untuk transaksi bersamaan.

DARI ARN '<ARN>'

AWS Glue Database ARN untuk digunakan untuk membuat database.

{DENGAN SKEMA KATALOG DATA <schema>" | TANPA SKEMA KATALOG DATA}

> **Note**
>
> Parameter ini hanya berlaku jika perintah CREATE DATABASE Anda juga menggunakan parameter FROM ARN.

Menentukan apakah akan membuat database menggunakan skema untuk membantu mengakses objek di AWS Glue Data Catalog.

IAM_ROLE {default | 'SESI' | 'arn:aws:iam: :role/ '} *<Akun AWS-id> <role-name>*

> **Note**
>
> Parameter ini hanya berlaku jika perintah CREATE DATABASE Anda juga menggunakan parameter FROM ARN.

Jika Anda menentukan peran IAM yang terkait dengan cluster saat menjalankan perintah CREATE DATABASE, Amazon Redshift akan menggunakan kredentif peran saat Anda menjalankan kueri pada database.

Menentukan `default` kata kunci berarti menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster.

Gunakan `'SESSION'` jika Anda terhubung ke cluster Amazon Redshift menggunakan identitas federasi dan mengakses tabel dari skema eksternal yang dibuat menggunakan perintah ini. Untuk contoh penggunaan identitas federasi, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift Spectrum, yang menjelaskan cara mengonfigurasi identitas federasi.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Minimal, peran IAM harus memiliki izin untuk melakukan operasi LIST pada bucket Amazon S3 untuk diakses dan operasi GET pada objek Amazon S3 yang berisi bucket. Untuk mempelajari lebih lanjut tentang menggunakan IAM_ROLE saat membuat database menggunakan AWS Glue Data Catalog untuk berbagi data, lihat Bek erja dengan berbagi data Lake Formation-managed sebagai konsumen.

Berikut ini menunjukkan sintaks untuk string parameter IAM_ROLE untuk satu ARN.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

Anda dapat mengrantai peran sehingga cluster Anda dapat mengambil peran IAM lain, mungkin milik akun lain. Anda dapat merangkai hingga 10 peran. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Amazon Redshift Spectrum.

Untuk peran IAM ini, lampirkan kebijakan izin IAM yang serupa dengan berikut ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:my-rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Untuk langkah-langkah membuat peran IAM untuk digunakan dengan kueri federasi, lihat. Membuat peran rahasia dan IAM untuk menggunakan kueri federasi

> **Note**
>
> Jangan sertakan spasi dalam daftar peran yang dirantai.

Berikut ini menunjukkan sintaks untuk rantai tiga peran.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

Sintaks untuk menggunakan CREATE DATABASE dengan datashare

Sintaks berikut menjelaskan perintah CREATE DATABASE yang digunakan untuk membuat database dari datashare untuk berbagi data dalam akun yang sama AWS .

```
CREATE DATABASE database_name
[ [ WITH PERMISSIONS ] FROM DATASHARE datashare_name OF [ ACCOUNT account_id ]
 NAMESPACE namespace_guid
```

Sintaks berikut menjelaskan perintah CREATE DATABASE yang digunakan untuk membuat database dari datashare untuk berbagi data di seluruh AWS akun.

```
CREATE DATABASE database_name
[ [ WITH PERMISSIONS ] FROM DATASHARE datashare_name OF ACCOUNT account_id
 NAMESPACE namespace_guid
```

Parameter untuk menggunakan CREATE DATABASE dengan datashare

DARI DATASHARE

Kata kunci yang menunjukkan di mana datashare berada.

datahare_name

Nama datashare tempat basis data konsumen dibuat.

DENGAN IZIN

Menentukan bahwa database yang dibuat dari datashare memerlukan izin tingkat objek untuk mengakses objek database individual. Tanpa klausa ini, pengguna atau peran yang diberikan izin USAGE pada database akan secara otomatis memiliki akses ke semua objek database dalam database.

NAMESPACE namespace_guid

Nilai yang menentukan namespace produsen milik datashare.

Akun akun_id

Nilai yang menentukan akun produsen milik datashare.

Catatan penggunaan untuk CREATE DATABASE untuk berbagi data

Sebagai superuser database, saat Anda menggunakan CREATE DATABASE untuk membuat database dari data share di dalam AWS akun, tentukan opsi NAMESPACE. Opsi AKUN adalah opsional. Saat Anda menggunakan CREATE DATABASE untuk membuat database dari data share di seluruh AWS akun, tentukan ACCOUNT dan NAMESPACE dari produsen.

Anda hanya dapat membuat satu database konsumen untuk satu datashare pada cluster konsumen. Anda tidak dapat membuat beberapa database konsumen yang merujuk ke datashare yang sama.

BUAT DATABASE dari AWS Glue Data Catalog

Untuk membuat database menggunakan AWS Glue database ARN, tentukan ARN dalam perintah CREATE DATABASE Anda.

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA;
```

Secara opsional, Anda juga dapat memberikan nilai ke dalam parameter IAM_ROLE. Untuk informasi selengkapnya tentang parameter dan nilai yang diterima, lihat Parameter.

Berikut ini adalah contoh yang menunjukkan cara membuat database dari ARN menggunakan peran IAM.

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA
 IAM_ROLE <iam-role-arn>
```

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA
 IAM_ROLE default;
```

Anda juga dapat membuat database menggunakan SKEMA KATALOG DATA.

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH DATA CATALOG SCHEMA
 <sample_schema> IAM_ROLE default;
```

Buat database untuk menerima hasil integrasi Zero-ETL

Untuk membuat database menggunakan identitas integrasi Zero-ETL, tentukan perintah CRE `integration_id` ATE DATABASE Anda.

```
CREATE DATABASE destination_db_name FROM INTEGRATION 'integration_id';
```

Misalnya, pertama, ambil id integrasi dari SVV_INTEGRATION;

```
SELECT integration_id FROM SVV_INTEGRATION;
```

Kemudian gunakan salah satu id integrasi yang diambil untuk membuat database yang menerima integrasi Zero-ETL.

```
CREATE DATABASE sampledb FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111';
```

Ketika database sumber integrasi Zero-ETL diperlukan, maka, misalnya, tentukan.

```
CREATE DATABASE sampledb FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'
 DATABASE sourcedb;
```

Anda juga dapat mengatur interval refresh untuk database. Misalnya, untuk mengatur interval refresh menjadi 7.200 detik untuk data dari sumber integrasi Zero-ETL:

```
CREATE DATABASE myacct_mysql FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'
 SET REFRESH_INTERVAL 7200;
```

Kueri tampilan katalog SVV_INTEGRATION untuk informasi tentang integrasi Zero-ETL, seperti, integration_id, target_database, source, refresh _interval, dan banyak lagi.

```
SELECT * FROM svv_integration;
```

Contoh berikut membuat database dari integrasi dengan mode riwayat aktif.

```
CREATE DATABASE sample_integration_db FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111' SET HISTORY_MODE = true;
```

Batas BUAT DATABASE

Amazon Redshift memberlakukan batasan ini untuk database:

- Maksimal 60 database yang ditentukan pengguna per cluster.
- Maksimal 127 byte untuk nama database.
- Nama database tidak bisa menjadi kata yang dicadangkan.

Kolasi database

Kolasi adalah seperangkat aturan yang mendefinisikan bagaimana mesin database membandingkan dan mengurutkan data tipe karakter dalam SQL. Case-insensitive kolasi adalah kolasi yang paling umum digunakan. Amazon Redshift menggunakan kolasi yang tidak peka huruf besar/kecil untuk memfasilitasi migrasi dari sistem gudang data lainnya. Dengan dukungan asli dari kolasi peka huruf besar-kecil, Amazon Redshift terus menggunakan metode penyetelan atau pengoptimalan penting, seperti kunci distribusi, kunci pengurutan, atau pemindaian terbatas jangkauan.

Klausa COLLATE menentukan kolasi default untuk semua kolom CHAR dan VARCHAR dalam database. Jika CASE_INSENSITIVE ditentukan, semua kolom CHAR atau VARCHAR menggunakan kolasi case-insensitive. Untuk informasi tentang kolasi, lihatUrutan kolasi.

Data yang disisipkan atau dicerna dalam kolom yang tidak peka huruf besar/kecil akan mempertahankan huruf besar aslinya. Tetapi semua operasi string berbasis perbandingan termasuk penyortiran dan pengelompokan tidak peka huruf besar-kecil. Operasi pencocokan pola seperti predikat LIKE, mirip dengan, dan fungsi ekspresi reguler juga tidak peka huruf besar/kecil.

Operasi SQL berikut mendukung semantik kolasi yang berlaku:

- Operator perbandingan: **=**, **<>**, **<**, **<=**, **>**, **>=**.

- Operator LIKE
- Klausul ORDER BY
- KLAUSUL GROUP BY
- Fungsi agregat yang menggunakan perbandingan string, seperti MIN dan MAX dan LISTAGG
- Fungsi jendela, seperti klausa PARTITION BY dan klausa ORDER BY
- Fungsi skalar terbesar () dan paling kecil (), STRPOS (), REGEXP_COUNT (), REGEXP_REPLACE (), REGEXP_INSTR (), REGEXP_SUBSTR ()
- Klausa yang berbeda
- UNION, INTERSECT dan KECUALI
- DALAM DAFTAR

Untuk kueri eksternal, termasuk kueri federasi Amazon Redshift Spectrum dan Aurora PostgreSQL, pengumpulan kolom VARCHAR atau CHAR sama dengan kolasi tingkat database saat ini.

Contoh berikut menanyakan tabel Amazon Redshift Spectrum:

```
SELECT ci_varchar FROM spectrum.test_collation
WHERE ci_varchar = 'AMAZON';

ci_varchar
----------
amazon
Amazon
AMAZON
AmaZon
(4 rows)
```

Untuk informasi tentang cara membuat tabel menggunakan kolasi database, lihatCREATE TABLE.

Untuk informasi tentang fungsi COLLATE, lihatFungsi COLLATE.

Batasan kolasi database

Berikut ini adalah batasan saat bekerja dengan kolasi database di Amazon Redshift:

- Semua tabel atau tampilan sistem, termasuk tabel katalog PG dan tabel sistem Amazon Redshift peka huruf besar-kecil.

- Ketika database konsumen dan database produsen memiliki kolasi tingkat database yang berbeda, Amazon Redshift tidak mendukung kueri lintas database dan lintas cluster.
- Amazon Redshift tidak mendukung kolasi yang tidak peka huruf besar/kecil dalam kueri hanya node pemimpin.

  Contoh berikut menunjukkan kueri peka huruf besar-kecil yang tidak didukung dan kesalahan yang dikirim Amazon Redshift:

```
SELECT collate(usename, 'case_insensitive') FROM pg_user;
ERROR:  Case insensitive collation is not supported in leader node only query.
```

- Amazon Redshift tidak mendukung interaksi antara kolom peka huruf besar/kecil dan tidak peka huruf besar/kecil, seperti operasi perbandingan, fungsi, gabungan, atau set.

  Contoh berikut menunjukkan kesalahan saat kolom peka huruf besar-kecil dan tidak peka huruf besar-kecil berinteraksi:

```
CREATE TABLE test
  (ci_col varchar(10) COLLATE case_insensitive,
   cs_col varchar(10) COLLATE case_sensitive,
   cint int,
   cbigint bigint);
```

```
SELECT ci_col = cs_col FROM test;
ERROR:  Query with different collations is not supported yet.
```

```
SELECT concat(ci_col, cs_col) FROM test;
ERROR:  Query with different collations is not supported yet.
```

```
SELECT ci_col FROM test UNION SELECT cs_col FROM test;
ERROR:  Query with different collations is not supported yet.
```

```
SELECT * FROM test a, test b WHERE a.ci_col = b.cs_col;
ERROR:  Query with different collations is not supported yet.
```

```
Select Coalesce(ci_col, cs_col) from test;
ERROR:  Query with different collations is not supported yet.
```

```
Select case when cint > 0 then ci_col else cs_col end from test;
ERROR:  Query with different collations is not supported yet.
```

Untuk membuat kueri ini berfungsi, gunakan fungsi COLLATE untuk mengonversi kolasi satu kolom agar cocok dengan yang lain. Untuk informasi selengkapnya, lihat Fungsi COLLATE.

Contoh

Membuat basis data

Contoh berikut membuat database bernama TICKIT dan memberikan kepemilikan kepada pengguna DWUSER.

```
create database tickit
with owner dwuser;
```

Untuk melihat detail tentang database, kueri tabel katalog PG_DATABASE_INFO.

```
select datname, datdba, datconnlimit
from pg_database_info
where datdba > 1;

 datname     | datdba | datconnlimit
-------------+--------+-------------
 admin       |    100 | UNLIMITED
 reports     |    100 | 100
 tickit      |    100 | 100
```

Contoh berikut membuat database bernama **`sampledb`** dengan tingkat isolasi SNAPSHOT.

```
CREATE DATABASE sampledb ISOLATION LEVEL SNAPSHOT;
```

Contoh berikut membuat database sales_db dari datashare salesshare.

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Contoh kolasi database

Membuat database yang tidak peka huruf besar/kecil

Contoh berikut membuat `sampledb` database, membuat `T1` tabel, dan menyisipkan data ke dalam `T1` tabel.

```
create database sampledb collate case_insensitive;
```

Hubungkan ke database baru yang baru saja Anda buat menggunakan klien SQL Anda. Saat menggunakan editor kueri Amazon Redshift v2, pilih `sampledb` di Editor. Saat menggunakan RSQL, gunakan perintah seperti berikut ini.

```
\connect sampledb;
```

```
CREATE TABLE T1 (
  col1 Varchar(20) distkey sortkey
);
```

```
INSERT INTO T1 VALUES ('bob'), ('john'), ('Mary'), ('JOHN'), ('Bob');
```

Kemudian kueri menemukan hasil dengan`John`.

```
SELECT * FROM T1 WHERE col1 = 'John';

 col1
 ------
 john
 JOHN
(2 row)
```

Memesan dalam urutan yang tidak peka huruf besar/kecil

Contoh berikut menunjukkan urutan case-insensitive dengan tabel T1. Urutan `Bob` dan `bob` atau `John` dan `john` tidak deterministik karena mereka sama dalam kolom yang tidak peka huruf besar-kecil.

```
SELECT * FROM T1 ORDER BY 1;
```

```
 col1
 ------
 bob
 Bob
 JOHN
 john
 Mary
(5 rows)
```

Demikian pula, contoh berikut menunjukkan urutan yang tidak peka huruf besar-kecil dengan klausa GROUP BY. Bob dan bob sama dan termasuk dalam kelompok yang sama. Tidak deterministik mana yang muncul dalam hasilnya.

```
SELECT col1, count(*) FROM T1 GROUP BY 1;

 col1 | count
 -----+------
 Mary |  1
 bob  |  2
 JOHN |  2
(3 rows)
```

Kueri dengan fungsi jendela pada kolom yang tidak peka huruf besar/kecil

Contoh berikut menanyakan fungsi jendela pada kolom yang tidak peka huruf besar/kecil.

```
SELECT col1, rank() over (ORDER BY col1) FROM T1;

 col1 | rank
 -----+------
 bob  |   1
 Bob  |   1
 john |   3
 JOHN |   3
 Mary |   5
(5 rows)
```

Kueri dengan kata kunci DISTINCT

Contoh berikut menanyakan T1 tabel dengan kata kunci DISTINCT.

```
SELECT DISTINCT col1 FROM T1;

 col1
 ------
 bob
 Mary
 john
(3 rows)
```

Menanyakan dengan klausa UNION

Contoh berikut menunjukkan hasil dari UNION tabel T1 danT2.

```
CREATE TABLE T2 AS SELECT * FROM T1;
```

```
SELECT col1 FROM T1 UNION SELECT col1 FROM T2;

 col1
 ------
 john
 bob
 Mary
(3 rows)
```

BUAT DATASHARE

Membuat datashare baru di database saat ini. Pemilik datashare ini adalah penerbit perintah CREATE DATASHARE.

Amazon Redshift mengaitkan setiap datashare dengan satu database Amazon Redshift. Anda hanya dapat menambahkan objek dari database terkait ke datashare. Anda dapat membuat beberapa data share pada database Amazon Redshift yang sama.

Untuk informasi tentang berbagi data, lihat. Berbagi data di Amazon Redshift

Untuk melihat informasi tentang berbagi data, gunakan. TAMPILKAN DATA SHARE

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE DATASHARE:

- Superuser
- Pengguna dengan hak istimewa CREATE DATASHARE
- Pemilik database

Sintaksis

```
CREATE DATASHARE datashare_name
[[SET] PUBLICACCESSIBLE [=] TRUE | FALSE ];
```

Parameter

datahare_name

Nama datashare. Nama datashare harus unik di namespace cluster.

[[SET] DAPAT DIAKSES PUBLIK]

Klausa yang menentukan apakah datashare dapat dibagikan ke cluster yang dapat diakses publik.

Nilai default-nya `SET PUBLICACCESSIBLE` is `FALSE`.

Catatan penggunaan

Secara default, pemilik datashare hanya memiliki saham tetapi bukan objek dalam share.

Hanya pengguna super dan pemilik database yang dapat menggunakan CREATE DATASHARE dan mendelegasikan hak ALTER ke pengguna atau grup lain.

Contoh

Contoh berikut membuat datashare`salesshare`.

```
CREATE DATASHARE salesshare;
```

Contoh berikut membuat datashare `demoshare` yang AWS Data Exchange mengelola.

```
CREATE DATASHARE demoshare SET PUBLICACCESSIBLE TRUE, MANAGEDBY ADX;
```

BUAT FUNGSI EKSTERNAL

Membuat fungsi skalar yang ditentukan pengguna (UDF) berdasarkan AWS Lambda untuk Amazon Redshift. Untuk informasi selengkapnya tentang fungsi yang ditentukan pengguna Lambda, lihat. UDF Lambda Skalar

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE EXTERNAL FUNCTION:

- Superuser
- Pengguna dengan hak istimewa CREATE [OR REPLACE] EXTERNAL FUNCTION

Sintaksis

```
CREATE [ OR REPLACE ] EXTERNAL FUNCTION external_fn_name ( [data_type] [, ...] )
RETURNS data_type
{ VOLATILE | STABLE }
LAMBDA 'lambda_fn_name'
IAM_ROLE { default | ‘arn:aws:iam::<Akun AWS-id>:role/<role-name>’
RETRY_TIMEOUT milliseconds
MAX_BATCH_ROWS count
MAX_BATCH_SIZE size [ KB | MB ];
```

Parameter

ATAU GANTI

Klausa yang menentukan bahwa jika fungsi dengan nama dan tipe data argumen input yang sama, atau tanda tangan, seperti yang ini sudah ada, fungsi yang ada diganti. Anda hanya dapat mengganti fungsi dengan fungsi baru yang mendefinisikan kumpulan tipe data yang identik. Anda harus menjadi superuser untuk mengganti fungsi.

Jika Anda mendefinisikan fungsi dengan nama yang sama dengan fungsi yang ada tetapi tanda tangan yang berbeda, Anda membuat fungsi baru. Dengan kata lain, nama fungsi kelebihan beban. Untuk informasi selengkapnya, lihat Membebani nama fungsi.

nama_fn_eksternal

Nama fungsi eksternal. Jika Anda menentukan nama skema (seperti myschema.myfunction), fungsi dibuat menggunakan skema yang ditentukan. Jika tidak, fungsi dibuat dalam skema saat ini. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

Sebaiknya awali semua nama UDF dengan`f_`. Amazon Redshift menyimpan `f_` awalan untuk nama UDF. Dengan menggunakan `f_` awalan, Anda membantu memastikan bahwa nama UDF Anda tidak akan bertentangan dengan nama fungsi SQL bawaan untuk Amazon Redshift sekarang atau di masa mendatang. Untuk informasi selengkapnya, lihat Mencegah konflik penamaan UDF.

tipe_data

Jenis data untuk argumen input. Untuk informasi selengkapnya, lihat UDF Python skalar dan UDF Lambda Skalar.

RETURNS data_type

Jenis data dari nilai yang dikembalikan oleh fungsi. Tipe data RETURNS dapat berupa tipe data Amazon Redshift standar apa pun. Untuk informasi selengkapnya, lihat UDF Python skalar dan UDF Lambda Skalar.

MUDAH MENGUAP | STABIL

Menginformasikan pengoptimal kueri tentang volatilitas fungsi.

Untuk mendapatkan pengoptimalan terbaik, beri label pada fungsi Anda dengan kategori volatilitas paling ketat yang berlaku untuk itu. Dalam urutan ketegasan, dimulai dengan yang paling tidak ketat, kategori volatilitas adalah sebagai berikut:

- MUDAH MENGUAP
- STABIL

MUDAH MENGUAP

Dengan argumen yang sama, fungsi dapat mengembalikan hasil yang berbeda pada panggilan berturut-turut, bahkan untuk baris dalam satu pernyataan. Pengoptimal kueri tidak dapat membuat asumsi tentang perilaku fungsi volatile. Kueri yang menggunakan fungsi volatile harus mengevaluasi kembali fungsi untuk setiap input.

STABIL

Dengan argumen yang sama, fungsi dijamin mengembalikan hasil yang sama pada panggilan berturut-turut yang diproses dalam satu pernyataan. Fungsi dapat mengembalikan hasil yang berbeda ketika dipanggil dalam pernyataan yang berbeda. Kategori ini membuatnya sehingga pengoptimal dapat mengurangi berapa kali fungsi dipanggil dalam satu pernyataan.

Perhatikan bahwa jika ketegasan yang dipilih tidak valid untuk fungsi, ada risiko bahwa pengoptimal mungkin melewatkan beberapa panggilan berdasarkan keketatan ini. Ini dapat mengakibatkan set hasil yang salah.

Klausa IMMUTABLE saat ini tidak didukung untuk UDF Lambda.

LAMBDA 'lambda_fn_name'

Nama fungsi yang dipanggil Amazon Redshift.

Untuk langkah-langkah membuat AWS Lambda fungsi, lihat Membuat fungsi Lambda dengan konsol di Panduan Peng AWS Lambda embang.

Untuk informasi mengenai izin yang diperlukan untuk fungsi Lambda, lihat AWS Lambda izin di Panduan Peng AWS Lambda embang.

IAM_ROLE {default | 'arn:aws:iam: :rol/ '*<Akun AWS-id><role-name>*

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREATE EXTERNAL FUNCTION dijalankan.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Perintah CREATE EXTERNAL FUNCTION diotorisasi untuk memanggil fungsi Lambda melalui peran IAM ini. Jika cluster Anda memiliki peran IAM yang ada dengan izin untuk memanggil fungsi Lambda yang terlampir, Anda dapat mengganti ARN peran Anda. Untuk informasi selengkapnya, lihat Mengkonfigurasi parameter otorisasi untuk Lambda UDF.

Berikut ini menunjukkan sintaks untuk parameter IAM_ROLE.

```
IAM_ROLE 'arn:aws:iam::aws-account-id:role/role-name'
```

RETRY_TIMEOUT milidetik

Jumlah total waktu dalam milidetik yang digunakan Amazon Redshift untuk penundaan balik percobaan ulang.

Alih-alih mencoba lagi segera untuk kueri yang gagal, Amazon Redshift melakukan backoff dan menunggu sejumlah waktu tertentu di antara percobaan ulang. Kemudian Amazon Redshift mencoba lagi permintaan untuk menjalankan kembali kueri yang gagal sampai jumlah semua penundaan sama dengan atau melebihi nilai RETRY_TIMEOUT yang Anda tentukan. Nilai defaultnya adalah 20.000 milidetik.

Saat fungsi Lambda dipanggil, Amazon Redshift mencoba lagi untuk kueri yang menerima kesalahan sepertiTooManyRequestsException,, dan. EC2ThrottledException ServiceException

Anda dapat mengatur parameter RETRY_TIMEOUT ke 0 milidetik untuk mencegah percobaan ulang untuk UDF Lambda.

Jumlah MAX_BATCH_ROWS

Jumlah maksimum baris yang dikirim Amazon Redshift dalam permintaan batch tunggal untuk satu pemanggilan lambda.

Nilai minimum parameter ini adalah 1. Nilai maksimum adalah INT_MAX, atau 2.147.483.647.

Parameter ini bersifat opsional. Nilai defaultnya adalah INT_MAX, atau 2.147.483.647.

Ukuran BATCH_MAKSIMUM [KB | MB]

Ukuran maksimum muatan data yang dikirim Amazon Redshift dalam permintaan batch tunggal untuk satu pemanggilan lambda.

Nilai minimum parameter ini adalah 1 KB. Nilai maksimum adalah 5 MB.

Nilai default parameter ini adalah 5 MB.

KB dan MB bersifat opsional. Jika Anda tidak menyetel unit pengukuran, Amazon Redshift secara default menggunakan KB.

Catatan penggunaan

Pertimbangkan hal berikut saat Anda membuat UDF Lambda:

- Urutan panggilan fungsi Lambda pada argumen input tidak tetap atau dijamin. Ini mungkin bervariasi antara instance kueri yang sedang berjalan, tergantung pada konfigurasi cluster.

- Fungsi tidak dijamin akan diterapkan pada setiap argumen input sekali dan hanya sekali. Interaksi antara Amazon Redshift dan AWS Lambda mungkin menyebabkan panggilan berulang dengan input yang sama.

Contoh

Berikut ini adalah contoh penggunaan fungsi yang ditentukan pengguna (UDF) skalar Lambda.

Contoh UDF Scalar Lambda menggunakan fungsi Lambda Node.js

Contoh berikut menciptakan fungsi eksternal yang disebut `exfunc_sum` yang mengambil dua bilangan bulat sebagai argumen input. Fungsi ini mengembalikan jumlah sebagai output integer. Nama fungsi Lambda yang akan dipanggil adalah`lambda_sum`. Bahasa yang digunakan untuk fungsi Lambda ini adalah Node.js 12.x. Pastikan untuk menentukan peran IAM. Contoh digunakan `'arn:aws:iam::123456789012:user/johndoe'` sebagai peran IAM.

```
CREATE EXTERNAL FUNCTION exfunc_sum(INT,INT)
RETURNS INT
VOLATILE
LAMBDA 'lambda_sum'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';
```

Fungsi Lambda mengambil payload permintaan dan mengulangi setiap baris. Semua nilai dalam satu baris ditambahkan untuk menghitung jumlah untuk baris itu, yang disimpan dalam array respons. Jumlah baris dalam array hasil mirip dengan jumlah baris yang diterima dalam payload permintaan.

Payload respons JSON harus memiliki data hasil di bidang 'hasil' agar dapat dikenali oleh fungsi eksternal. Bidang argumen dalam permintaan yang dikirim ke fungsi Lambda berisi payload data. Mungkin ada beberapa baris dalam payload data jika ada permintaan batch. Fungsi Lambda berikut mengulangi semua baris dalam payload data permintaan. Ini juga secara individual mengulangi semua nilai dalam satu baris.

```
exports.handler = async (event) => {
    // The 'arguments' field in the request sent to the Lambda function contains the
 data payload.
    var t1 = event['arguments'];

    // 'len(t1)' represents the number of rows in the request payload.
    // The number of results in the response payload should be the same as the number
 of rows received.
```

```
    const resp = new Array(t1.length);

    // Iterating over all the rows in the request payload.
    for (const [i, x] of t1.entries())
    {
        var sum = 0;
        // Iterating over all the values in a single row.
        for (const y of x) {
            sum = sum + y;
        }
        resp[i] = sum;
    }
    // The 'results' field should contain the results of the lambda call.
    const response = {
        results: resp
    };
    return JSON.stringify(response);
};
```

Contoh berikut memanggil fungsi eksternal dengan nilai literal.

```
select exfunc_sum(1,2);
exfunc_sum
------------
 3
(1 row)
```

Contoh berikut membuat tabel yang disebut t_sum dengan dua kolom, c1 dan c2, dari tipe data integer dan menyisipkan dua baris data. Kemudian fungsi eksternal dipanggil dengan meneruskan nama kolom tabel ini. Dua baris tabel dikirim dalam permintaan batch dalam payload permintaan sebagai pemanggilan Lambda tunggal.

```
CREATE TABLE t_sum(c1 int, c2 int);
INSERT INTO t_sum VALUES (4,5), (6,7);
SELECT exfunc_sum(c1,c2) FROM t_sum;
 exfunc_sum
---------------
 9
 13
(2 rows)
```

Contoh UDF Scalar Lambda menggunakan atribut RETRY_TIMEOUT

Pada bagian berikut, Anda dapat menemukan contoh cara menggunakan atribut RETRY_TIMEOUT di Lambda UDF.

AWS Lambda fungsi memiliki batas konkurensi yang dapat Anda tetapkan untuk setiap fungsi. Untuk informasi selengkapnya tentang batas konkurensi, lihat M engelola konkurensi untuk fungsi Lambda di Panduan Peng AWS Lambda embang dan posting M engelola Koncurrency AWS Lambda Fungsi di Blog Komputasi. AWS

Ketika jumlah permintaan yang dilayani oleh UDF Lambda melebihi batas konkurensi, permintaan baru menerima kesalahan. `TooManyRequestsException` Lambda UDF mencoba lagi pada kesalahan ini sampai jumlah semua penundaan antara permintaan yang dikirim ke fungsi Lambda sama dengan atau melebihi nilai RETRY_TIMEOUT yang Anda tetapkan. Nilai RETRY_TIMEOUT default adalah 20.000 milidetik.

Contoh berikut menggunakan fungsi Lambda bernama`exfunc_sleep_3`. Fungsi ini mengambil payload permintaan, mengulangi setiap baris, dan mengubah input menjadi huruf besar. Kemudian tidur selama 3 detik dan mengembalikan hasilnya. Bahasa yang digunakan untuk fungsi Lambda ini adalah Python 3.8.

Jumlah baris dalam array hasil mirip dengan jumlah baris yang diterima dalam payload permintaan. Payload respons JSON harus memiliki data hasil di `results` lapangan agar dapat dikenali oleh fungsi eksternal. Bid `arguments` ang dalam permintaan yang dikirim ke fungsi Lambda berisi payload data. Dalam kasus permintaan batch, beberapa baris dapat muncul di payload data.

Batas konkurensi untuk fungsi ini secara khusus diatur ke 1 dalam konkurensi yang dicadangkan untuk menunjukkan penggunaan atribut RETRY_TIMEOUT. Ketika atribut diatur ke 1, fungsi Lambda hanya dapat melayani satu permintaan pada satu waktu.

```
import json
import time
def lambda_handler(event, context):
    t1 = event['arguments']
    # 'len(t1)' represents the number of rows in the request payload.
    # The number of results in the response payload should be the same as the number of
 rows received.
    resp = [None]*len(t1)

    # Iterating over all rows in the request payload.
    for i, x in enumerate(t1):
```

```
        # Iterating over all the values in a single row.
        for j, y in enumerate(x):
            resp[i] = y.upper()

    time.sleep(3)
    ret = dict()
    ret['results'] = resp
    ret_json = json.dumps(ret)
    return ret_json
```

Berikut ini, dua contoh tambahan menggambarkan atribut RETRY_TIMEOUT. Mereka masing-masing memanggil satu Lambda UDF. Saat memanggil Lambda UDF, setiap contoh menjalankan query SQL yang sama untuk memanggil Lambda UDF dari dua sesi database bersamaan pada saat yang sama. Ketika kueri pertama yang memanggil Lambda UDF sedang dilayani oleh UDF, kueri kedua menerima kesalahan. `TooManyRequestsException` Hasil ini terjadi karena Anda secara khusus mengatur konkurensi yang dicadangkan di UDF ke 1. Untuk informasi tentang cara menyetel konkurensi yang dicadangkan untuk fungsi Lambda, lihat Mengonfigurasi konkurensi yang dic adangkan.

Contoh pertama, berikut, menetapkan atribut RETRY_TIMEOUT untuk Lambda UDF ke 0 milidetik. Jika permintaan Lambda menerima pengecualian dari fungsi Lambda, Amazon Redshift tidak melakukan percobaan ulang apa pun. Hasil ini terjadi karena atribut RETRY_TIMEOUT diatur ke 0.

```
CREATE OR REPLACE EXTERNAL FUNCTION exfunc_upper(varchar)
RETURNS varchar
VOLATILE
LAMBDA 'exfunc_sleep_3'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test'
RETRY_TIMEOUT 0;
```

Dengan RETRY_TIMEOUT disetel ke 0, Anda dapat menjalankan dua kueri berikut dari sesi database terpisah untuk melihat hasil yang berbeda.

Query SQL pertama yang menggunakan Lambda UDF berjalan dengan sukses.

```
select exfunc_upper('Varchar');
 exfunc_upper
 --------------
 VARCHAR
(1 row)
```

Kueri kedua, yang dijalankan dari sesi database terpisah pada saat yang sama, menerima `TooManyRequestsException` kesalahan.

```
select exfunc_upper('Varchar');
ERROR:  Rate Exceeded.; Exception: TooManyRequestsException; ShouldRetry: 1
DETAIL:
-----------------------------------------------
error:  Rate Exceeded.; Exception: TooManyRequestsException; ShouldRetry: 1
code:      32103
context:query:     0
location:  exfunc_client.cpp:102
process:   padbmaster [pid=26384]
-----------------------------------------------
```

Contoh kedua, berikut, menetapkan atribut RETRY_TIMEOUT untuk Lambda UDF menjadi 3.000 milidetik. Bahkan jika kueri kedua dijalankan secara bersamaan, Lambda UDF mencoba lagi sampai total penundaan adalah 3.000 milidetik. Dengan demikian, kedua query berjalan dengan sukses.

```
CREATE OR REPLACE EXTERNAL FUNCTION exfunc_upper(varchar)
RETURNS varchar
VOLATILE
LAMBDA 'exfunc_sleep_3'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test'
RETRY_TIMEOUT 3000;
```

Dengan RETRY_TIMEOUT disetel ke 3.000 milidetik, Anda dapat menjalankan dua kueri berikut dari sesi database terpisah untuk melihat hasil yang sama.

Query SQL pertama yang menjalankan Lambda UDF berjalan dengan sukses.

```
select exfunc_upper('Varchar');
 exfunc_upper
 --------------
 VARCHAR
(1 row)
```

Kueri kedua berjalan secara bersamaan, dan Lambda UDF mencoba lagi sampai total penundaan adalah 3.000 milidetik.

```
select exfunc_upper('Varchar');
 exfunc_upper
```

```
--------------
 VARCHAR
(1 row)
```

Contoh UDF Scalar Lambda menggunakan fungsi Python Lambda

Contoh berikut membuat fungsi eksternal yang diberi nama `exfunc_multiplication` dan yang mengalikan angka dan mengembalikan bilangan bulat. Contoh ini menggabungkan keberhasilan dan `error_msg` bidang dalam respons Lambda. Bidang sukses diatur ke false ketika ada luapan bilangan bulat dalam hasil perkalian, dan `error_msg` pesan disetel ke`Integer multiplication overflow`. `exfunc_multiplication`Fungsi mengambil tiga bilangan bulat sebagai argumen input dan mengembalikan jumlah sebagai output integer.

Nama fungsi Lambda yang dipanggil adalah`lambda_multiplication`. Bahasa yang digunakan untuk fungsi Lambda ini adalah Python 3.8. Pastikan untuk menentukan peran IAM.

```
CREATE EXTERNAL FUNCTION exfunc_multiplication(int, int, int)
RETURNS INT
VOLATILE
LAMBDA 'lambda_multiplication'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';
```

Fungsi Lambda mengambil payload permintaan dan mengulangi setiap baris. Semua nilai dalam satu baris dikalikan untuk menghitung hasil untuk baris itu, yang disimpan dalam daftar respons. Contoh ini menggunakan nilai sukses Boolean yang disetel ke true secara default. Jika hasil perkalian untuk baris memiliki luapan bilangan bulat, maka nilai keberhasilan diatur ke false. Kemudian loop iterasi putus.

Saat membuat payload respons, jika nilai keberhasilan salah, fungsi Lambda berikut menambahkan `error_msg` bidang dalam payload. Ini juga menyetel pesan kesalahan ke`Integer multiplication overflow`. Jika nilai keberhasilan benar, maka data hasil ditambahkan di bidang hasil. Jumlah baris dalam array hasil, jika ada, mirip dengan jumlah baris yang diterima dalam payload permintaan.

Bidang argumen dalam permintaan yang dikirim ke fungsi Lambda berisi payload data. Mungkin ada beberapa baris dalam payload data jika ada permintaan batch. Fungsi Lambda berikut mengulangi semua baris dalam payload data permintaan dan secara individual mengulangi semua nilai dalam satu baris.

```
import json
```

```
def lambda_handler(event, context):
    t1 = event['arguments']
    # 'len(t1)' represents the number of rows in the request payload.
    # The number of results in the response payload should be the same as the number of
 rows received.
    resp = [None]*len(t1)

    # By default success is set to 'True'.
    success = True
    # Iterating over all rows in the request payload.
    for i, x in enumerate(t1):
        mul = 1
        # Iterating over all the values in a single row.
        for j, y in enumerate(x):
            mul = mul*y

        # Check integer overflow.
        if (mul >= 9223372036854775807 or mul <= -9223372036854775808):
            success = False
            break
        else:
            resp[i] = mul
    ret = dict()
    ret['success'] = success
    if not success:
        ret['error_msg'] = "Integer multiplication overflow"
    else:
        ret['results'] = resp
    ret_json = json.dumps(ret)

    return ret_json
```

Contoh berikut memanggil fungsi eksternal dengan nilai literal.

```
SELECT exfunc_multiplication(8, 9, 2);
  exfunc_multiplication
--------------------------
          144
(1 row)
```

Contoh berikut membuat tabel bernama t_multi dengan tiga kolom, c1, c2, dan c3, dari tipe data integer. Fungsi eksternal dipanggil dengan meneruskan nama kolom tabel ini. Data dimasukkan

sedemikian rupa untuk menyebabkan luapan bilangan bulat untuk menunjukkan bagaimana kesalahan disebarkan.

```
CREATE TABLE t_multi (c1 int, c2 int, c3 int);
INSERT INTO t_multi VALUES (2147483647, 2147483647, 4);
SELECT exfunc_multiplication(c1, c2, c3) FROM t_multi;
DETAIL:
  -----------------------------------------------
  error:  Integer multiplication overflow
  code:      32004context:
  context:
  query:     38
  location:  exfunc_data.cpp:276
  process:   query2_16_38 [pid=30494]
  -----------------------------------------------
```

BUAT MODEL EKSTERNAL

Topik

- Prasyarat untuk MEMBUAT MODEL EKSTERNAL
- Hak istimewa yang diperlukan
- Pengendalian biaya
- BUAT sintaks MODEL EKSTERNAL
- BUAT parameter dan pengaturan MODEL EKSTERNAL
- BUAT parameter fungsi inferensi MODEL EKSTERNAL

Prasyarat untuk MEMBUAT MODEL EKSTERNAL

Sebelum Anda menggunakan pernyataan CREATE EXTERNAL MODEL, lengkapi prasyarat diPengaturan cluster untuk menggunakan Amazon Redshift ML. Berikut ini adalah ringkasan prasyarat tingkat tinggi.

- Buat cluster Amazon Redshift dengan AWS Management Console atau AWS Command Line Interface (AWS CLI).
- Lampirkan kebijakan AWS Identitas dan Manajemen Akses (IAM) saat membuat cluster.
- Agar Amazon Redshift dan Amazon Bedrock dapat mengambil peran untuk berinteraksi dengan layanan lain, tambahkan kebijakan kepercayaan yang sesuai ke peran IAM.

- Aktifkan akses ke LLM tertentu yang ingin Anda gunakan dari konsol Amazon Bedrock.
- (Opsional) Jika Anda menemukan pengecualian pelambatan yang berasal dari Amazon Bedrock sepertiTooManyRequestsException, bahkan dengan sejumlah kecil data, periksa kuota di bawah Kuota Layanan di akun Amazon Bedrock Anda. Periksa apakah kuota tingkat akun yang diterapkan setidaknya sama dengan nilai kuota AWS default untuk InvokeModel permintaan untuk model yang Anda gunakan.

Untuk detail tentang peran IAM, kebijakan kepercayaan, dan prasyarat lainnya, lihatPengaturan cluster untuk menggunakan Amazon Redshift ML.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk MEMBUAT MODEL EKSTERNAL:

- Superuser
- Pengguna dengan hak istimewa CREATE MODEL
- Peran dengan hak istimewa GRANT CREATE MODEL

Pengendalian biaya

Amazon Redshift ML menggunakan sumber daya cluster yang ada untuk membuat model prediksi, jadi Anda tidak perlu membayar biaya tambahan. Namun, AWS biaya untuk menggunakan Amazon Bedrock berdasarkan model yang Anda pilih. Untuk informasi selengkapnya, lihat Biaya untuk menggunakan Amazon Redshift ML.

BUAT sintaks MODEL EKSTERNAL

Berikut ini adalah sintaks lengkap dari pernyataan CREATE EXTERNAL MODEL.

```
CREATE EXTERNAL MODEL model_name
FUNCTION function_name
IAM_ROLE {default/'arn:aws:iam::<account-id>:role/<role-name>'}
MODEL_TYPE BEDROCK
SETTINGS (
   MODEL_ID model_id
   [, PROMPT 'prompt prefix']
   [, SUFFIX 'prompt suffix']
   [, REQUEST_TYPE {RAW|UNIFIED}]
   [, RESPONSE_TYPE {VARCHAR|SUPER}]
```

```
);
```

Per `CREATE EXTERNAL MODEL` intah membuat fungsi inferensi yang Anda gunakan untuk menghasilkan konten.

Berikut ini adalah sintaks dari fungsi inferensi yang dibuat `CREATE EXTERNAL MODEL` menggunakan a `REQUEST_TYPE` of: `RAW`

```
SELECT inference_function_name(request_super)
[FROM table];
```

Berikut ini adalah sintaks dari fungsi inferensi yang dibuat `CREATE EXTERNAL MODEL` menggunakan a `REQUEST_TYPE` of: `UNIFIED`

```
SELECT inference_function_name(input_text, [, inference_config [,
 additional_model_request_fields]])
[FROM table];
```

Untuk informasi tentang cara menggunakan fungsi inferensi, lihatMenggunakan model eksternal untuk integrasi Amazon Redshift ML dengan Amazon Bedrock.

BUAT parameter dan pengaturan MODEL EKSTERNAL

Bagian ini menjelaskan parameter dan pengaturan untuk `CREATE EXTERNAL MODEL` perintah.

Topik

- BUAT parameter MODEL EKSTERNAL
- BUAT pengaturan MODEL EKSTERNAL

BUAT parameter MODEL EKSTERNAL

nama_model

Nama untuk model eksternal. Nama model dalam skema harus unik.

Fungsi nam a_fungsi (data_type [,...])

Nama untuk fungsi inferensi yang `CREATE EXTERNAL MODEL` menciptakan. Anda menggunakan fungsi inferensi untuk mengirim permintaan ke Amazon Bedrock dan mengambil ML-generated teks.

<account-id><role-name>IAM_ROLE {default | 'arn:aws:iam: :rol/ '}

Peran IAM yang digunakan Amazon Redshift untuk mengakses Amazon Bedrock. Untuk informasi tentang peran IAM, lihat Membuat atau memperbarui peran IAM untuk integrasi Amazon Redshift ML dengan Amazon Bedrock.

MODEL_TYPE BATUAN DASAR

Menentukan tipe model. Satu-satunya nilai yang valid adalah `BEDROCK`.

PENGATURAN (MODEL_ID model_id [,...])

Menentukan pengaturan model eksternal. Lihat bagian berikut untuk detailnya.

BUAT pengaturan MODEL EKSTERNAL

Model_ID model_id

Pengidentifikasi untuk model eksternal, misalnya,`anthropic.claude-v2`. Untuk informasi tentang ID model Amazon Bedrock, lihat ID model Amazon Bedrock.

PROMPT 'prefiks prompt'

Menentukan prompt statis yang ditambahkan Amazon Redshift ke awal setiap permintaan inferensi. Hanya didukung dengan a `REQUEST_TYPE` of`UNIFIED`.

AKHIRAN 'akhiran prompt'

Menentukan prompt statis yang ditambahkan Amazon Redshift ke akhir setiap permintaan inferensi. Hanya didukung dengan a `REQUEST_TYPE` of`UNIFIED`.

REQUEST_TYPE {MENTAH | TERPADU}

Menentukan format permintaan yang dikirim ke Amazon Bedrock. Nilai-nilai yang valid meliputi:

- RAW: Fungsi inferensi mengambil input sebagai nilai super tunggal, dan selalu mengembalikan nilai super. Format nilai super khusus untuk model Amazon Bedrock yang dipilih. Super adalah model prediksi yang menggabungkan beberapa algoritma untuk menghasilkan satu prediksi yang lebih baik.
- UNIFIED: Fungsi inferensi menggunakan API terpadu. Semua model memiliki antarmuka yang terpadu dan konsisten dengan Amazon Bedrock. Ini berfungsi untuk semua model yang mendukung pesan. Nilai ini adalah default.

  Untuk informasi selengkapnya, lihat dokumentasi API Converse di dokumentasi Amazon Bedrock API.

RESPONSE_TYPE {VARCHAR | SUPER}

Menentukan format respons. Jika `REQUEST_TYPE` ya`RAW`, `RESPONSE_TYPE` diperlukan dan satu-satunya nilai yang valid adalah`SUPER`. Untuk semua `REQUEST TYPE` nilai lainnya, nilai defaultnya adalah`VARCHAR`, dan `RESPONSE_TYPE` opsional. Nilai-nilai yang valid meliputi:

- VARCHAR: Amazon Redshift hanya mengembalikan respons teks yang dihasilkan oleh model.
- SUPER: Amazon Redshift mengembalikan seluruh respons JSON yang dihasilkan oleh model sebagai super. Ini termasuk respons teks, dan informasi seperti alasan berhenti, dan penggunaan token input dan output model. Super adalah model prediksi yang menggabungkan beberapa algoritma untuk menghasilkan satu prediksi yang lebih baik.

BUAT parameter fungsi inferensi MODEL EKSTERNAL

Bagian ini menjelaskan parameter yang valid untuk fungsi inferensi yang dibuat `CREATE EXTERNAL MODEL` perintah.

BUAT parameter fungsi inferensi MODEL EKSTERNAL untuk **`REQUEST_TYPE RAW`**

Fungsi inferensi yang dibuat dengan a `REQUEST_TYPE` of `RAW` memiliki satu argumen input super dan selalu mengembalikan tipe data super. Sintaks input super mengikuti sintaks permintaan model spesifik yang dipilih dari Amazon Bedrock.

BUAT parameter fungsi inferensi MODEL EKSTERNAL untuk **`REQUEST_TYPE dari UNIFIED`**

input_text

Teks yang dikirim Amazon Redshift ke Amazon Bedrock.

inference_config

Nilai super yang berisi parameter opsional yang dikirim Amazon Redshift ke Amazon Bedrock. Ini dapat mencakup yang berikut:

- MaxToken
- StopSequences
- suhu
- TOP

Parameter ini semuanya opsional dan semuanya peka huruf besar/kecil. Untuk informasi tentang parameter ini, lihat InferenceConfiguration di Refer ensi API Amazon Bedrock.

BUAT SKEMA EKSTERNAL

Membuat skema eksternal baru di database saat ini. Anda dapat menggunakan skema eksternal ini untuk terhubung ke Amazon RDS untuk database PostgreSQL atau Amazon Aurora PostgreSQL-Compatible Edition. Anda juga dapat membuat skema eksternal yang mereferensikan database dalam katalog data eksternal seperti AWS Glue, Athena, atau database di metastore Apache Hive, seperti Amazon EMR.

Pemilik skema ini adalah penerbit perintah CREATE EXTERNAL SCHEMA. Untuk mentransfer kepemilikan skema eksternal, gunakan ALTER SCHEMA untuk mengubah pemilik. Untuk memberikan akses ke skema ke pengguna lain atau grup pengguna, gunakan HIBAH perintah.

Anda tidak dapat menggunakan perintah GRANT atau REVOKE untuk izin pada tabel eksternal. Sebagai gantinya, berikan atau cabut izin pada skema eksternal.

> **Note**
>
> Jika saat ini Anda memiliki tabel eksternal Redshift Spectrum di katalog data Amazon Athena, Anda dapat memigrasikan katalog data Athena Anda ke. AWS Glue Data Catalog Untuk menggunakan Katalog AWS Glue Data dengan Redshift Spectrum, Anda mungkin perlu mengubah kebijakan AWS Identity and Access Management (IAM) Anda. Untuk informasi selengkapnya, lihat Meng upgrade ke Katalog AWS Glue Data di Panduan Pengguna Athena.

Untuk melihat detail skema eksternal, kueri tampilan SVV_EXTERNAL_SCHEMAS sistem.

Sintaksis

Sintaks berikut menjelaskan perintah CREATE EXTERNAL SCHEMA yang digunakan untuk mereferensikan data menggunakan katalog data eksternal. Untuk informasi selengkapnya, lihat Spektrum Pergeseran Merah Amazon.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM [ [ DATA CATALOG ] | HIVE METASTORE | POSTGRES | MYSQL | KINESIS | MSK | REDSHIFT
 | KAFKA ]
[ DATABASE 'database_name' ]
[ SCHEMA 'schema_name' ]
[ REGION 'aws-region' ]
[ IAM_ROLE [ default | 'SESSION' | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>' ] ]
```

```
[ AUTHENTICATION [ none | iam | mtls] ]
[ AUTHENTICATION_ARN 'acm-certificate-arn' | SECRET_ARN 'ssm-secret- arn' ]
[ URI ['hive_metastore_uri' [ PORT port_number ] | 'hostname' [ PORT port_number ] |
 'Kafka bootstrap URL'] ]
[ CLUSTER_ARN 'arn:aws:kafka:<region>:<Akun AWS-id>:cluster/msk/<cluster uuid>' ]
[ CATALOG_ROLE [ 'SESSION' | 'catalog-role-arn-string' ] ]
[ CREATE EXTERNAL DATABASE IF NOT EXISTS ]
[ CATALOG_ID 'Amazon Web Services account ID containing Glue or Lake Formation
 database' ]
```

Sintaks berikut menjelaskan perintah CREATE EXTERNAL SCHEMA yang digunakan untuk mereferensikan data menggunakan kueri federasi ke RDS POSTGRES atau Aurora PostgreSQL. Anda juga dapat membuat skema eksternal yang mereferensikan sumber streaming, seperti Kinesis Data Streams. Untuk informasi selengkapnya, lihat Mengkueri data dengan kueri federasi di Amazon Redshift.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM POSTGRES
DATABASE 'federated_database_name' [SCHEMA 'schema_name']
URI 'hostname' [ PORT port_number ]
IAM_ROLE [ default | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>' ]
SECRET_ARN 'ssm-secret-arn'
```

Sintaks berikut menjelaskan perintah CREATE EXTERNAL SCHEMA yang digunakan untuk mereferensikan data menggunakan kueri federasi ke RDS MySQL atau Aurora MySQL. Untuk informasi selengkapnya, lihat Mengkueri data dengan kueri federasi di Amazon Redshift.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM MYSQL
DATABASE 'federated_database_name'
URI 'hostname' [ PORT port_number ]
IAM_ROLE [ default | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>' ]
SECRET_ARN 'ssm-secret-arn'
```

Sintaks berikut menjelaskan perintah CREATE EXTERNAL SCHEMA yang digunakan untuk mereferensikan data dalam aliran Kinesis. Untuk informasi selengkapnya, lihat Streaming konsumsi ke tampilan yang terwujud.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] schema_name
FROM KINESIS
```

```
IAM_ROLE [ default | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>' ]
```

Sintaks berikut menjelaskan perintah CREATE EXTERNAL SCHEMA yang digunakan untuk mereferensikan Amazon Managed Streaming untuk Apache Kafka atau cluster Confluent Cloud dan topiknya untuk diserap. Untuk terhubung, Anda memberikan URI broker. Untuk informasi selengkapnya, lihat Streaming konsumsi ke tampilan yang terwujud.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] schema_name
FROM KAFKA
[ IAM_ROLE [ default | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>' ] ]
URI 'Kafka bootstrap URI'
AUTHENTICATION [ none | iam | mtls ]
[ AUTHENTICATION_ARN 'acm-certificate-arn' | SECRET_ARN 'ssm-secret- arn' ];
```

Sintaks berikut menjelaskan perintah CREATE EXTERNAL SCHEMA yang digunakan untuk mereferensikan data menggunakan query lintas database.

```
CREATE EXTERNAL SCHEMA local_schema_name
FROM  REDSHIFT
DATABASE 'redshift_database_name' SCHEMA 'redshift_schema_name'
```

Parameter

JIKA TIDAK ADA

Klausa yang menunjukkan bahwa jika skema yang ditentukan sudah ada, perintah tidak boleh membuat perubahan dan mengembalikan pesan bahwa skema itu ada, daripada diakhiri dengan kesalahan. Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika CREATE EXTERNAL SCHEMA mencoba membuat skema yang sudah ada.

nama_skema lokal

Nama skema eksternal baru. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

DARI [KATALOG DATA] | HIVE METASTORE | POSTGRES | MYSQL | KINESIS | MSK | PERGESERAN MERAH

Kata kunci yang menunjukkan di mana basis data eksternal berada.

KATALOG DATA menunjukkan bahwa database eksternal didefinisikan dalam katalog data Athena atau. AWS Glue Data Catalog

Jika database eksternal didefinisikan dalam Katalog Data eksternal di Wilayah yang berbeda AWS , parameter REGION diperlukan. KATALOG DATA adalah default.

HIVE METASTORE menunjukkan bahwa database eksternal didefinisikan dalam metastore Apache Hive. Jika HIVE METASTORE, ditentukan, URI diperlukan.

POSTGRES menunjukkan bahwa database eksternal didefinisikan dalam RDS PostgreSQL atau Aurora PostgreSQL.

MYSQL menunjukkan bahwa database eksternal didefinisikan dalam RDS MySQL atau Aurora MySQL.

KINESIS menunjukkan bahwa sumber data adalah aliran dari Kinesis Data Streams.

MSK menunjukkan bahwa sumber data adalah cluster Amazon MSK yang disediakan atau tanpa server.

KAFKA menunjukkan bahwa sumber data adalah cluster Kafka. Anda dapat menggunakan kata kunci ini untuk Amazon MSK dan Confluent Cloud.

DARI PERGESERAN MERAH

Kata kunci yang menunjukkan bahwa database terletak di Amazon Redshift.

DATABASE 'redshift_database_name' SKEMA 'redshift_schema_name'

Nama database Amazon Redshift.

Redshift_schema_name menunjukkan skema di Amazon Redshift. Redshift_schema_name default adalah. `public`

DATABASE 'federat ed_database_nama'

Kata kunci yang menunjukkan nama database eksternal dalam mesin database PostgreSQL atau MySQL yang didukung.

[SKEMA 'nama skema']

S chema_name menunjukkan skema dalam mesin database PostgreSQL yang didukung. Nama schema_ default adalah. `public`

Anda tidak dapat menentukan SKEMA saat Anda mengatur kueri federasi ke mesin database MySQL yang didukung.

WILAYAH 'aws- region'

Jika database eksternal didefinisikan dalam katalog data Athena atau AWS Glue Data Catalog, AWS Wilayah tempat database berada. Parameter ini diperlukan jika database didefinisikan dalam Katalog Data eksternal.

URI ['hive_metastore_uri' [PORT port_number] | 'nama host' [nomor port_port] | 'URI bootstrap Kafka']

Nama host URI dan port_number dari mesin database PostgreSQL atau MySQL yang didukung. Nama host adalah simpul kepala dari set replika. Titik akhir harus dapat dijangkau (dapat dirutekan) dari cluster Amazon Redshift. PostgreSQL port_number default adalah 5432. Port_number MySQL default adalah 3306.

> **Note**
>
> Mesin database PostgreSQL atau MySQL yang didukung harus berada di VPC yang sama dengan cluster Amazon Redshift Anda dengan grup keamanan yang menghubungkan Amazon Redshift dan URL RDS-RSPostgreSQL atau Aurora PostgreSQL. Selain itu, Anda dapat menggunakan perutean VPC yang disempurnakan untuk mengonfigurasi kasus penggunaan lintas-VPC. Untuk informasi selengkapnya, lihat titik Redshift-managed akhir VPC.

Menentukan URI metastor sarang

Jika database berada di metastore Hive, tentukan URI dan secara opsional nomor port untuk metastore. Nomor port default adalah 9083.

URI tidak berisi spesifikasi protokol (“http://”). Contoh URI yang valid:`uri '172.10.10.10'`.

Menentukan URI broker untuk penyerapan streaming

Termasuk URI bootstrap broker menyediakan kemampuan untuk terhubung ke cluster Amazon MSK atau Confluent Cloud dan menerima data streaming. Untuk informasi selengkapnya dan untuk melihat contoh, lihat Mem ulai penyerapan streaming dari Amazon Managed Streaming untuk Apache Kafka.

IAM_ROLE [default | 'SESI' | 'arn:aws:iam: :role/ '] *<Akun AWS-id> <role-name>*

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREATE EXTERNAL SCHEMA berjalan.

Gunakan `'SESSION'` jika Anda terhubung ke cluster Amazon Redshift menggunakan identitas federasi dan mengakses tabel dari skema eksternal yang dibuat menggunakan perintah ini. Untuk informasi selengkapnya, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift Spectrum, yang menjelaskan cara mengonfigurasi identitas federasi. Perhatikan bahwa konfigurasi ini, menggunakan `'SESSION'` sebagai pengganti ARN, dapat digunakan hanya jika skema dibuat menggunakan`DATA CATALOG`.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Minimal, peran IAM harus memiliki izin untuk melakukan operasi LIST pada bucket Amazon S3 untuk diakses dan operasi GET pada objek Amazon S3 yang berisi bucket.

Berikut ini menunjukkan sintaks untuk string parameter IAM_ROLE untuk satu ARN.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

Anda dapat mengrantai peran sehingga cluster Anda dapat mengambil peran IAM lain, mungkin milik akun lain. Anda dapat merangkai hingga 10 peran. Untuk contoh peran rantai, lihatMengrantai peran IAM di Amazon Redshift Spectrum.

Untuk peran IAM ini, lampirkan kebijakan izin IAM yang serupa dengan berikut ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:my-rds-secret-VNenFy"
        },
        {
```

```
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk langkah-langkah membuat peran IAM untuk digunakan dengan kueri federasi, lihat. Membuat peran rahasia dan IAM untuk menggunakan kueri federasi

> **Note**
>
> Jangan sertakan spasi dalam daftar peran yang dirantai.

Berikut ini menunjukkan sintaks untuk rantai tiga peran.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

SECRET_ARN 'ssm-rahasia-arn'

Nama Sumber Daya Amazon (ARN) dari rahasia mesin database PostgreSQL atau MySQL yang didukung dibuat menggunakan. AWS Secrets Manager Untuk informasi tentang cara membuat dan mengambil ARN untuk rahasia, lihat Mengel ola rahasia dengan AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna, dan Mengambil Nama Sumber Daya Amazon (ARN) dari rahasia di Amazon Redshift..

CATALOG_ROLE ['SESI' | katalog-role-arn-string]

Gunakan `'SESSION'` untuk terhubung ke cluster Amazon Redshift Anda menggunakan identitas federasi untuk otentikasi dan otorisasi ke katalog data. Untuk informasi selengkapnya tentang menyelesaikan langkah-langkah untuk identitas federasi, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift Spectrum. Perhatikan bahwa `'SESSION'` peran hanya dapat digunakan jika skema dibuat di KATALOG DATA.

Gunakan Amazon Resource Name ARN untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi untuk katalog data.

Jika CATALOG_ROLE tidak ditentukan, Amazon Redshift menggunakan IAM_ROLE yang ditentukan. Peran katalog harus memiliki izin untuk mengakses Katalog Data di AWS Glue atau Athena. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk Amazon Redshift Spectrum.

Berikut ini menunjukkan sintaks untuk string parameter CATALOG_ROLE untuk satu ARN.

```
CATALOG_ROLE 'arn:aws:iam::<aws-account-id>:role/<catalog-role>'
```

Anda dapat mengrantai peran sehingga cluster Anda dapat mengambil peran IAM lain, mungkin milik akun lain. Anda dapat merangkai hingga 10 peran. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Amazon Redshift Spectrum.

> **Note**
>
> Daftar peran yang dirantai tidak boleh menyertakan spasi.

Berikut ini menunjukkan sintaks untuk rantai tiga peran.

```
CATALOG_ROLE 'arn:aws:iam::<aws-account-id>:role/<catalog-role-1-
name>,arn:aws:iam::<aws-account-id>:role/<catalog-role-2-name>,arn:aws:iam::<aws-
account-id>:role/<catalog-role-3-name>'
```

BUAT DATABASE EKSTERNAL JIKA TIDAK ADA

Klausa yang membuat database eksternal dengan nama yang ditentukan oleh argumen DATABASE, jika database eksternal yang ditentukan tidak ada. Jika database eksternal yang ditentukan ada, perintah tidak membuat perubahan. Dalam hal ini, perintah mengembalikan pesan bahwa database eksternal ada, bukan berakhir dengan kesalahan.

> **Note**
>
> Anda tidak dapat menggunakan BUAT DATABASE EKSTERNAL JIKA TIDAK ADA dengan HIVE METASTORE.

Untuk menggunakan BUAT DATABASE EKSTERNAL JIKA TIDAK ADA dengan Katalog Data diaktifkan AWS Lake Formation, Anda memerlukan `CREATE_DATABASE` izin pada Katalog Data.

CATALOG_ID 'ID akun Amazon Web Services yang berisi database Glue atau Lake Formation'

ID akun tempat database katalog data disimpan.

`CATALOG_ID`dapat ditentukan hanya jika Anda berencana untuk terhubung ke cluster Amazon Redshift atau ke Amazon Redshift Serverless menggunakan identitas federasi untuk otentikasi dan otorisasi ke katalog data dengan menyetel salah satu dari berikut ini:

- `CATALOG_ROLE` untuk `'SESSION'`
- `IAM_ROLE`ke `'SESSION'` dan `'CATALOG_ROLE'` diatur ke defaultnya

Untuk informasi selengkapnya tentang menyelesaikan langkah-langkah untuk identitas federasi, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift Spectrum.

OTENTIKASI

Jenis otentikasi yang ditentukan untuk penyerapan streaming. Penyerapan streaming dengan jenis otentikasi berfungsi dengan Amazon Managed Streaming untuk Apache Kafka. Jen `AUTHENTICATION` is-jenisnya adalah sebagai berikut:

- none — Menentukan bahwa tidak ada otentikasi yang diperlukan. Ini sesuai dengan akses yang tidak diautentikasi pada MSK atau plaintext dengan TLS di Apache Kafka.
- iam — Menentukan otentikasi IAM. Ketika Anda memilih ini, pastikan bahwa peran IAM memiliki izin untuk otentikasi IAM. Untuk informasi selengkapnya tentang mendefinisikan skema eksternal, lihatMemulai dengan konsumsi streaming dari sumber Apache Kafka.
- mtls — Menentukan bahwa keamanan lapisan transportasi bersama menyediakan komunikasi yang aman dengan memfasilitasi otentikasi antara klien dan server. Dalam hal ini, kliennya adalah Redshift dan servernya adalah Amazon MSK. Untuk informasi selengkapnya tentang mengonfigurasi konsumsi streaming dengan MTL, lihatOtentikasi dengan MTL untuk penyerapan streaming Redshift dari sumber Apache Kafka.

OTENTIKASI_ARN

ARN AWS Certificate Manager sertifikat yang digunakan oleh Amazon Redshift untuk otentikasi mtls dengan Amazon MSK. ARN tersedia di konsol ACM ketika Anda memilih sertifikat yang dikeluarkan.

CLUSTER_ARN

Untuk penyerapan streaming, CLUSTER_ARN adalah pengidentifikasi cluster untuk Amazon Managed Streaming untuk cluster Apache Kafka tempat Anda melakukan streaming. Saat menggunakan CLUSTER_ARN, diperlukan kebijakan peran IAM yang menyertakan izin. `kafka:GetBootstrapBrokers` Opsi ini disediakan untuk kompatibilitas mundur. Saat ini, kami sarankan menggunakan opsi URI bootstrap-broker untuk terhubung ke Amazon Managed Streaming untuk cluster Apache Kafka. Untuk informasi selengkapnya, lihat Konsumsi streaming.

Catatan penggunaan

Untuk batasan saat menggunakan katalog data Athena, lihat Batas Athena di. Referensi Umum AWS

Untuk batas saat menggunakan AWS Glue Data Catalog, lihat AWS Glue B atas di Referensi Umum AWS.

Batasan ini tidak berlaku untuk metastor Hive.

Ada maksimal 9.900 skema per database. Untuk informasi selengkapnya, lihat Kuota dan batasan di Panduan Manajemen Amazon Redshift.

Untuk membatalkan pendaftaran skema, gunakan DROP SCHEMA perintah.

Untuk melihat detail skema eksternal, kueri tampilan sistem berikut:

- SVV_EXTERNAL_SCHEMAS
- SVV_EXTERNAL_TABEL
- SVV_KOLON_EKSTERNAL_

Contoh

Contoh berikut membuat skema eksternal menggunakan database dalam katalog data bernama `sampledb` di Wilayah AS Barat (Oregon). Gunakan contoh ini dengan Athena atau katalog AWS Glue data.

```
create external schema spectrum_schema
from data catalog
database 'sampledb'
region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole';
```

Contoh berikut membuat skema eksternal dan membuat database eksternal baru bernama`spectrum_db`.

```
create external schema spectrum_schema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
create external database if not exists;
```

Contoh berikut membuat skema eksternal menggunakan database metastore Hive bernama`hive_db`.

```
create external schema hive_schema
from hive metastore
database 'hive_db'
uri '172.10.10.10' port 99
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole';
```

Contoh berikut merangkai peran untuk menggunakan peran `myS3Role` untuk mengakses Amazon S3 dan penggunaan `myAthenaRole` untuk akses katalog data. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Amazon Redshift Spectrum.

```
create external schema spectrum_schema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myRedshiftRole,arn:aws:iam::123456789012:role/
myS3Role'
catalog_role 'arn:aws:iam::123456789012:role/myAthenaRole'
```

```
create external database if not exists;
```

Contoh berikut membuat skema eksternal yang mereferensikan database Aurora PostgreSQL.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] myRedshiftSchema
FROM POSTGRES
DATABASE 'my_aurora_db' SCHEMA 'my_aurora_schema'
URI 'endpoint to aurora hostname' PORT 5432
IAM_ROLE 'arn:aws:iam::123456789012:role/MyAuroraRole'
SECRET_ARN 'arn:aws:secretsmanager:us-east-2:123456789012:secret:development/
MyTestDatabase-AbCdEf'
```

Contoh berikut membuat skema eksternal untuk merujuk ke sales_db yang diimpor pada cluster konsumen.

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA 'public';
```

Contoh berikut membuat skema eksternal yang mereferensikan database Aurora MySQL.

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] myRedshiftSchema
FROM MYSQL
DATABASE 'my_aurora_db'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyAuroraRole'
SECRET_ARN 'arn:aws:secretsmanager:us-east-2:123456789012:secret:development/
MyTestDatabase-AbCdEf'
```

CREATE EXTERNAL TABLE

Membuat tabel eksternal baru dalam skema yang ditentukan. Semua tabel eksternal harus dibuat dalam skema eksternal. Jalur pencarian tidak didukung untuk skema eksternal dan tabel eksternal. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

Selain tabel eksternal yang dibuat menggunakan perintah CREATE EXTERNAL TABLE, Amazon Redshift dapat mereferensikan tabel eksternal yang ditentukan dalam AWS Lake Formation katalog AWS Glue atau metastore Apache Hive. Gunakan BUAT SKEMA EKSTERNAL perintah untuk mendaftarkan database eksternal yang ditentukan dalam katalog eksternal dan membuat tabel eksternal tersedia untuk digunakan di Amazon Redshift. Jika tabel eksternal ada di AWS Lake Formation katalog AWS Glue atau metastore Hive, Anda tidak perlu membuat tabel menggunakan

CREATE EXTERNAL TABLE. Untuk melihat tabel eksternal, kueri SVV_EXTERNAL_TABEL tampilan sistem.

Dengan menjalankan perintah CREATE EXTERNAL TABLE AS, Anda dapat membuat tabel eksternal berdasarkan definisi kolom dari kueri dan menulis hasil kueri tersebut ke Amazon S3. Hasilnya dalam format Apache Parquet atau teks terbatas. Jika tabel eksternal memiliki kunci partisi atau kunci, Amazon Redshift mempartisi file baru sesuai dengan kunci partisi tersebut dan mendaftarkan partisi baru ke katalog eksternal secara otomatis. Untuk informasi selengkapnya tentang MEMBUAT TABEL EKSTERNAL SEBAGAI, lihatCatatan penggunaan.

Anda dapat menanyakan tabel eksternal menggunakan sintaks SELECT yang sama yang Anda gunakan dengan tabel Amazon Redshift lainnya. Anda juga dapat menggunakan sintaks INSERT untuk menulis file baru ke lokasi tabel eksternal di Amazon S3. Untuk informasi selengkapnya, lihat INSERT (tabel eksternal).

Untuk membuat tampilan dengan tabel eksternal, sertakan klausa WITH NO SCHEMA BINDING dalam CREATE VIEW pernyataan.

Anda tidak dapat menjalankan CREATE EXTERNAL TABLE di dalam transaksi (BEGIN... END). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Hak istimewa yang diperlukan

Untuk membuat tabel eksternal, Anda harus menjadi pemilik skema eksternal atau superuser. Untuk mentransfer kepemilikan skema eksternal, gunakan ALTER SCHEMA untuk mengubah pemilik. Akses ke tabel eksternal dikendalikan oleh akses ke skema eksternal. Anda tidak dapat HIBAH atau MENCABUT izin pada tabel eksternal. Sebagai gantinya, berikan atau cabut PENGGUNAAN pada skema eksternal.

Memiliki Catatan penggunaan informasi tambahan tentang izin khusus untuk tabel eksternal.

Sintaksis

```
CREATE EXTERNAL TABLE
external_schema.table_name
(column_name data_type [, …] )
[ PARTITIONED BY (col_name data_type [, … ] )]
[ { ROW FORMAT DELIMITED row_format |
  ROW FORMAT SERDE 'serde_name'
  [ WITH SERDEPROPERTIES ( 'property_name' = 'property_value' [, ...] ) ] } ]
```

```
STORED AS file_format
LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
```

Berikut ini adalah sintaks untuk CREATE EXTERNAL TABLE AS.

```
CREATE EXTERNAL TABLE
external_schema.table_name
[ PARTITIONED BY (col_name [, … ] ) ]
[ ROW FORMAT DELIMITED row_format ]
STORED AS file_format
LOCATION { 's3://bucket/folder/' }
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
 AS
 { select_statement }
```

Parameter

nama_tabel skema eksternal_

Nama tabel yang akan dibuat, memenuhi syarat dengan nama skema eksternal. Tabel eksternal harus dibuat dalam skema eksternal. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

Panjang maksimum untuk nama tabel adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Anda dapat menggunakan karakter UTF-8 multibyte hingga maksimal empat byte. Amazon Redshift memberlakukan batas 9.900 tabel per cluster, termasuk tabel sementara yang ditentukan pengguna dan tabel sementara yang dibuat oleh Amazon Redshift selama pemrosesan kueri atau pemeliharaan sistem. Secara opsional, Anda dapat memenuhi syarat nama tabel dengan nama database. Dalam contoh berikut, nama database adalah`spectrum_db`, nama skema eksternal adalah`spectrum_schema`, dan nama tabel adalah`test`.

```
create external table spectrum_db.spectrum_schema.test (c1 int)
stored as parquet
location 's3://amzn-s3-demo-bucket/myfolder/';
```

Jika database atau skema yang ditentukan tidak ada, tabel tidak dibuat, dan pernyataan mengembalikan kesalahan. Anda tidak dapat membuat tabel atau tampilan di database sistem`template0`,`template1`,`padb_harvest`, atau`sys:internal`.

Nama tabel harus berupa nama unik untuk skema yang ditentukan.

Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

(tipe_ data nama_kolom)

Nama dan tipe data dari setiap kolom yang sedang dibuat.

Panjang maksimum untuk nama kolom adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Anda dapat menggunakan karakter UTF-8 multibyte hingga maksimal empat byte. Anda tidak dapat menentukan nama kolom `"$path"` atau`"$size"`. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

Secara default, Amazon Redshift membuat tabel eksternal dengan pseudocolumn dan`$path`. `$size` Anda dapat menonaktifkan pembuatan pseudocolumn untuk sesi dengan menyetel parameter `spectrum_enable_pseudo_columns` konfigurasi ke`false`. Untuk informasi selengkapnya, lihat Pseudokolom.

Jika pseudocolumn diaktifkan, jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.598. Jika pseudocolumn tidak diaktifkan, jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.600.

Jika Anda membuat “tabel lebar”, pastikan daftar kolom Anda tidak melebihi batas lebar baris untuk hasil perantara selama pemuatan dan pemrosesan kueri. Untuk informasi selengkapnya, lihat Catatan penggunaan.

Untuk perintah CREATE EXTERNAL TABLE AS, daftar kolom tidak diperlukan, karena kolom diturunkan dari query.

tipe_data

Jenis Data berikut didukung:

- SMALL (INT2)
- BILANGAN BULAT (INT, INT4)
- BIG (INT8)
- DESIMAL (NUMERIK)
- NYATA (FLOAT4)
- PRESISI GANDA (FLOAT8)

- BOOLEAN (BOOL)
- CHAR (KARAKTER)
- VARCHAR (KARAKTER BERVARIASI)
- VARBYTE (CHARACTER VARIING) - dapat digunakan dengan file data Parquet dan ORC, dan hanya dengan tabel yang tidak dipartisi.
- TANGGAL - hanya dapat digunakan dengan teks, parket, atau file data ORC, atau sebagai kolom partisi.
- TIMESTAMP

Untuk DATE, Anda dapat menggunakan format seperti yang dijelaskan berikut. Untuk nilai bulan yang diwakili menggunakan digit, format berikut didukung:

- `mm-dd-yyyy`Misalnya,`05-01-2017`. Ini adalah opsi default.
- `yyyy-mm-dd`, di mana tahun diwakili oleh lebih dari 2 digit. Misalnya, `2017-05-01`.

Untuk nilai bulan yang diwakili menggunakan singkatan tiga huruf, format berikut didukung:

- `mmm-dd-yyyy`Misalnya,`may-01-2017`. Ini adalah opsi default.
- `dd-mmm-yyyy`, di mana tahun diwakili oleh lebih dari 2 digit. Misalnya, `01-may-2017`.
- `yyyy-mmm-dd`, di mana tahun diwakili oleh lebih dari 2 digit. Misalnya, `2017-may-01`.

Untuk nilai tahun yang secara konsisten kurang dari 100, tahun dihitung dengan cara berikut:

- Jika tahun kurang dari 70, tahun dihitung sebagai tahun ditambah 2000. Misalnya, tanggal 05-01-17 dalam `mm-dd-yyyy` format diubah menjadi. `05-01-2017`
- Jika tahun kurang dari 100 dan lebih besar dari 69, tahun dihitung sebagai tahun ditambah 1900. Misalnya tanggal 05-01-89 dalam `mm-dd-yyyy` format diubah menjadi. `05-01-1989`
- Untuk nilai tahun yang diwakili oleh dua digit, tambahkan nol di depan untuk mewakili tahun dalam 4 digit.

Nilai stempel waktu dalam file teks harus dalam format`yyyy-mm-dd HH:mm:ss.SSSSSS`, seperti yang ditunjukkan nilai stempel waktu berikut:. `2017-05-01 11:30:59.000000`

Panjang kolom VARCHAR didefinisikan dalam byte, bukan karakter. Misalnya, kolom VARCHAR (12) dapat berisi 12 karakter byte tunggal atau 6 karakter dua byte. Saat Anda menanyakan tabel eksternal, hasil dipotong agar sesuai dengan ukuran kolom yang ditentukan tanpa mengembalikan kesalahan. Untuk informasi selengkapnya, lihat Penyimpanan dan rentang.

Untuk kinerja terbaik, sebaiknya tentukan ukuran kolom terkecil yang sesuai dengan data Anda. Untuk menemukan ukuran maksimum dalam byte untuk nilai dalam kolom, gunakan fungsi OCTET_ LENGTH. Contoh berikut mengembalikan ukuran maksimum nilai di kolom email.

```
select max(octet_length(email)) from users;

max
---
 62
```

DIPARTISI OLEH (col_name data_type [,...])

Klausa yang mendefinisikan tabel yang dipartisi dengan satu atau lebih kolom partisi. Direktori data terpisah digunakan untuk setiap kombinasi yang ditentukan, yang dapat meningkatkan kinerja kueri dalam beberapa keadaan. Kolom yang dipartisi tidak ada dalam data tabel itu sendiri. Jika Anda menggunakan nilai untuk col_name yang sama dengan kolom tabel, Anda mendapatkan kesalahan.

Setelah membuat tabel yang dipartisi, ubah tabel menggunakan pernyataanALTER TABLE... ADD PARTITION untuk mendaftarkan partisi baru ke katalog eksternal. Saat menambahkan partisi, Anda menentukan lokasi subfolder di Amazon S3 yang berisi data partisi.

Misalnya, jika tabel `spectrum.lineitem_part` didefinisikan dengan`PARTITIONED BY (l_shipdate date)`, jalankan perintah ALTER TABLE berikut untuk menambahkan partisi.

```
ALTER TABLE spectrum.lineitem_part ADD PARTITION (l_shipdate='1992-01-29')
LOCATION 's3://spectrum-public/lineitem_partition/l_shipdate=1992-01-29';
```

Jika Anda menggunakan CREATE EXTERNAL TABLE AS, Anda tidak perlu menjalankan ALTER TABLE... ADD PARTITION. Amazon Redshift secara otomatis mendaftarkan partisi baru di katalog eksternal. Amazon Redshift juga secara otomatis menulis data yang sesuai ke partisi di Amazon S3 berdasarkan kunci partisi atau kunci yang ditentukan dalam tabel.

Untuk melihat partisi, kueri SVV_PARTISI_EKSTERNAL tampilan sistem.

> **Note**
>
> Untuk perintah CREATE EXTERNAL TABLE AS, Anda tidak perlu menentukan tipe data kolom partisi karena kolom ini berasal dari kueri.

FORMAT BARIS DIBATASI Format baris

Sebuah klausa yang menentukan format data yang mendasarinya. Nilai yang mungkin untuk rowformat adalah sebagai berikut:

- GARIS DIAKHIRI OLEH 'pembatas'
- BIDANG DIAKHIRI OLEH 'pembatas'

Tentukan karakter ASCII tunggal untuk 'pembatas'. Anda dapat menentukan karakter ASCII yang tidak dicetak menggunakan oktal, dalam format di `'\ ddd '` mana *d* adalah digit oktal (0—7) hingga '\ 177'. Contoh berikut menentukan karakter BEL (bel) menggunakan oktal.

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\007'
```

Jika ROW FORMAT dihilangkan, format defaultnya adalah DELIMITED FIELDS TERMINATED BY '\ A' (awal judul) dan GARIS YANG DIAKHIRI OLEH'\ n '(baris baru).

FORMAT BARIS SERDE 'ser de_name' , [DENGAN SERDEPROPERTIES ('property_name' = 'property_value' [,...])]

Klausa yang menentukan format SERDE untuk data yang mendasarinya.

'nam a_serde '

Nama SerDe. Anda dapat menentukan format berikut:

- org.apache.hadoop.hive.serde2. RegexSerDe
- com.amazonaws.glue.serde. GrokSerDe
- org.apache.hadoop.hive.serde2.opencsvserdari

  Parameter ini mendukung SerDe properti berikut untuk OpenCSvserde:

  ```
  'wholeFile' = 'true'
  ```

  Atur `wholeFile` properti `true` untuk mengurai karakter baris baru (\n) dengan benar dalam string yang dikutip untuk permintaan OpenCSV.
- org.openx.data.jsonserde. JsonSerDe
  - JSON SERDE juga mendukung file Ion.
  - JSON harus dibentuk dengan baik.

- Stempel waktu dalam Ion dan JSON harus menggunakan format ISO8601.
- Parameter ini mendukung SerDe properti berikut untuk JsonSerDe:

```
'strip.outer.array'='true'
```

Memproses Ion/JSON file yang berisi satu array yang sangat besar tertutup dalam tanda kurung luar ([...]) seolah-olah berisi beberapa catatan JSON dalam array.

- com.amazon.ionhivesered. IonHiveSerDe

Format Amazon ION menyediakan teks dan format biner, selain tipe data. Untuk tabel eksternal yang mereferensikan data dalam format ION, Anda memetakan setiap kolom di tabel eksternal ke elemen yang sesuai dalam data format ION. Untuk informasi selengkapnya, lihat Amazon Ion. Anda juga perlu menentukan format input dan output.

DENGAN SERDEPROPERTIES ('property_name' = 'property_value' [,...])]

Secara opsional, tentukan nama properti dan nilai, dipisahkan dengan koma.

Jika ROW FORMAT dihilangkan, format defaultnya adalah DELIMITED FIELDS TERMINATED BY '\ A' (awal judul) dan GARIS YANG DIAKHIRI OLEH'\ n '(baris baru).

DISIMPAN SEBAGAI file_format

Format file untuk file data.

Format yang valid adalah sebagai berikut:

- PARQUET
- RCFILE (ColumnarSerDe hanya untuk data yang menggunakan, tidak LazyBinaryColumnarSerDe)
- SEQUENCEFILE
- TEXTFILE (untuk file teks, termasuk file JSON).
- ORC
- AVRO
- INPUTFORMAT 'input_format_class name' OUTPUTFORMAT 'output_format_classname'

Perintah CREATE EXTERNAL TABLE AS hanya mendukung dua format file, TEXTFILE dan PARQUET.

Untuk INPUTFORMAT dan OUTPUTFORMAT, tentukan nama kelas, seperti yang ditunjukkan contoh berikut.

```
'org.apache.hadoop.mapred.TextInputFormat'
```

LOKASI {'s3://bucket/folder/' | 's3://'} bucket/manifest_file

Jalur ke bucket atau folder Amazon S3 yang berisi file data atau file manifes yang berisi daftar jalur objek Amazon S3. Bucket harus berada di Wil AWS ayah yang sama dengan cluster Amazon Redshift. Untuk daftar Wilayah AWS yang didukung, lihat Batasan Amazon Redshift Spectrum.

Jika jalur menentukan bucket atau folder, misalnya`'s3://amzn-s3-demo-bucket/custdata/'`, Redshift Spectrum memindai file di bucket atau folder yang ditentukan dan subfolder apa pun. Redshift Spectrum mengabaikan file tersembunyi dan file yang dimulai dengan titik atau garis bawah.

Jika path menentukan file manifes, `'s3://bucket/manifest_file'` argumen harus secara eksplisit mereferensikan satu file tunggal—misalnya,. `'s3://amzn-s3-demo-bucket/manifest.txt'` Itu tidak dapat mereferensikan awalan kunci.

Manifes adalah file teks dalam format JSON yang mencantumkan URL setiap file yang akan dimuat dari Amazon S3 dan ukuran file, dalam byte. URL menyertakan nama bucket dan jalur objek lengkap untuk file. File yang ditentukan dalam manifes dapat berada di bucket yang berbeda, tetapi semua bucket harus berada di Wilayah yang sama AWS dengan cluster Amazon Redshift. Jika file terdaftar dua kali, file dimuat dua kali. Contoh berikut menunjukkan JSON untuk manifes yang memuat tiga file.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata.1", "meta": { "content_length":
 5956875 } },
    {"url":"s3://amzn-s3-demo-bucket1/custdata.2", "meta": { "content_length":
 5997091 } },
    {"url":"s3://amzn-s3-demo-bucket2/custdata.1", "meta": { "content_length":
 5978675 } }
  ]
}
```

Anda dapat membuat penyertaan file tertentu wajib. Untuk melakukan ini, sertakan `mandatory` opsi di tingkat file dalam manifes. Saat Anda menanyakan tabel eksternal dengan file wajib yang

hilang, pernyataan SELECT gagal. Pastikan bahwa semua file yang termasuk dalam definisi tabel eksternal hadir. Jika tidak semuanya ada, kesalahan muncul yang menunjukkan file wajib pertama yang tidak ditemukan. Contoh berikut menunjukkan JSON untuk manifes dengan `mandatory` opsi disetel `ketrue`.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata.1", "mandatory":true, "meta":
 { "content_length": 5956875 } },
    {"url":"s3://amzn-s3-demo-bucket1/custdata.2", "mandatory":false, "meta":
 { "content_length": 5997091 } },
    {"url":"s3://amzn-s3-demo-bucket2/custdata.1", "meta": { "content_length":
 5978675 } }
  ]
}
```

Untuk referensi file yang dibuat menggunakan UNLOAD, Anda dapat menggunakan manifes yang dibuat menggunakan MEMBONGKAR dengan parameter MANIFEST. File manifes kompatibel dengan file manifes untukSALIN dari Amazon S3, tetapi menggunakan kunci yang berbeda. Kunci yang tidak digunakan diabaikan.

PROPERTI TABEL ('property_name' = 'property_value' [,...])

Klausa yang menetapkan definisi tabel untuk properti tabel.

> **Note**
>
> Properti tabel peka huruf besar/kecil.

'compression _type'=' nilai '

Properti yang mengatur jenis kompresi yang akan digunakan jika nama file tidak berisi ekstensi. Jika Anda mengatur properti ini dan ada ekstensi file, ekstensi diabaikan dan nilai yang ditetapkan oleh properti digunakan. Nilai yang valid untuk tipe kompresi adalah sebagai berikut:

- bzip 2
- gzip
- none

- tajam

'data_cleaning_enabled' = 'benar/salah '

Properti ini mengatur apakah penanganan data aktif untuk tabel. Ketika 'data_cleaning_enabled' disetel ke true, penanganan data aktif untuk tabel. Ketika 'data_cleaning_enabled' disetel ke false, penanganan data tidak aktif untuk tabel. Berikut ini adalah daftar properti penanganan data tingkat tabel yang dikendalikan oleh properti ini:

- penanganan kolom_hitun_ketidakcocokan
- penanganan karakter tidak valid
- penanganan_limpahan numerik
- pengganti_char
- penanganan karakter surplus

Sebagai contoh, lihat Contoh penanganan data.

'invalid _char_handling'=' nilai '

Menentukan tindakan yang akan dilakukan ketika hasil kueri berisi nilai UTF-8 karakter yang tidak valid. Anda dapat menentukan tindakan berikut:

DISABLED

Tidak melakukan penanganan karakter yang tidak valid.

GAGAL

Membatalkan kueri yang mengembalikan data yang berisi UTF-8 nilai yang tidak valid.

SETELUKKAN_KE_NOL

Mengganti UTF-8 nilai yang tidak valid dengan null.

DROP_ROW

Mengganti setiap nilai dalam baris dengan null.

MENGGANTI

Mengganti karakter yang tidak valid dengan karakter pengganti yang Anda tentukan menggunakan`replacement_char`.

'replacement _char'=' karakter '

Menentukan karakter pengganti yang akan digunakan saat Anda menyet `invalid_char_handling` el ke`REPLACE`.

'numeric_overflow_handling' = 'nilai'

Menentukan tindakan yang akan dilakukan ketika data ORC berisi integer (misalnya, BIGINT atau int64) yang lebih besar dari definisi kolom (misalnya, SMALLINT atau int16). Anda dapat menentukan tindakan berikut:

DISABLED

Penanganan karakter yang tidak valid dimatikan.

GAGAL

Batalkan kueri saat data menyertakan karakter yang tidak valid.

SETELUKKAN_KE_NOL

Tetapkan karakter yang tidak valid ke null.

DROP_ROW

Atur setiap nilai dalam baris ke null.

'surplus _bytes_handling'=' nilai '

Menentukan cara menangani data yang dimuat yang melebihi panjang tipe data yang ditentukan untuk kolom yang berisi data VARBYTE. Secara default, Redshift Spectrum menetapkan nilai ke null untuk data yang melebihi lebar kolom.

Anda dapat menentukan tindakan berikut untuk dilakukan ketika kueri mengembalikan data yang melebihi panjang tipe data:

SETELUKKAN_KE_NOL

Mengganti data yang melebihi lebar kolom dengan null.

DISABLED

Tidak melakukan penanganan byte surplus.

GAGAL

Membatalkan kueri yang mengembalikan data melebihi lebar kolom.

DROP_ROW

Jatuhkan semua baris yang berisi data melebihi lebar kolom.

MEMOTONG

Menghapus karakter yang melebihi jumlah karakter maksimum yang ditentukan untuk kolom.

'surplus _char_handling'=' nilai '

Menentukan cara menangani data yang dimuat yang melebihi panjang tipe data yang ditentukan untuk kolom yang berisi VARCHAR, CHAR, atau data string. Secara default, Redshift Spectrum menetapkan nilai ke null untuk data yang melebihi lebar kolom.

Anda dapat menentukan tindakan berikut untuk dilakukan saat kueri mengembalikan data yang melebihi lebar kolom:

SETELUKKAN_KE_NOL

Mengganti data yang melebihi lebar kolom dengan null.

DISABLED

Tidak melakukan penanganan karakter surplus.

GAGAL

Membatalkan kueri yang mengembalikan data melebihi lebar kolom.

DROP_ROW

Mengganti setiap nilai dalam baris dengan null.

MEMOTONG

Menghapus karakter yang melebihi jumlah karakter maksimum yang ditentukan untuk kolom.

'column_count_mismatch_handling'='nilai'

Mengidentifikasi jika file berisi nilai lebih sedikit atau lebih untuk baris daripada jumlah kolom yang ditentukan dalam definisi tabel eksternal. Properti ini hanya tersedia untuk format file teks yang tidak terkompresi. Anda dapat menentukan tindakan berikut:

DISABLED

Penanganan ketidakcocokan jumlah kolom dimatikan.

GAGAL

Gagal dalam kueri jika ketidakcocokan jumlah kolom terdeteksi.

SETELUKKAN_KE_NOL

Isi nilai yang hilang dengan NULL dan abaikan nilai tambahan di setiap baris.

DROP_ROW

Jatuhkan semua baris yang berisi kesalahan ketidakcocokan jumlah kolom dari pemindaian.

'numrow s' = 'hitung_baris'

Properti yang menetapkan nilai numRows untuk definisi tabel. Untuk memperbarui statistik tabel eksternal secara eksplisit, atur properti numRows untuk menunjukkan ukuran tabel. Amazon Redshift tidak menganalisis tabel eksternal untuk menghasilkan statistik tabel yang digunakan pengoptimal kueri untuk menghasilkan rencana kueri. Jika statistik tabel tidak disetel untuk tabel eksternal, Amazon Redshift menghasilkan rencana eksekusi kueri berdasarkan asumsi bahwa tabel eksternal adalah tabel yang lebih besar dan tabel lokal adalah tabel yang lebih kecil.

'skip.header.line.count'=' hitung_baris '

Properti yang menetapkan jumlah baris untuk dilewati di awal setiap file sumber.

'serialization.null.format'=' '

Properti yang menentukan Spectrum harus mengembalikan `NULL` nilai ketika ada kecocokan yang tepat dengan teks yang disediakan di bidang.

'orc.schema.resolution'='type_pemetaan'

Properti yang menetapkan jenis pemetaan kolom untuk tabel yang menggunakan format data ORC. Properti ini diabaikan untuk format data lainnya.

Nilai yang valid untuk jenis pemetaan kolom adalah sebagai berikut:

- name
- posisi

Jika properti orc.schema.resolution dihilangkan, kolom dipetakan berdasarkan nama secara default. Jika orc.schema.resolution disetel ke nilai apa pun selain 'nama' atau 'posisi', kolom dipetakan berdasarkan posisi. Untuk informasi selengkapnya tentang pemetaan kolom, lihatMemetakan kolom tabel eksternal ke kolom ORC.

> **Note**
>
> Perintah COPY memetakan ke file data ORC hanya berdasarkan posisi. Properti tabel orc.schema.resolution tidak berpengaruh pada perilaku perintah COPY.

'write.parallel'='hidup/mati'

Properti yang mengatur apakah CREATE EXTERNAL TABLE AS harus menulis data secara paralel. Secara default, CREATE EXTERNAL TABLE AS menulis data secara paralel ke beberapa file, sesuai dengan jumlah irisan dalam cluster. Opsi default aktif. Saat 'write.parallel' disetel ke nonaktif, CREATE EXTERNAL TABLE AS menulis ke satu atau beberapa file data secara serial ke Amazon S3. Properti tabel ini juga berlaku untuk setiap pernyataan INSERT berikutnya ke dalam tabel eksternal yang sama.

'write.maxfilesize.mb'='ukuran'

Properti yang menetapkan ukuran maksimum (dalam MB) dari setiap file yang ditulis ke Amazon S3 oleh CREATE EXTERNAL TABLE AS. Ukuran harus berupa bilangan bulat yang valid antara 5 dan 6200. Ukuran file maksimum default adalah 6.200 MB. Properti tabel ini juga berlaku untuk setiap pernyataan INSERT berikutnya ke dalam tabel eksternal yang sama.

'write. kms.key.id'=' nilai '

Anda dapat menentukan AWS Key Management Service kunci untuk mengaktifkan Enkripsi Sisi Server (SSE) untuk objek Amazon S3, di mana nil ainya adalah salah satu dari berikut ini:

- `auto`untuk menggunakan AWS KMS kunci default yang disimpan di bucket Amazon S3.
- kms-key yang Anda tentukan untuk mengenkripsi data.

select_statement

Pernyataan yang menyisipkan satu atau lebih baris ke tabel eksternal dengan mendefinisikan kueri apa pun. Semua baris yang dihasilkan kueri ditulis ke Amazon S3 baik dalam format teks atau parket berdasarkan definisi tabel.

Contoh

Kumpulan contoh tersedia diContoh.

Catatan penggunaan

Topik ini berisi catatan penggunaan untukCREATE EXTERNAL TABLE. Anda tidak dapat melihat detail untuk tabel Amazon Redshift Spectrum menggunakan sumber daya yang sama yang Anda gunakan untuk tabel Amazon Redshift standar, seperti,, PG_CLASS PG_TABLE_DEFSTV_TBL_PERM, atau information_schema. Jika intelijen bisnis atau alat analitik Anda tidak mengenali tabel eksternal Redshift Spectrum, konfigurasikan aplikasi Anda untuk menanyakan SVV_EXTERNAL_TABEL danSVV_KOLON_EKSTERNAL_.

BUAT TABEL EKSTERNAL SEBAGAI

Dalam beberapa kasus, Anda mungkin menjalankan perintah CREATE EXTERNAL TABLE AS pada Katalog AWS Glue Data, katalog AWS Lake Formation eksternal, atau metastore Apache Hive. Dalam kasus seperti itu, Anda menggunakan peran AWS Identity and Access Management (IAM) untuk membuat skema eksternal. Peran IAM ini harus memiliki izin baca dan tulis di Amazon S3.

Jika Anda menggunakan katalog Lake Formation, peran IAM harus memiliki izin untuk membuat tabel dalam katalog. Dalam hal ini, ia juga harus memiliki izin lokasi danau data pada jalur Amazon S3 target. Peran IAM ini menjadi pemilik AWS Lake Formation tabel baru.

Untuk memastikan bahwa nama file unik, Amazon Redshift menggunakan format berikut untuk nama setiap file yang diunggah ke Amazon S3 secara default.

*<date>*_*<time>*_*<microseconds>*_*<query_id>*_*<slice-number>*_part_*<part-number>*.*<format>*.

Contohnya adalah `20200303_004509_810669_1007_0001_part_00.parquet`.

Pertimbangkan hal berikut saat menjalankan perintah CREATE EXTERNAL TABLE AS:

- Lokasi Amazon S3 harus kosong.
- Amazon Redshift hanya mendukung format PARQUET dan TEXTFILE saat menggunakan klausa STORED AS.
- Anda tidak perlu menentukan daftar definisi kolom. Nama kolom dan tipe data kolom dari tabel eksternal baru diturunkan langsung dari kueri SELECT.
- Anda tidak perlu menentukan tipe data kolom partisi dalam klausa PARTITIONED BY. Jika Anda menentukan kunci partisi, nama kolom ini harus ada dalam hasil kueri SELECT. Saat memiliki beberapa kolom partisi, urutannya dalam kueri SELECT tidak masalah. Amazon Redshift menggunakan urutan yang ditentukan dalam klausa PARTITIONED BY untuk membuat tabel eksternal.

- Amazon Redshift secara otomatis mempartisi file output ke folder partisi berdasarkan nilai kunci partisi. Secara default, Amazon Redshift menghapus kolom partisi dari file output.
- Klausa GARIS TERMINATED BY 'delimiter' tidak didukung.
- Klausa 'serde_name' ROW FORMAT SERDE tidak didukung.
- Penggunaan file manifes tidak didukung. Dengan demikian, Anda tidak dapat menentukan klausa LOCATION ke file manifes di Amazon S3.
- Amazon Redshift secara otomatis memperbarui properti tabel 'numRows' di akhir perintah.
- Properti tabel 'compression_type' hanya menerima 'none' atau 'snappy' untuk format file PARQUET.
- Amazon Redshift tidak mengizinkan klausa LIMIT di kueri SELECT luar. Sebagai gantinya, Anda dapat menggunakan klausa LIMIT bersarang.
- Anda dapat menggunakan STL_UNLOAD_LOG untuk melacak file yang ditulis ke Amazon S3 oleh setiap operasi CREATE EXTERNAL TABLE AS.

Izin untuk membuat dan menanyakan tabel eksternal

Untuk membuat tabel eksternal, pastikan bahwa Anda adalah pemilik skema eksternal atau superuser. Untuk mentransfer kepemilikan skema eksternal, gunakanALTER SCHEMA. Contoh berikut mengubah pemilik `spectrum_schema` skema menjadi`newowner`.

```
alter schema spectrum_schema owner to newowner;
```

Untuk menjalankan kueri Redshift Spectrum, Anda memerlukan izin berikut:

- Izin penggunaan pada skema
- Izin untuk membuat tabel sementara di database saat ini

Contoh berikut memberikan izin penggunaan pada skema `spectrum_schema` ke grup `spectrumusers` pengguna.

```
grant usage on schema spectrum_schema to group spectrumusers;
```

Contoh berikut memberikan izin sementara pada database `spectrumdb` ke grup `spectrumusers` pengguna.

```
grant temp on database spectrumdb to group spectrumusers;
```

Pseudokolom

Secara default, Amazon Redshift membuat tabel eksternal dengan pseudocolumn $path dan $size. Pilih kolom ini untuk melihat jalur ke file data di Amazon S3 dan ukuran file data untuk setiap baris yang dikembalikan oleh kueri. Nama kolom $path dan $size harus dibatasi dengan tanda kutip ganda. Kla usa SEL ECT * tidak mengembalikan pseudocolumn. Anda harus secara eksplisit menyertakan nama kolom $path dan $size dalam kueri Anda, seperti yang ditunjukkan contoh berikut.

```
select "$path", "$size"
from spectrum.sales_part
where saledate = '2008-12-01';
```

Anda dapat menonaktifkan pembuatan pseudocolumn untuk sesi dengan menyetel parameter konfigurasi spectrum_enable_pseudo_column ke false.

> **Important**
>
> Memilih $size atau $path dikenakan biaya karena Redshift Spectrum memindai file data di Amazon S3 untuk menentukan ukuran set hasil. Untuk informasi selengkapnya, lihat Harga Amazon Redshift.

Untuk tabel Iceberg v3, dua kolom semu tambahan _row_id dan _last_updated_sequence_number tersedia. Untuk informasi selengkapnya, lihat the section called “Garis garis keturunan”.

Mengatur opsi penanganan data

Anda dapat mengatur parameter tabel untuk menentukan penanganan input untuk data yang ditanyakan di tabel eksternal, termasuk:

- Karakter surplus dalam kolom yang berisi VARCHAR, CHAR, dan data string. Untuk informasi selengkapnya, lihat properti tabel eksternal`surplus_char_handling`.
- Karakter tidak valid dalam kolom yang berisi VARCHAR, CHAR, dan data string. Untuk informasi selengkapnya, lihat properti tabel eksternal`invalid_char_handling`.
- Karakter pengganti yang akan digunakan saat Anda menentukan REPLACE untuk properti tabel eksternal`invalid_char_handling`.

- Penanganan Cast Overflow dalam kolom yang berisi data integer dan desimal. Untuk informasi selengkapnya, lihat properti tabel eksternal`numeric_overflow_handling`.
- Surplus_bytes_handling untuk menentukan penanganan input untuk surplus byte dalam kolom yang berisi data varbyte. Untuk informasi selengkapnya, lihat properti tabel eksternal`surplus_bytes_handling`.

Contoh

Contoh berikut membuat tabel bernama SALES dalam skema eksternal Amazon Redshift bernama`spectrum`. Data ada dalam file teks yang dibatasi tab. Klausa TABLE PROPERTIES menetapkan properti numRows ke 170.000 baris.

Tergantung pada identitas yang Anda gunakan untuk menjalankan CREATE EXTERNAL TABLE, mungkin ada izin IAM yang harus Anda konfigurasi. Sebagai praktik terbaik, sebaiknya lampirkan kebijakan izin ke peran IAM dan kemudian menetapkannya ke pengguna dan grup sesuai kebutuhan. Untuk informasi selengkapnya, lihat Identitas dan manajemen akses di Amazon Redshift.

```
create external table spectrum.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
saledate date,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/'
table properties ('numRows'='170000');
```

Contoh berikut membuat tabel yang menggunakan JsonSerDe untuk mereferensikan data dalam format JSON.

```
create external table spectrum.cloudtrail_json (
event_version int,
event_id bigint,
```

```
event_time timestamp,
event_type varchar(10),
awsregion varchar(20),
event_name varchar(max),
event_source varchar(max),
requesttime timestamp,
useragent varchar(max),
recipientaccountid bigint)
row format serde 'org.openx.data.jsonserde.JsonSerDe'
with serdeproperties (
'dots.in.keys' = 'true',
'mapping.requesttime' = 'requesttimestamp'
) location 's3://amzn-s3-demo-bucket/json/cloudtrail';
```

Contoh CREATE EXTERNAL TABLE AS berikut membuat tabel eksternal yang tidak dipartisi. Kemudian menulis hasil kueri SELECT sebagai Apache Parquet ke lokasi target Amazon S3.

```
CREATE EXTERNAL TABLE spectrum.lineitem
STORED AS parquet
LOCATION 'S3://amzn-s3-demo-bucket/cetas/lineitem/'
AS SELECT * FROM local_lineitem;
```

Contoh berikut membuat tabel eksternal yang dipartisi dan menyertakan kolom partisi dalam kueri SELECT.

```
CREATE EXTERNAL TABLE spectrum.partitioned_lineitem
PARTITIONED BY (l_shipdate, l_shipmode)
STORED AS parquet
LOCATION 'S3://amzn-s3-demo-bucket/cetas/partitioned_lineitem/'
AS SELECT l_orderkey, l_shipmode, l_shipdate, l_partkey FROM local_table;
```

Untuk daftar database yang ada di katalog data eksternal, kueri SVV_EXTERNAL_DATABASE tampilan sistem.

```
select eskind,databasename,esoptions from svv_external_databases order by databasename;
```

```
eskind | databasename | esoptions
-------+--------------
+-------------------------------------------------------------------------------
     1 | default      | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

```
     1 | sampledb     | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
     1 | spectrumdb   | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

Untuk melihat detail tabel eksternal, kueri SVV_EXTERNAL_TABEL tampilan dan SVV_KOLON_EKSTERNAL_ sistem.

Contoh berikut menanyakan tampilan SVV_EXTERNAL_TABLES.

```
select schemaname, tablename, location from svv_external_tables;
```

```
schemaname | tablename            | location
-----------+----------------------
+-----------------------------------------------------
spectrum   | sales                | s3://redshift-downloads/tickit/spectrum/sales
spectrum   | sales_part           | s3://redshift-downloads/tickit/spectrum/
sales_partition
```

Contoh berikut menanyakan tampilan SVV_EXTERNAL_COLUMNS.

```
select * from svv_external_columns where schemaname like 'spectrum%' and tablename
 ='sales';
```

```
schemaname | tablename | columnname | external_type | columnnum | part_key
-----------+-----------+------------+---------------+-----------+---------
spectrum   | sales     | salesid    | int           |         1 |        0
spectrum   | sales     | listid     | int           |         2 |        0
spectrum   | sales     | sellerid   | int           |         3 |        0
spectrum   | sales     | buyerid    | int           |         4 |        0
spectrum   | sales     | eventid    | int           |         5 |        0
spectrum   | sales     | saledate   | date          |         6 |        0
spectrum   | sales     | qtysold    | smallint      |         7 |        0
spectrum   | sales     | pricepaid  | decimal(8,2)  |         8 |        0
spectrum   | sales     | commission | decimal(8,2)  |         9 |        0
spectrum   | sales     | saletime   | timestamp     |        10 |        0
```

Untuk melihat partisi tabel, gunakan kueri berikut.

```
select schemaname, tablename, values, location
```

```
from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename  | values         | location
-----------+------------+----------------
+-------------------------------------------------------------------------
spectrum   | sales_part | ["2008-01-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-01
spectrum   | sales_part | ["2008-02-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-02
spectrum   | sales_part | ["2008-03-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-04
spectrum   | sales_part | ["2008-05-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-05
spectrum   | sales_part | ["2008-06-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-06
spectrum   | sales_part | ["2008-07-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-07
spectrum   | sales_part | ["2008-08-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-08
spectrum   | sales_part | ["2008-09-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-09
spectrum   | sales_part | ["2008-10-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-10
spectrum   | sales_part | ["2008-11-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-11
spectrum   | sales_part | ["2008-12-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-12
```

Contoh berikut mengembalikan ukuran total file data terkait untuk tabel eksternal.

```
select distinct "$path", "$size"
   from spectrum.sales_part;

 $path                                                                      | $size
----------------------------------------------------------------------------+-------
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/ |  1616
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/ |  1444
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/ |  1444
```

Contoh partisi

Untuk membuat tabel eksternal yang dipartisi berdasarkan tanggal, jalankan perintah berikut.

```
create external table spectrum.sales_part(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
partitioned by (saledate date)
row format delimited
fields terminated by '|'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales_partition/'
table properties ('numRows'='170000');
```

Untuk menambahkan partisi, jalankan perintah ALTER TABLE berikut.

```
alter table spectrum.sales_part
add if not exists partition (saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-02-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-03-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-04-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-04/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-05-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-05/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-06-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-06/';
alter table spectrum.sales_part
```

```
add if not exists partition (saledate='2008-07-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-07/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-08-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-08/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-09-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-09/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-10-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-10/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-11-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-11/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-12-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-12/';
```

Untuk memilih data dari tabel yang dipartisi, jalankan kueri berikut.

```
select top 10 spectrum.sales_part.eventid, sum(spectrum.sales_part.pricepaid)
from spectrum.sales_part, event
where spectrum.sales_part.eventid = event.eventid
  and spectrum.sales_part.pricepaid > 30
  and saledate = '2008-12-01'
group by spectrum.sales_part.eventid
order by 2 desc;
```

```
eventid | sum
--------+---------
    914 | 36173.00
   5478 | 27303.00
   5061 | 26383.00
   4406 | 26252.00
   5324 | 24015.00
   1829 | 23911.00
   3601 | 23616.00
   3665 | 23214.00
   6069 | 22869.00
   5638 | 22551.00
```

Untuk melihat partisi tabel eksternal, kueri SVV_PARTISI_EKSTERNAL tampilan sistem.

```
select schemaname, tablename, values, location from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename  | values         | location
-----------+------------+----------------
+-----------------------------------------------
spectrum   | sales_part | ["2008-01-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-01
spectrum   | sales_part | ["2008-02-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-02
spectrum   | sales_part | ["2008-03-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-04
spectrum   | sales_part | ["2008-05-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-05
spectrum   | sales_part | ["2008-06-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-06
spectrum   | sales_part | ["2008-07-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-07
spectrum   | sales_part | ["2008-08-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-08
spectrum   | sales_part | ["2008-09-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-09
spectrum   | sales_part | ["2008-10-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-10
spectrum   | sales_part | ["2008-11-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-11
spectrum   | sales_part | ["2008-12-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-12
```

Contoh format baris

Berikut ini menunjukkan contoh menentukan parameter ROW FORMAT SERDE untuk file data yang disimpan dalam format AVRO.

```
create external table spectrum.sales(salesid int, listid int, sellerid int,
 buyerid int, eventid int, dateid int, qtysold int, pricepaid decimal(8,2), comment
 VARCHAR(255))
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
```

```
WITH SERDEPROPERTIES ('avro.schema.literal'='{\"namespace\": \"dory.sample\",\"name\":
 \"dory_avro\",\"type\": \"record\", \"fields\": [{\"name\":\"salesid\", \"type\":\"int
\"},
{\"name\":\"listid\", \"type\":\"int\"},
{\"name\":\"sellerid\", \"type\":\"int\"},
{\"name\":\"buyerid\", \"type\":\"int\"},
{\"name\":\"eventid\",\"type\":\"int\"},
{\"name\":\"dateid\",\"type\":\"int\"},
{\"name\":\"qtysold\",\"type\":\"int\"},
{\"name\":\"pricepaid\", \"type\": {\"type\": \"bytes\", \"logicalType\": \"decimal\",
 \"precision\": 8, \"scale\": 2}}, {\"name\":\"comment\",\"type\":\"string\"}]}')
STORED AS AVRO
location 's3://amzn-s3-demo-bucket/avro/sales' ;
```

Berikut ini menunjukkan contoh menentukan parameter ROW FORMAT SERDE menggunakan RegEx.

```
create external table spectrum.types(
cbigint bigint,
cbigint_null bigint,
cint int,
cint_null int)
row format serde 'org.apache.hadoop.hive.serde2.RegexSerDe'
with serdeproperties ('input.regex'='([^\\x01]+)\\x01([^\\x01]+)\\x01([^\\x01]+)\
\x01([^\\x01]+)')
stored as textfile
location 's3://amzn-s3-demo-bucket/regex/types';
```

Berikut ini menunjukkan contoh menentukan parameter ROW FORMAT SERDE menggunakan Grok.

```
create external table spectrum.grok_log(
timestamp varchar(255),
pid varchar(255),
loglevel varchar(255),
progname varchar(255),
message varchar(255))
row format serde 'com.amazonaws.glue.serde.GrokSerDe'
with serdeproperties ('input.format'='[DFEWI], \\[%{TIMESTAMP_ISO8601:timestamp} #
%{POSINT:pid:int}\\] *(?<loglevel>:DEBUG|FATAL|ERROR|WARN|INFO) -- +%{DATA:progname}:
 %{GREEDYDATA:message}')
stored as textfile
location 's3://DOC-EXAMPLE-BUCKET/grok/logs';
```

Berikut ini menunjukkan contoh mendefinisikan log akses server Amazon S3 di bucket S3. Anda dapat menggunakan Redshift Spectrum untuk menanyakan log akses Amazon S3.

```
CREATE EXTERNAL TABLE spectrum.mybucket_s3_logs(
bucketowner varchar(255),
bucket varchar(255),
requestdatetime varchar(2000),
remoteip varchar(255),
requester varchar(255),
requested varchar(255),
operation varchar(255),
key varchar(255),
requesturi_operation varchar(255),
requesturi_key varchar(255),
requesturi_httpprotoversion varchar(255),
httpstatus varchar(255),
errorcode varchar(255),
bytessent bigint,
objectsize bigint,
totaltime varchar(255),
turnaroundtime varchar(255),
referrer varchar(255),
useragent varchar(255),
versionid varchar(255)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
'input.regex' = '([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 \"([^ ]*)\\s*([^ ]*)\\s*([^ ]*)\" (- |[^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\") ([^ ]*).*$')
LOCATION 's3://amzn-s3-demo-bucket/s3logs’;
```

Berikut ini menunjukkan contoh menentukan parameter ROW FORMAT SERDE untuk data format ION.

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'com.amazon.ionhiveserde.IonHiveSerDe'
STORED AS
INPUTFORMAT 'com.amazon.ionhiveserde.formats.IonInputFormat'
OUTPUTFORMAT 'com.amazon.ionhiveserde.formats.IonOutputFormat'
LOCATION 's3://amzn-s3-demo-bucket/prefix'
```

Contoh penanganan data

Contoh berikut mengakses file: spi_global_rankings.csv. Anda dapat mengunggah `spi_global_rankings.csv` file ke bucket Amazon S3 untuk mencoba contoh-contoh ini.

Contoh berikut membuat skema eksternal `schema_spectrum_uddh` dan database`spectrum_db_uddh`. Untuk`aws-account-id`, masukkan ID AWS akun Anda dan untuk `role-name` masukkan nama peran Redshift Spectrum Anda.

```
create external schema schema_spectrum_uddh
from data catalog
database 'spectrum_db_uddh'
iam_role 'arn:aws:iam::aws-account-id:role/role-name'
create external database if not exists;
```

Contoh berikut membuat tabel eksternal `soccer_league` dalam skema eksternal`schema_spectrum_uddh`.

```
CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league
(
  league_rank smallint,
  prev_rank   smallint,
  club_name   varchar(15),
  league_name varchar(20),
  league_off  decimal(6,2),
  league_def  decimal(6,2),
  league_spi  decimal(6,2),
  league_nspi integer
)
ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n\l'
stored as textfile
LOCATION 's3://spectrum-uddh/league/'
table properties ('skip.header.line.count'='1');
```

Periksa jumlah baris dalam `soccer_league` tabel.

```
select count(*) from schema_spectrum_uddh.soccer_league;
```

Jumlah baris ditampilkan.

```
count
645
```

Kueri berikut menampilkan 10 klub teratas. Karena klub `Barcelona` memiliki karakter yang tidak valid dalam string, NULL ditampilkan untuk nama tersebut.

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

```
league_rank club_name league_name   league_nspi
1  Manchester City Barclays Premier Lea  34595
2  Bayern Munich German Bundesliga  34151
3  Liverpool Barclays Premier Lea  33223
4  Chelsea  Barclays Premier Lea  32808
5  Ajax  Dutch Eredivisie  32790
6  Atletico  Madrid Spanish Primera Divi 31517
7  Real Madrid Spanish Primera Divi  31469
8  NULL         Spanish Primera Divi            31321
9  RB Leipzig German Bundesliga  31014
10  Paris Saint-Ger French Ligue 1   30929
```

Contoh berikut mengubah `soccer_league` tabel untuk menentukan`invalid_char_handling`,`replacement_char`, dan properti tabel `data_cleansing_enabled` eksternal untuk menyisipkan tanda tanya (?) sebagai pengganti karakter yang tidak terduga.

```
alter  table schema_spectrum_uddh.soccer_league
set table properties
 ('invalid_char_handling'='REPLACE','replacement_char'='?','data_cleansing_enabled'='true');
```

Contoh berikut menanyakan tabel `soccer_league` untuk tim dengan peringkat dari 1 hingga 10.

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

Karena properti tabel diubah, hasilnya menunjukkan 10 klub teratas, dengan tanda tanya (?) karakter pengganti di baris kedelapan untuk klub`Barcelona`.

```
league_rank club_name league_name  league_nspi
1  Manchester City Barclays Premier Lea 34595
2  Bayern Munich German Bundesliga 34151
3  Liverpool Barclays Premier Lea 33223
4  Chelsea  Barclays Premier Lea 32808
5  Ajax  Dutch Eredivisie 32790
6  Atletico Madrid Spanish Primera Divi 31517
7  Real Madrid Spanish Primera Divi 31469
8  Barcel?na Spanish Primera Divi 31321
9  RB Leipzig German Bundesliga 31014
10  Paris Saint-Ger French Ligue 1  30929
```

Contoh berikut mengubah tabel untuk menentukan `soccer_league` properti tabel `invalid_char_handling` eksternal untuk melepaskan baris dengan karakter tak terduga.

```
alter table schema_spectrum_uddh.soccer_league
set table properties
 ('invalid_char_handling'='DROP_ROW','data_cleansing_enabled'='true');
```

Contoh berikut menanyakan tabel `soccer_league` untuk tim dengan peringkat dari 1 hingga 10.

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

Hasilnya menampilkan klub-klub top, tidak termasuk baris kedelapan untuk klub`Barcelona`.

```
league_rank   club_name         league_name            league_nspi
1             Manchester City   Barclays Premier Lea   34595
2             Bayern Munich     German Bundesliga      34151
3             Liverpool         Barclays Premier Lea   33223
4             Chelsea           Barclays Premier Lea   32808
5             Ajax              Dutch Eredivisie       32790
6             Atletico Madrid   Spanish Primera Divi   31517
7             Real Madrid       Spanish Primera Divi   31469
9             RB Leipzig        German Bundesliga      31014
10            Paris Saint-Ger   French Ligue 1         30929
```

BUAT TAMPILAN EKSTERNAL

Fitur pratinjau tampilan Katalog Data hanya tersedia di Wilayah berikut.

- AS Timur (Ohio) (us-east-2)
- AS Timur (Virginia Utara) (us-east-1)
- AS Barat (California Utara) (us-west-1)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Europe (Ireland) (eu-west-1)
- Eropa (Stockholm) (eu-north-1)

Membuat tampilan di Katalog Data. Tampilan Katalog Data adalah skema tampilan tunggal yang bekerja dengan mesin SQL lain seperti Amazon Athena dan Amazon EMR. Anda dapat menanyakan tampilan dari mesin pilihan Anda. Untuk informasi selengkapnya tentang tampilan Katalog Data, lihat Membuat tampilan Katalog Data.

Sintaksis

```
CREATE EXTERNAL VIEW schema_name.view_name [ IF NOT EXISTS ]
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
 external_schema_name.view_name}
AS query_definition;
```

Parameter

nama_skema.nama_tampilan

Skema yang dilampirkan ke AWS Glue database Anda, diikuti dengan nama tampilan.

DILINDUNGI

Menentukan bahwa perintah CREATE EXTERNAL VIEW seharusnya hanya selesai jika kueri dalam query_definition berhasil diselesaikan.

JIKA TIDAK ADA

Membuat tampilan jika tampilan belum ada.

nama_katalog.schema_nama.view_name | awsdatacatalog.dbname.view_name | nama_schema_eksternal_nama.view_name

Notasi skema yang akan digunakan saat membuat tampilan. Anda dapat menentukan untuk menggunakan AWS Glue Data Catalog, database Glue yang Anda buat, atau skema eksternal yang Anda buat. Lihat MEMBUAT DATABASE dan MEMBUAT SKEMA EKSTERNAL untuk informasi selengkapnya.

definisi kueri

Definisi kueri SQL yang dijalankan Amazon Redshift untuk mengubah tampilan.

Contoh

Contoh berikut membuat tampilan Katalog Data bernama sample_schema.glue_data_catalog_view.

```
CREATE EXTERNAL PROTECTED VIEW sample_schema.glue_data_catalog_view IF NOT EXISTS
AS SELECT * FROM sample_database.remote_table "remote-table-name";
```

CREATE FUNCTION

Membuat fungsi skalar yang ditentukan pengguna (UDF) baru menggunakan klausa SQL SELECT atau program Python.

Untuk informasi selengkapnya dan contoh tambahan, lihat User-defined fungsi di Amazon Redshift.

Hak istimewa yang diperlukan

Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan FUNGSI CREATE OR REPLACE:

- Untuk FUNGSI BUAT:
  - Superuser dapat menggunakan bahasa tepercaya dan tidak tepercaya untuk membuat fungsi.
  - Pengguna dengan hak istimewa CREATE [OR REPLACE] FUNCTION dapat membuat fungsi dengan bahasa tepercaya.
- Untuk FUNGSI GANTI:
  - Superuser
  - Pengguna dengan hak istimewa CREATE [OR REPLACE] FUNCTION

- Pemilik fungsi

Sintaksis

```
CREATE [ OR REPLACE ] FUNCTION f_function_name
( { [py_arg_name  py_arg_data_type |
sql_arg_data_type } [ , ... ] ] )
RETURNS data_type
{ VOLATILE | STABLE | IMMUTABLE }
AS $$
  { python_program | SELECT_clause }
$$ LANGUAGE { plpythonu | sql }
```

Parameter

ATAU GANTI

Menentukan bahwa jika fungsi dengan nama dan tipe data argumen input yang sama, atau tanda tangan, seperti yang ini sudah ada, fungsi yang ada diganti. Anda hanya dapat mengganti fungsi dengan fungsi baru yang mendefinisikan kumpulan tipe data yang identik. Anda harus menjadi superuser untuk mengganti fungsi.

Jika Anda mendefinisikan fungsi dengan nama yang sama dengan fungsi yang ada tetapi tanda tangan yang berbeda, Anda membuat fungsi baru. Dengan kata lain, nama fungsi kelebihan beban. Untuk informasi selengkapnya, lihat Membebani nama fungsi.

f_nama_fungsi

Nama fungsi. Jika Anda menentukan nama skema (seperti`myschema.myfunction`), fungsi dibuat menggunakan skema yang ditentukan. Jika tidak, fungsi dibuat dalam skema saat ini. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

Sebaiknya awali semua nama UDF dengan`f_`. Amazon Redshift menyimpan aw `f_` alan untuk nama UDF, jadi dengan menggunakan `f_` awalan, Anda memastikan bahwa nama UDF Anda tidak akan bertentangan dengan nama fungsi SQL bawaan Amazon Redshift yang ada atau yang akan datang. Untuk informasi selengkapnya, lihat Mencegah konflik penamaan UDF.

Anda dapat menentukan lebih dari satu fungsi dengan nama fungsi yang sama jika tipe data untuk argumen input berbeda. Dengan kata lain, nama fungsi kelebihan beban. Untuk informasi selengkapnya, lihat Membebani nama fungsi.

py_arg_nama_py_arg_data_type | sql_arg_data_type

Untuk Python UDF, daftar nama argumen input dan tipe data. Untuk SQL UDF, daftar tipe data, tanpa nama argumen. Dalam Python UDF, lihat argumen menggunakan nama argumen. Dalam SQL UDF, lihat argumen menggunakan $1, $2, dan seterusnya, berdasarkan urutan argumen dalam daftar argumen.

Untuk SQL UDF, tipe data input dan return dapat berupa tipe data Amazon Redshift standar apa pun. Untuk Python UDF, tipe data input dan return dapat berupa SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, atau TIMESTAMP. Selain itu, fungsi yang ditentukan pengguna Python (UDF) mendukung tipe data ANYELEMENT. Ini secara otomatis dikonversi ke tipe data standar berdasarkan tipe data dari argumen yang sesuai yang diberikan saat runtime. Jika beberapa argumen menggunakan ANYELEMENT, semuanya diselesaikan ke tipe data yang sama saat runtime, berdasarkan argumen ANYELEMENT pertama dalam daftar. Untuk informasi selengkapnya, lihat Jenis data Python UDF dan Jenis Data.

Anda dapat menentukan maksimum 32 argumen.

RETURNS data_type

Jenis data dari nilai yang dikembalikan oleh fungsi. Tipe data RETURNS dapat berupa tipe data Amazon Redshift standar apa pun. Selain itu, Python UDF dapat menggunakan tipe data ANYELEMENT, yang secara otomatis dikonversi ke tipe data standar berdasarkan argumen yang diberikan saat runtime. Jika Anda menentukan ANYELEMENT untuk tipe data yang dikembalikan, setidaknya satu argumen harus menggunakan ANYELEMENT. Jenis data pengembalian aktual cocok dengan tipe data yang disediakan untuk argumen ANYELEMENT ketika fungsi dipanggil. Untuk informasi selengkapnya, lihat Jenis data Python UDF.

MUDAH MENGUAP | STABIL | TIDAK BERUBAH

Menginformasikan pengoptimal kueri tentang volatilitas fungsi.

Anda akan mendapatkan pengoptimalan terbaik jika Anda memberi label fungsi Anda dengan kategori volatilitas paling ketat yang berlaku untuk itu. Namun, jika kategorinya terlalu ketat, ada risiko bahwa pengoptimal akan secara keliru melewatkan beberapa panggilan, mengakibatkan set hasil yang salah. Dalam urutan ketegasan, dimulai dengan yang paling tidak ketat, kategori volatilitas adalah sebagai berikut:

- MUDAH MENGUAP
- STABIL

- TIDAK BERUBAH

MUDAH MENGUAP

Dengan argumen yang sama, fungsi dapat mengembalikan hasil yang berbeda pada panggilan berturut-turut, bahkan untuk baris dalam satu pernyataan. Pengoptimal kueri tidak dapat membuat asumsi apa pun tentang perilaku fungsi volatile, sehingga kueri yang menggunakan fungsi volatile harus mengevaluasi kembali fungsi untuk setiap baris input.

STABIL

Dengan argumen yang sama, fungsi dijamin mengembalikan hasil yang sama untuk semua baris yang diproses dalam satu pernyataan. Fungsi dapat mengembalikan hasil yang berbeda ketika dipanggil dalam pernyataan yang berbeda. Kategori ini memungkinkan pengoptimal untuk mengoptimalkan beberapa panggilan fungsi dalam satu pernyataan ke satu panggilan untuk pernyataan tersebut.

TIDAK BERUBAH

Dengan argumen yang sama, fungsi selalu mengembalikan hasil yang sama, selamanya. Ketika kueri memanggil `IMMUTABLE` fungsi dengan argumen konstan, pengoptimal melakukan pra-evaluasi fungsi.

Pernyataan AS $$ $$

Konstruksi yang menyertakan pernyataan yang akan dijalankan. Kata kunci literal `AS $$` dan `$$` diperlukan.

Amazon Redshift mengharuskan Anda untuk melampirkan pernyataan dalam fungsi Anda dengan menggunakan format yang disebut kuotasi dolar. Apa pun di dalam kandang dilewatkan persis apa adanya. Anda tidak perlu melarikan diri dari karakter khusus karena isi string ditulis secara harfiah.

Dengan kutipan dolar, Anda menggunakan sepasang tanda dolar ($$) untuk menandakan awal dan akhir pernyataan yang akan dijalankan, seperti yang ditunjukkan pada contoh berikut.

```
$$ my statement $$
```

Secara opsional, di antara tanda-tanda dolar di setiap pasangan, Anda dapat menentukan string untuk membantu mengidentifikasi pernyataan. String yang Anda gunakan harus sama di awal dan akhir pasangan penutup. String ini peka huruf besar-kecil, dan mengikuti batasan yang sama

dengan pengidentifikasi yang tidak dikutip kecuali bahwa itu tidak dapat berisi tanda dolar. Contoh berikut menggunakan string`test`.

```
$test$ my statement $test$
```

Untuk informasi lebih lanjut tentang kutipan dolar, lihat "Konstanta Dollar-quoted String” di bawah Struktur Lek sikal dalam dokumentasi PostgreSQL.

Python Programme

Program Python yang dapat dieksekusi yang valid yang mengembalikan nilai. Pernyataan yang Anda berikan dengan fungsi harus sesuai dengan persyaratan lekukan seperti yang ditentukan dalam Panduan Gaya untuk Kode Python di situs web Python. Untuk informasi selengkapnya, lihat Dukungan bahasa Python untuk UDF.

SQL_klausa

Sebuah klausa SQL SELECT.

Klausa SELECT tidak dapat menyertakan salah satu dari jenis klausa berikut:

- FROM
- KE DALAM
- WHERE
- GROUP BY
- ORDER BY
- LIMIT

BAHASA {plpythonu | sql}

Untuk Python, tentukan`plpythonu`. Untuk SQL, tentukan`sql`. Anda harus memiliki izin untuk penggunaan bahasa untuk SQL atau plpythonu. Untuk informasi selengkapnya, lihat Keamanan dan izin UDF.

Catatan penggunaan

Fungsi bersarang

Anda dapat memanggil fungsi SQL yang ditentukan pengguna lain (UDF) dari dalam SQL UDF. Fungsi bersarang harus ada saat Anda menjalankan perintah CREATE FUNCTION. Amazon

Redshift tidak melacak dependensi untuk UDF, jadi jika Anda menghapus fungsi bersarang, Amazon Redshift tidak mengembalikan kesalahan. Namun, UDF akan gagal jika fungsi bersarang tidak ada. Misalnya, fungsi berikut memanggil `f_sql_greater` fungsi dalam klausa SELECT.

```
create function f_sql_commission (float, float )
  returns float
stable
as $$
  select f_sql_greater ($1, $2)
$$ language sql;
```

Keamanan dan hak istimewa UDF

Untuk membuat UDF, Anda harus memiliki izin untuk penggunaan bahasa untuk SQL atau plpythonu (Python). Secara default, USAGE ON LANGUAGE SQL diberikan kepada PUBLIK. Namun, Anda harus secara eksplisit memberikan USAGE ON LANGUAGE PLPYTHONU kepada pengguna atau grup tertentu.

Untuk mencabut penggunaan untuk SQL, pertama-tama cabut penggunaan dari PUBLIK. Kemudian berikan penggunaan pada SQL hanya untuk pengguna atau grup tertentu yang diizinkan untuk membuat SQL UDF. Contoh berikut mencabut penggunaan pada SQL dari PUBLIC kemudian memberikan penggunaan ke grup pengguna. `udf_devs`

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

Untuk menjalankan UDF, Anda harus memiliki izin eksekusi untuk setiap fungsi. Secara default, izin eksekusi untuk UDF baru diberikan kepada PUBLIC. Untuk membatasi penggunaan, cabut izin eksekusi dari PUBLIK untuk fungsi tersebut. Kemudian berikan hak istimewa kepada individu atau kelompok tertentu.

Contoh berikut mencabut izin eksekusi pada fungsi `f_py_greater` dari PUBLIC kemudian memberikan penggunaan ke grup pengguna. `udf_devs`

```
revoke execute on function f_py_greater(a float, b float) from PUBLIC;
grant execute on function f_py_greater(a float, b float) to group udf_devs;
```

Pengguna super memiliki semua hak istimewa secara default.

Untuk informasi selengkapnya, lihat HIBAH dan MENCABUT.

Contoh

Contoh UDF Python Scalar

Contoh berikut membuat Python UDF yang membandingkan dua bilangan bulat dan mengembalikan nilai yang lebih besar.

```
create function f_py_greater (a float, b float)
  returns float
stable
as $$
  if a > b:
    return a
  return b
$$ language plpythonu;
```

Contoh berikut menanyakan tabel PENJUALAN dan memanggil `f_py_greater` fungsi baru untuk mengembalikan COMMISSION atau 20 persen dari PRICEPAID, mana yang lebih besar.

```
select f_py_greater (commission, pricepaid*0.20) from sales;
```

Contoh UDF SQL Scalar

Contoh berikut menciptakan fungsi yang membandingkan dua angka dan mengembalikan nilai yang lebih besar.

```
create function f_sql_greater (float, float)
  returns float
stable
as $$
  select case when $1 > $2 then $1
    else $2
  end
$$ language sql;
```

Kueri berikut memanggil `f_sql_greater` fungsi baru untuk menanyakan tabel PENJUALAN dan mengembalikan COMMISSION atau 20 persen dari PRICEPAID, mana yang lebih besar.

```
select f_sql_greater (commission, pricepaid*0.20) from sales;
```

BUAT GRUP

Mendefinisikan grup pengguna baru. Hanya pengguna super yang dapat membuat grup.

Sintaksis

```
CREATE GROUP group_name
[ [ WITH ] [ USER username ] [, ...] ]
```

Parameter

group_name

Nama grup pengguna baru. Nama grup yang dimulai dengan dua garis bawah disediakan untuk penggunaan internal Amazon Redshift. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

DENGAN

Sintaks opsional untuk menunjukkan parameter tambahan untuk CREATE GROUP.

USER

Tambahkan satu atau lebih pengguna ke grup.

username

Nama pengguna yang akan ditambahkan ke grup.

Contoh

Contoh berikut membuat grup pengguna bernama ADMIN_GROUP dengan dua pengguna, ADMIN1 dan ADMIN2.

```
create group admin_group with user admin1, admin2;
```

BUAT PENYEDIA IDENTITAS

Mendefinisikan penyedia identitas baru. Hanya pengguna super yang dapat membuat penyedia identitas.

Sintaksis

```
CREATE IDENTITY PROVIDER identity_provider_name TYPE type_name
NAMESPACE namespace_name
[PARAMETERS parameter_string]
[APPLICATION_ARN arn]
[IAM_ROLE iam_role]
[AUTO_CREATE_ROLES
    [ TRUE [ { INCLUDE | EXCLUDE } GROUPS LIKE filter_pattern] |
      FALSE
    ]
  ];
```

Parameter

identity_provider_nama_

Nama penyedia identitas baru. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

nama tipe

Penyedia identitas untuk berinteraksi dengan. Azure dan AWSIDC saat ini satu-satunya penyedia identitas yang didukung.

nama_ruang nama

Namespace. Ini adalah pengenal singkatan unik untuk direktori penyedia identitas.

parameter_string

String yang berisi objek JSON yang diformat dengan benar yang berisi parameter dan nilai yang diperlukan untuk penyedia identitas.

arn

Nama sumber daya Amazon (ARN) untuk aplikasi yang dikelola Pusat Identitas IAM. Parameter ini hanya berlaku jika tipe penyedia identitas adalah AWSIDC.

iam_role

Peran IAM yang memberikan izin untuk membuat koneksi ke Pusat Identitas IAM. Parameter ini hanya berlaku jika tipe penyedia identitas adalah AWSIDC.

auto_create_role

Mengaktifkan atau menonaktifkan fitur peran buat otomatis. Jika nilainya TRUE, Amazon Redshift mengaktifkan fitur pembuatan peran otomatis. Jika nilainya SALAH, Amazon Redshift menonaktifkan fitur peran buat otomatis. Jika nilai untuk parameter ini tidak ditentukan, Amazon Redshift menentukan nilai menggunakan logika berikut:

- Jika `AUTO_CREATE_ROLES` disediakan tetapi nilainya tidak ditentukan, nilainya disetel ke TRUE.
- Jika `AUTO_CREATE_ROLES` tidak disediakan dan penyedia identitas adalah AWSIDC, nilainya disetel ke FALSE.
- Jika `AUTO_CREATE_ROLES` tidak disediakan dan penyedia identitasnya adalah Azure, nilainya disetel ke TRUE.

Untuk memasukkan grup, tentukan`INCLUDE`. Defaultnya kosong, yang berarti sertakan semua grup `AUTO_CREATE_ROLES` jika aktif.

Untuk mengecualikan grup, tentukan`EXCLUDE`. Defaultnya kosong, yang berarti jangan mengecualikan grup apa pun jika `AUTO_CREATE_ROLES` aktif.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama grup. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan setiap urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.

Jika filter_pattern tidak mengandung metacharacters, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator sama.

filter_pattern mendukung karakter berikut:

- Karakter alfabet huruf besar dan huruf kecil (dan a-z) A-Z
- Angka (0-9)

- Karakter khusus berikut:

```
_ % ^ * + ? { } , $
```

Contoh

Contoh berikut membuat penyedia identitas bernama oauth_standard, dengan TYPE azure, untuk menjalin komunikasi dengan Microsoft Azure Active Directory (AD).

```
CREATE IDENTITY PROVIDER oauth_standard TYPE azure
NAMESPACE 'aad'
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
"client_id":"87f4aa26-78b7-410e-bf29-57b39929ef9a",
"client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]
}'
```

Anda dapat menghubungkan aplikasi yang dikelola IAM Identity Center dengan cluster yang telah disediakan atau grup kerja Amazon Redshift Serverless yang sudah ada. Ini memberi Anda kemampuan untuk mengelola akses ke database Redshift melalui Pusat Identitas IAM. Untuk melakukannya, jalankan perintah SQL seperti contoh berikut. Anda harus menjadi administrator database.

```
CREATE IDENTITY PROVIDER "redshift-idc-app" TYPE AWSIDC
NAMESPACE 'awsidc'
APPLICATION_ARN 'arn:aws:sso::123456789012:application/ssoins-12345f67fe123d4/apl-
a0b0a12dc123b1a4'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyRedshiftRole';
```

Aplikasi ARN dalam hal ini mengidentifikasi aplikasi terkelola untuk terhubung. Anda dapat menemukannya dengan berlari`SELECT * FROM SVV_IDENTITY_PROVIDERS;`.

Untuk informasi selengkapnya tentang menggunakan CREATE IDENTITY PROVIDER, termasuk contoh tambahan, lihat Federasi penyedia identitas asli (IdP) untuk Amazon Red shift. Untuk informasi selengkapnya tentang menyiapkan koneksi ke Pusat Identitas IAM dari Redshift, lihat Menghubun gkan Redshift dengan Pusat Identitas IAM untuk memberi pengguna pengalaman masuk tunggal.

BUAT PERPUSTAKAAN

Menginstal pustaka Python, yang tersedia bagi pengguna untuk digabungkan saat membuat fungsi yang ditentukan pengguna (UDF) dengan perintahCREATE FUNCTION. Ukuran total pustaka yang diinstal pengguna tidak boleh melebihi 100 MB.

CREATE LIBRARY tidak dapat dijalankan di dalam blok transaksi (BEGIN... END). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Amazon Redshift mendukung Python versi 2.7. Untuk informasi lebih lanjut, lihat www.python.org.

Untuk informasi selengkapnya, lihat Contoh: Mengimpor modul pustaka Python khusus.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE LIBRARY:

- Superuser
- Pengguna dengan hak istimewa CREATE LIBRARY atau dengan hak istimewa bahasa yang ditentukan

Sintaksis

```
CREATE [ OR REPLACE ] LIBRARY library_name LANGUAGE plpythonu
FROM
{ 'https://file_url'
| 's3://bucketname/file_name'
authorization
  [ REGION [AS] 'aws_region']
  IAM_ROLE { default | ‘arn:aws:iam::<Akun AWS-id>:role/<role-name>’ }
}
```

Parameter

ATAU GANTI

Menentukan bahwa jika pustaka dengan nama yang sama seperti ini sudah ada, pustaka yang ada diganti. REPLACE segera melakukan commit. Jika UDF yang bergantung pada pustaka berjalan secara bersamaan, UDF mungkin gagal atau mengembalikan hasil yang tidak terduga,

bahkan jika UDF berjalan dalam transaksi. Anda harus menjadi pemilik atau superuser untuk mengganti perpustakaan.

nama_perpustakaan

Nama perpustakaan yang akan diinstal. Anda tidak dapat membuat pustaka yang berisi modul dengan nama yang sama dengan modul Pustaka Standar Python atau modul Python yang sudah diinstal sebelumnya Amazon Redshift. Jika pustaka yang diinstal pengguna yang ada menggunakan paket Python yang sama dengan pustaka yang akan diinstal, Anda harus menghapus pustaka yang ada sebelum menginstal pustaka baru. Untuk informasi selengkapnya, lihat Dukungan bahasa Python untuk UDF.

LANGUAGE plpythonu

Bahasa yang digunakan. Python (plpythonu) adalah satu-satunya bahasa yang didukung. Amazon Redshift mendukung Python versi 2.7. Untuk informasi lebih lanjut, lihat www.python.org.

FROM

Lokasi file perpustakaan. Anda dapat menentukan bucket Amazon S3 dan nama objek, atau Anda dapat menentukan URL untuk mengunduh file dari situs web publik. Perpustakaan harus dikemas dalam bentuk `.zip` file. Untuk informasi selengkapnya, lihat Membangun dan Menginstal Modul Python dalam dokumentasi Python.

https://file_url

URL untuk mengunduh file dari situs web publik. URL dapat berisi hingga tiga pengalihan. Berikut ini adalah contoh URL file.

```
'https://www.example.com/pylib.zip'
```

s3:// bucket_name/file_name

Jalur ke objek Amazon S3 tunggal yang berisi file pustaka. Berikut ini adalah contoh jalur objek Amazon S3.

```
's3://amzn-s3-demo-bucket/my-pylib.zip'
```

Jika Anda menentukan bucket Amazon S3, Anda juga harus memberikan kredenSIAL untuk AWS pengguna yang memiliki izin untuk mengunduh file.

> **Important**
>
> Jika bucket Amazon S3 tidak berada di Wil AWS ayah yang sama dengan cluster Amazon Redshift Anda, Anda harus menggunakan opsi REGION untuk menentukan AWS Wilayah tempat data berada. Nilai untuk aws_region harus cocok dengan AWS Region yang tercantum dalam tabel dalam deskripsi REGION parameter untuk perintah COPY.

otorisasi

Klausa yang menunjukkan metode yang digunakan cluster Anda untuk otentikasi dan otorisasi untuk mengakses bucket Amazon S3 yang berisi file pustaka. Cluster Anda harus memiliki izin untuk mengakses Amazon S3 dengan tindakan LIST dan GET.

Sintaks untuk otorisasi sama dengan otorisasi perintah COPY. Untuk informasi selengkapnya, lihat Parameter otorisasi.

```
IAM_ROLE { default | 'arn:aws:iam::<Akun AWS-id>:role/<role-name>'
```

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREATE LIBRARY dijalankan.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Jika Anda menentukan IAM_ROLE, Anda tidak dapat menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY, SESSION_TOKEN, atau CREDENTIALS.

Secara opsional, jika bucket Amazon S3 menggunakan enkripsi sisi server, berikan kunci enkripsi dalam string kredensials-args. Jika Anda menggunakan kredentif keamanan sementara, berikan token sementara dalam string credentials- args.

Untuk informasi selengkapnya, lihat Kredensial keamanan sementara.

WILAYAH [AS] aws_region

Wil AWS ayah tempat bucket Amazon S3 berada. REGION diperlukan jika bucket Amazon S3 tidak berada di Wilayah yang sama AWS dengan cluster Amazon Redshift. Nilai untuk aws_region harus cocok dengan AWS Region yang tercantum dalam tabel dalam deskripsi REGION parameter untuk perintah COPY.

Secara default, CREATE LIBRARY mengasumsikan bahwa bucket Amazon S3 terletak di Wil AWS ayah yang sama dengan cluster Amazon Redshift.

Contoh

Dua contoh berikut menginstal modul Python urlparse, yang dikemas dalam file bernama. `urlparse3-1.0.3.zip`

Perintah berikut menginstal pustaka UDF yang diberi `f_urlparse` nama dari paket yang telah diunggah ke bucket Amazon S3 yang terletak di Wilayah Timur AS.

```
create library f_urlparse
language plpythonu
from 's3://amzn-s3-demo-bucket/urlparse3-1.0.3.zip'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
region as 'us-east-1';
```

Contoh berikut menginstal pustaka bernama `f_urlparse` dari file pustaka di situs web.

```
create library f_urlparse
language plpythonu
from 'https://example.com/packages/urlparse3-1.0.3.zip';
```

BUAT KEBIJAKAN MASKING

Membuat kebijakan penyembunyian data dinamis baru untuk mengaburkan data dari format tertentu. Untuk informasi selengkapnya tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat membuat kebijakan masking.

Sintaksis

```
CREATE MASKING POLICY
   { policy_name | database_name.policy_name } [IF NOT EXISTS]
   WITH (input_columns)
   USING (masking_expression);
```

Parameter

nama_kebijakan

Nama kebijakan masking. Kebijakan masking tidak dapat memiliki nama yang sama dengan kebijakan masking lain yang sudah ada di database.

database_name

Nama database tempat kebijakan akan dibuat. Kebijakan dapat dibuat di database yang terhubung atau di Katalog Izin Federasi Amazon Redshift.

masukan_kolom

Tupel nama kolom dalam format (tipe col1, tipe col2...).

Nama kolom digunakan sebagai input untuk ekspresi masking. Nama kolom tidak harus cocok dengan nama kolom yang ditutupi, tetapi tipe data input dan output harus cocok.

ekspresi penyamaran

Ekspresi SQL digunakan untuk mengubah kolom target. Ini dapat ditulis menggunakan fungsi manipulasi data seperti fungsi manipulasi String, atau dalam hubungannya dengan fungsi yang ditentukan pengguna yang ditulis dalam SQL, Python, atau dengan AWS Lambda. Anda dapat menyertakan tupel ekspresi kolom untuk kebijakan masking yang memiliki beberapa output. Jika Anda menggunakan konstanta sebagai ekspresi masking, Anda harus secara eksplisit menempatkannya ke tipe yang cocok dengan tipe input.

Anda harus memiliki izin PENGGUNAAN pada fungsi yang ditentukan pengguna yang Anda gunakan dalam ekspresi masking.

Untuk penggunaan BUAT KEBIJAKAN MASKING di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

BUAT TAMPILAN YANG TERWUJUD

Membuat tampilan terwujud berdasarkan satu atau beberapa tabel Amazon Redshift. Anda juga dapat mendasarkan tampilan terwujud pada tabel eksternal yang dibuat menggunakan Spectrum atau kueri federasi. Untuk informasi tentang Spectrum, lihatSpektrum Pergeseran Merah Amazon. Untuk informasi tentang kueri federasi, lihatMengkueri data dengan kueri federasi di Amazon Redshift.

Sintaksis

```
CREATE MATERIALIZED VIEW mv_name
[ BACKUP { YES | NO } ]
[ table_attributes ]
[ AUTO REFRESH { YES | NO } ]
AS query
```

Parameter

CADANGAN

Klausa yang menentukan apakah tampilan yang terwujud harus disertakan dalam snapshot cluster otomatis dan manual.

Untuk tampilan terwujud yang tidak berisi data penting, tentukan BACKUP NO untuk menghemat waktu pemrosesan saat membuat snapshot dan memulihkan dari snapshot serta mengurangi ruang penyimpanan di Amazon Simple Storage Service. Pengaturan BACKUP NO tidak berpengaruh pada replikasi otomatis data ke node lain di dalam cluster, sehingga tampilan yang terwujud dengan BACKUP NO yang ditentukan dipulihkan jika terjadi kegagalan node. Defaultnya adalah BACKUP YES.

tabel_atribut

Klausa yang menentukan bagaimana data dalam tampilan terwujud didistribusikan, termasuk yang berikut:

- Gaya distribusi untuk tampilan yang terwujud, dalam format`DISTSTYLE { EVEN | ALL | KEY }`. Jika Anda menghilangkan klausa ini, gaya distribusinya adalah`EVEN`. Untuk informasi selengkapnya, lihat Gaya distribusi.
- Kunci distribusi untuk tampilan terwujud, dalam format`DISTKEY ( distkey_identifier )`. Untuk informasi selengkapnya, lihat Menunjuk gaya distribusi.
- Kunci pengurutan untuk tampilan terwujud, dalam format`SORTKEY ( column_name [, ...] )`. Untuk informasi selengkapnya, lihat Urutkan kunci.

Ku eri AS

Per `SELECT` nyataan valid yang mendefinisikan tampilan terwujud dan isinya. Kumpulan hasil dari kueri mendefinisikan kolom dan baris dari tampilan yang terwujud. Untuk informasi tentang keterbatasan saat membuat tampilan yang terwujud, lihatBatasan.

Selanjutnya, konstruksi bahasa SQL tertentu yang digunakan dalam kueri menentukan apakah tampilan yang terwujud dapat diperbarui secara bertahap atau sepenuhnya. Untuk informasi tentang metode penyegaran, lihatMENYEGARKAN TAMPILAN YANG TERWUJUD. Untuk informasi tentang batasan untuk penyegaran tambahan, lihatBatasan untuk penyegaran tambahan.

Jika kueri berisi perintah SQL yang tidak mendukung penyegaran tambahan, Amazon Redshift menampilkan pesan yang menunjukkan bahwa tampilan yang terwujud akan menggunakan penyegaran penuh. Pesan mungkin atau mungkin tidak ditampilkan, tergantung pada aplikasi klien SQL. Periksa `state` kolom STV_MV_INFO untuk melihat jenis refresh yang digunakan oleh tampilan yang terwujud.

PENYEGARAN OTOMATIS

Klausa yang mendefinisikan apakah tampilan yang terwujud harus diperbarui secara otomatis dengan perubahan terbaru dari tabel dasarnya. Nilai default-nya adalah `NO`. Untuk informasi selengkapnya, lihat Menyegarkan tampilan yang terwujud.

Catatan penggunaan

Untuk membuat tampilan yang terwujud, Anda harus memiliki hak istimewa berikut:

- BUAT hak istimewa untuk skema.
- Table-level atau hak istimewa SELECT tingkat kolom pada tabel dasar untuk membuat tampilan yang terwujud. Jika Anda memiliki hak istimewa tingkat kolom pada kolom tertentu, Anda dapat membuat tampilan yang terwujud hanya pada kolom tersebut.

Anda dapat membuat tampilan terwujud dari cluster pembagian data jarak jauh dengan memberikan nama database eksternal di. `mv_name`

Penyegaran tambahan untuk tampilan yang terwujud dalam datashare

Amazon Redshift mendukung penyegaran otomatis dan inkremental untuk tampilan yang terwujud di datashare konsumen saat tabel dasar dibagikan. Penyegaran inkremental adalah operasi di mana Amazon Redshift mengidentifikasi perubahan pada tabel dasar atau tabel yang terjadi setelah penyegaran sebelumnya dan hanya memperbarui catatan yang sesuai dalam tampilan yang terwujud. Ini berjalan lebih cepat daripada penyegaran penuh dan meningkatkan kinerja beban kerja. Anda tidak perlu mengubah definisi tampilan terwujud untuk memanfaatkan penyegaran tambahan.

Ada beberapa batasan yang perlu diperhatikan untuk memanfaatkan penyegaran tambahan dengan tampilan yang terwujud:

- Tampilan terwujud harus mereferensikan hanya satu database, baik lokal atau jarak jauh.
- Penyegaran tambahan hanya tersedia pada tampilan baru yang terwujud. Oleh karena itu, Anda harus menghapus tampilan terwujud yang ada dan membuatnya kembali agar penyegaran tambahan terjadi.

Untuk informasi selengkapnya tentang membuat tampilan terwujud di datashare, lihat Bek erja dengan tampilan di berbagi data Amazon Redshift, yang berisi beberapa contoh kueri.

Pembaruan DDL ke tampilan terwujud atau tabel dasar

Saat menggunakan tampilan terwujud di Amazon Redshift, ikuti catatan penggunaan ini untuk pembaruan bahasa definisi data (DDL) ke tampilan terwujud atau tabel dasar.

- Anda dapat menambahkan kolom ke tabel dasar tanpa mempengaruhi tampilan terwujud yang mereferensikan tabel dasar.
- Beberapa operasi dapat meninggalkan tampilan terwujud dalam keadaan yang tidak dapat disegarkan sama sekali. Contohnya adalah operasi seperti mengganti nama atau menjatuhkan kolom, mengubah jenis kolom, dan mengubah nama skema. Pandangan yang terwujud seperti itu dapat ditanyakan tetapi tidak dapat disegarkan. Dalam hal ini, Anda harus melepaskan dan membuat ulang tampilan yang terwujud.
- Secara umum, Anda tidak dapat mengubah definisi tampilan yang terwujud (pernyataan SQL-nya).
- Anda tidak dapat mengganti nama tampilan yang terwujud.

Batasan

Anda tidak dapat menentukan tampilan terwujud yang mereferensikan atau menyertakan salah satu dari berikut ini:

- Tampilan standar, atau tabel sistem dan tampilan.
- Tabel sementara.
- User-defined fungsi.
- Klausul ORDER BY, LIMIT, atau OFFSET.

- Late-binding referensi ke tabel dasar. Dengan kata lain, setiap tabel dasar atau kolom terkait yang direferensikan dalam kueri SQL yang menentukan dari tampilan terwujud harus ada dan harus valid.
- Fungsi hanya node pemimpin: CURRENT_SCHEMA, CURRENT_SCHEMAS, HAS_DATABASE_PRIVILEGE, HAS_SCHEMA_PRIVILEGE, HAS_TABLE_PRIVILEGE.
- Anda tidak dapat menggunakan opsi AUTO REFRESH YES ketika definisi tampilan terwujud menyertakan fungsi yang dapat diubah atau skema eksternal. Anda juga tidak dapat menggunakannya saat Anda menentukan tampilan terwujud pada tampilan terwujud lainnya.
- Anda tidak perlu menjalankan secara manual MENGANALISA pada tampilan yang terwujud. Ini terjadi saat ini hanya dengan menggunakan AUTO ANALYSIS. Untuk informasi selengkapnya, lihat Menganalisis tabel.
- RLS-protected atau DDM-protected tabel.
- Pembuatan tampilan terwujud dari cluster pembagian data jarak jauh tidak mendukung referensi pada tampilan terwujud lainnya, tabel Spectrum, tabel yang ditentukan dalam cluster Redshift dan UDF yang berbeda. Ini didukung untuk pembuatan tampilan terwujud dari cluster lokal (produsen).

Contoh

Contoh berikut membuat tampilan terwujud dari tiga tabel dasar yang digabungkan dan digabungkan. Setiap baris mewakili kategori dengan jumlah tiket yang terjual. Saat Anda menanyakan tampilan terwujud tickets_mv, Anda langsung mengakses data yang telah dihitung sebelumnya dalam tampilan terwujud tickets_mv.

```
CREATE MATERIALIZED VIEW tickets_mv AS
    select   catgroup,
    sum(qtysold) as sold
    from     category c, event e, sales s
    where    c.catid = e.catid
    and      e.eventid = s.eventid
    group by catgroup;
```

Contoh berikut membuat tampilan terwujud mirip dengan contoh sebelumnya dan menggunakan fungsi agregat MAX ().

```
CREATE MATERIALIZED VIEW tickets_mv_max AS
    select   catgroup,
    max(qtysold) as sold
```

```
    from     category c, event e, sales s
    where    c.catid = e.catid
    and      e.eventid = s.eventid
    group by catgroup;

SELECT name, state FROM STV_MV_INFO;
```

Contoh berikut menggunakan klausa UNION ALL untuk bergabung dengan tabel Amazon Redshift dan `public_sales` tabel Redshift Spectrum `spectrum.sales` untuk membuat tampilan material. `mv_sales_vw` Untuk informasi tentang perintah CREATE EXTERNAL TABLE untuk Amazon Redshift Spectrum, lihatCREATE EXTERNAL TABLE. Tabel eksternal Redshift Spectrum mereferensikan data di Amazon S3.

```
CREATE MATERIALIZED VIEW mv_sales_vw as
select salesid, qtysold, pricepaid, commission, saletime from public.sales
union all
select salesid, qtysold, pricepaid, commission, saletime from spectrum.sales
```

Contoh berikut membuat tampilan terwujud `mv_fq` berdasarkan tabel eksternal kueri federasi. Untuk informasi tentang kueri federasi, lihatBUAT SKEMA EKSTERNAL.

```
CREATE MATERIALIZED VIEW mv_fq as select firstname, lastname from apg.mv_fq_example;

select firstname, lastname from mv_fq;
 firstname | lastname
-----------+----------
 John      | Day
 Jane      | Doe
(2 rows)
```

Contoh berikut menunjukkan definisi tampilan yang terwujud.

```
SELECT pg_catalog.pg_get_viewdef('mv_sales_vw'::regclass::oid, true);

pg_get_viewdef
-------------------------------------------------
create materialized view mv_sales_vw as select a from t;
```

Contoh berikut menunjukkan cara mengatur AUTO REFRESH dalam definisi tampilan terwujud dan juga menentukan DISTSTYLE. Pertama, buat tabel dasar sederhana.

```
CREATE TABLE baseball_table (ball int, bat int);
```

Kemudian, buat tampilan yang terwujud.

```
CREATE MATERIALIZED VIEW mv_baseball DISTSTYLE ALL AUTO REFRESH YES AS SELECT ball AS
 baseball FROM baseball_table;
```

Sekarang Anda dapat menanyakan tampilan yang terwujud mv_baseball. Untuk memeriksa apakah AUTO REFRESH diaktifkan untuk tampilan yang terwujud, lihatSTV_MV_INFO.

Contoh berikut membuat tampilan terwujud yang mereferensikan tabel sumber di database lain. Ini mengasumsikan bahwa database yang berisi tabel sumber, Database_a, berada di cluster atau kelompok kerja yang sama dengan tampilan terwujud Anda, yang Anda buat di Database_b. (Anda dapat mengganti database Anda sendiri untuk sampel.) Pertama, buat tabel di Database_A bernama cities, dengan kolom cityname. Jadikan tipe data kolom sebagai VARCHAR. Setelah Anda membuat tabel sumber, jalankan perintah berikut di Database_b untuk membuat tampilan terwujud yang sumbernya adalah tabel kota Anda. Pastikan untuk menentukan database dan skema tabel sumber dalam klausa FROM:

```
CREATE MATERIALIZED VIEW cities_mv AS
SELECT  cityname
FROM    database_A.public.cities;
```

Kueri tampilan terwujud yang Anda buat. Kueri mengambil catatan yang sumber aslinya adalah tabel kota di Database_a:

```
select * from cities_mv;
```

Saat Anda menjalankan pernyataan SELECT, cities_mv mengembalikan catatan. Catatan disegarkan dari tabel sumber hanya ketika pernyataan REFRESH dijalankan. Juga, perhatikan bahwa Anda tidak dapat memperbarui catatan secara langsung di tampilan yang terwujud. Untuk informasi tentang menyegarkan data dalam tampilan terwujud, lihatMENYEGARKAN TAMPILAN YANG TERWUJUD.

Untuk detail tentang ikhtisar tampilan terwujud dan perintah SQL yang digunakan untuk menyegarkan dan menghapus tampilan yang terwujud, lihat topik berikut:

- Tampilan yang terwujud di Amazon Redshift
- MENYEGARKAN TAMPILAN YANG TERWUJUD

- JATUHKAN TAMPILAN TERWUJUD

BUAT MODEL

Topik

- Prasyarat
- Hak istimewa yang diperlukan
- Pengendalian biaya
- LENGKAP MEMBUAT MODEL
- Parameter
- Catatan penggunaan
- Kasus penggunaan

Prasyarat

Sebelum Anda menggunakan pernyataan CREATE MODEL, lengkapi prasyarat diPengaturan cluster untuk menggunakan Amazon Redshift ML. Berikut ini adalah ringkasan prasyarat tingkat tinggi.

- Buat cluster Amazon Redshift dengan AWS Management Console atau AWS Command Line Interface (AWS CLI).
- Lampirkan kebijakan AWS Identitas dan Manajemen Akses (IAM) saat membuat cluster.
- Untuk memungkinkan Amazon Redshift dan SageMaker AI mengambil peran untuk berinteraksi dengan layanan lain, tambahkan kebijakan kepercayaan yang sesuai ke peran IAM.

Untuk detail tentang peran IAM, kebijakan kepercayaan, dan prasyarat lainnya, lihatPengaturan cluster untuk menggunakan Amazon Redshift ML.

Berikut ini, Anda dapat menemukan berbagai kasus penggunaan untuk pernyataan CREATE MODEL.

- CIPTAKAN MODEL Sederhana
- BUAT MODEL dengan panduan pengguna
- BUAT model XGBoost dengan AUTO OFF
- Bawa model Anda sendiri (BYOM) - inferensi lokal

- Bawa model Anda sendiri (BYOM) - inferensi jarak jauh
- BUAT MODEL dengan K-MEANS
- LENGKAP MEMBUAT MODEL

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE MODEL:

- Superuser
- Pengguna dengan hak istimewa CREATE MODEL
- Peran dengan hak istimewa GRANT CREATE MODEL

Pengendalian biaya

Amazon Redshift ML menggunakan sumber daya cluster yang ada untuk membuat model prediksi, jadi Anda tidak perlu membayar biaya tambahan. Namun, Anda mungkin memiliki biaya tambahan jika Anda perlu mengubah ukuran cluster Anda atau ingin melatih model Anda. Amazon Redshift ML menggunakan Amazon SageMaker AI untuk melatih model, yang memang memiliki biaya terkait tambahan. Ada cara untuk mengontrol biaya tambahan, seperti membatasi jumlah waktu maksimum pelatihan yang dapat diambil atau dengan membatasi jumlah contoh pelatihan yang digunakan untuk melatih model Anda. Untuk informasi selengkapnya, lihat Biaya untuk menggunakan Amazon Redshift ML.

LENGKAP MEMBUAT MODEL

Berikut ini merangkum opsi dasar sintaks CREATE MODEL lengkap.

Sintaks CREATE MODEL penuh

Berikut ini adalah sintaks lengkap dari pernyataan CREATE MODEL.

> **Important**
>
> Saat membuat model menggunakan pernyataan CREATE MODEL, ikuti urutan kata kunci dalam sintaks berikut.

```
CREATE MODEL model_name
```

```
FROM { table_name | ( select_statement )  | 'job_name' }
[ TARGET column_name ]
FUNCTION function_name [ ( data_type [, ...] ) ]
[ RETURNS data_type ]
  -- supported only for BYOM
[ SAGEMAKER 'endpoint_name'[:'model_name']]
  -- supported only for BYOM remote inference
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
[ AUTO ON / OFF ]
  -- default is AUTO ON
[ MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER | KMEANS | FORECAST } ]
  -- not required for non AUTO OFF case, default is the list of all supported types
  -- required for AUTO OFF
[ PROBLEM_TYPE ( REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION ) ]
  -- not supported when AUTO OFF
[ OBJECTIVE ( 'MSE' | 'Accuracy' | 'F1' | 'F1_Macro' | 'AUC' |
             'reg:squarederror' | 'reg:squaredlogerror'| 'reg:logistic'|
             'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' |
 'binary:hinge',
             'multi:softmax' | 'RMSE' | 'WAPE' | 'MAPE' | 'MASE' |
 'AverageWeightedQuantileLoss' ) ]
  -- for AUTO ON: first 5 are valid
  -- for AUTO OFF: 6-13 are valid
  -- for FORECAST: 14-18 are valid
[ PREPROCESSORS 'string' ]
  -- required for AUTO OFF, when it has to be 'none'
  -- optional for AUTO ON
[ HYPERPARAMETERS { DEFAULT | DEFAULT EXCEPT ( Key 'value' (,...) ) } ]
  -- support XGBoost hyperparameters, except OBJECTIVE
  -- required and only allowed for AUTO OFF
  -- default NUM_ROUND is 100
  -- NUM_CLASS is required if objective is multi:softmax (only possible for AUTO OFF)
 [ SETTINGS (
   S3_BUCKET 'amzn-s3-demo-bucket',  |
    -- required
  TAGS 'string', |
    -- optional
  KMS_KEY_ID 'kms_string', |
    -- optional
  S3_GARBAGE_COLLECT on / off, |
    -- optional, defualt is on.
  MAX_CELLS integer, |
    -- optional, default is 1,000,000
  MAX_RUNTIME integer (, ...) |
```

```
    -- optional, default is 5400 (1.5 hours)
  HORIZON integer, |
    -- required if creating a forecast model
  FREQUENCY integer, |
    -- required if creating a forecast model
  PERCENTILES string, |
    -- optional if creating a forecast model
  MAX_BATCH_ROWS integer -- optional for BYOM remote inference
    ) ]
```

Parameter

nama_model

Nama modul. Nama model dalam skema harus unik.

DARI {table_name | (select_query) | 'job_name'}

Table_name atau kueri yang menentukan data pelatihan. Mereka dapat berupa tabel yang ada di sistem, atau kueri Amazon SELECT yang Redshift-compatible dilampirkan dengan tanda kurung, yaitu (). Setidaknya harus ada dua kolom dalam hasil kueri.

Nama kolom TARGET

Nama kolom yang menjadi target prediksi. Kolom harus ada dalam klausa FROM.

Fungsi nam a_fungsi (data_type [,...])

Nama fungsi yang akan dibuat dan tipe data dari argumen input. Anda dapat memberikan nama skema skema di database Anda alih-alih nama fungsi.

RETURNS data_type

Jenis data yang akan dikembalikan dari fungsi model. Jenis SUPER data yang dikembalikan hanya berlaku untuk BYOM dengan inferensi jarak jauh.

SAGEMAKER 'nama titik akhir' [: 'nama_model']

Nama titik akhir Amazon SageMaker AI. Jika nama titik akhir menunjuk ke titik akhir multimodel, tambahkan nama model yang akan digunakan. Titik akhir harus di-host Wilayah AWS sama dengan cluster Amazon Redshift.

<account-id><role-name>IAM_ROLE {default | 'arn:aws:iam: :rol/ '}

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREATE MODEL dijalankan. Atau, Anda dapat menentukan ARN dari peran IAM untuk menggunakan peran itu.

[OTOMATIS HIDUP/MATI]

Mengaktifkan atau menonaktifkan penemuan otomatis CREATE MODEL dari pemilihan preprosesor, algoritma, dan hiper-parameter. Menentukan saat membuat model Prakiraan menunjukkan untuk menggunakan AutoPredictor, di mana Amazon Forecast menerapkan kombinasi algoritma yang optimal untuk setiap deret waktu dalam kumpulan data Anda.

TIPE_MODEL {XGBOOST | MLP | PEMBELAJAR_LINIER | KMEANS | PRAKIRAAN}

(Opsional) Menentukan jenis model. Anda dapat menentukan apakah Anda ingin melatih model tipe model tertentu, seperti XGBoost, multilayer perceptron (MLP), KMEANS, atau Linear Learner, yang semuanya merupakan algoritma yang didukung Amazon AI Autopilot. SageMaker Jika Anda tidak menentukan parameter, maka semua jenis model yang didukung dicari selama pelatihan untuk model terbaik. Anda juga dapat membuat model perkiraan di Redshift ML untuk membuat perkiraan deret waktu yang akurat.

TIPE_MASALAH (REGRESI | KLASIFIKASI_BINER | KLASASI_MULTIKLASIFIKASI)

(Opsional) Menentukan jenis masalah. Jika Anda mengetahui jenis masalahnya, Anda dapat membatasi Amazon Redshift untuk hanya mencari model terbaik dari jenis model tertentu. Jika Anda tidak menentukan parameter ini, jenis masalah akan ditemukan selama pelatihan, berdasarkan data Anda.

TUJUAN ('MSE' | 'Akurasi' | 'F1' | 'F1Macro' | 'AUC' | 'reg:squarederror' | 'reg:squaredlogerror' | 'reg:logistik' | 'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' | 'binary:logistik' | 'multi:softmax' | 'RMSE' | 'WAPE' | 'MAPE' | 'MASE' | '') AverageWeightedQuantileLoss

(Opsional) Menentukan nama metrik objektif yang digunakan untuk mengukur kualitas prediktif sistem pembelajaran mesin. Metrik ini dioptimalkan selama pelatihan untuk memberikan perkiraan terbaik untuk nilai parameter model dari data. Jika Anda tidak menentukan metrik secara eksplisit, perilaku defaultnya adalah secara otomatis menggunakan MSE: untuk regresi, F1: untuk klasifikasi biner, Akurasi: untuk klasifikasi multiclass. Untuk informasi selengkapnya tentang tujuan, lihat AutoMLJobObjective di parameter tugas Refer SageMaker ensi Amazon AI API dan Pem belajaran di dokumentasi XGBOOST. Nilai RMSE, WAPE, MAPE, MASE, dan hanya berlaku untuk model AverageWeightedQuantileLoss Prakiraan. Untuk informasi selengkapnya, lihat operasi API CreateAutoPredictor.

PREPROCESSOR 'string'

(Opsional) Menentukan kombinasi tertentu dari preprocessor ke kumpulan kolom tertentu. Format adalah daftar ColumnSets, dan transformasi yang sesuai untuk diterapkan pada setiap set kolom. Amazon Redshift menerapkan semua transformator dalam daftar transformer tertentu ke semua kolom yang sesuai ColumnSet. Misalnya, untuk menerapkan OneHotEncoder dengan Imputer ke kolom t1 dan t2, gunakan perintah sampel berikut.

```
CREATE MODEL customer_churn
FROM customer_data
TARGET 'Churn'
FUNCTION predict_churn
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
PROBLEM_TYPE BINARY_CLASSIFICATION
OBJECTIVE 'F1'
PREPROCESSORS '[
...
  {"ColumnSet": [
      "t1",
      "t2"
    ],
    "Transformers": [
      "OneHotEncoder",
      "Imputer"
    ]
  },
  {"ColumnSet": [
      "t3"
    ],
    "Transformers": [
      "OneHotEncoder"
    ]
  },
  {"ColumnSet": [
      "temp"
    ],
    "Transformers": [
      "Imputer",
      "NumericPassthrough"
    ]
  }
]'
SETTINGS (
```

```
  S3_BUCKET 'amzn-s3-demo-bucket'
)
```

HYPERPARAMETER {DEFAULT | DEFAULT KECUALI (kunci 'nilai' (,..))}

Menentukan apakah parameter XGBoost default digunakan atau diganti oleh nilai yang ditentukan pengguna. Nilai-nilai harus ditutup dengan tanda kutip tunggal. Berikut ini adalah contoh parameter untuk XGBoost dan defaultnya.

Nama parameter	Nilai parameter	Nilai default	Catatan
nom_class	Bilangan Bulat	Diperlu n untuk klasifik si Multicl s.	N/A
num_putar an	Bilangan Bulat	100	N/A
metode pohon	String	Otomat	N/A
kedalaman maksimum	Bilangan Bulat	6	[0, 10]
berat min_anak	Desimal	1	MinValue: 0, MaxValue: 120
subsampel	Desimal	1	MinValue: 0.5, MaxValue: 1
gama	Desimal	0	MinValue: 0, MaxValue: 5
alfa	Desimal	0	MinValue: 0, MaxValue: 1000
eTea	Desimal	0.3	MinValue: 0,1, MaxValue: 0,5

Nama parameter	Nilai parameter	Nilai default	Catatan
colsample_byleve	Desimal	1	MinValue: 0,1, MaxValue: 1
colsample_bynode	Desimal	1	MinValue: 0,1, MaxValue: 1
colsample_bytree	Desimal	1	MinValue: 0.5, MaxValue: 1
Lambda	Desimal	1	MinValue: 0, MaxValue: 1000
max_delta_step	Bilangan Bulat	0	[0, 10]

PENGATURAN (S3_BUCKET 'amzn-s3-demo-bucket', | TAGS 'string', | KMS_KEY_ID 'kms_string', | S3_GARBAGE_COLLECT hidup/mati, | MAX_ CELLS integer, | MAX_RUNTIME (,...), | bilangan bulat HORIZON, | frekuensi predikast_frekuensi, | array string persentil)

Klausa S3_BUCKET menentukan lokasi Amazon S3 yang digunakan untuk menyimpan hasil perantara.

(Opsional) Parameter TAGS adalah daftar pasangan kunci-nilai yang dipisahkan koma yang dapat Anda gunakan untuk menandai sumber daya yang dibuat di Amazon SageMaker AI; dan Amazon Forecast. Tag membantu Anda mengatur sumber daya dan mengalokasikan biaya. Nilai dalam pasangan adalah opsional, sehingga Anda dapat membuat tag dengan menggunakan format `key=value` atau hanya dengan membuat kunci. Untuk informasi selengkapnya tentang tag di Amazon Redshift, lihat Ringkasan pen andaan.

(Opsional) KMS_KEY_ID menentukan apakah Amazon Redshift menggunakan enkripsi sisi server dengan kunci untuk melindungi data saat diam. AWS KMS Data yang sedang transit dilindungi dengan Secure Sockets Layer (SSL).

(Opsional) S3_GARBAGE_COLLECT {ON | OFF} menentukan apakah Amazon Redshift melakukan pengumpulan sampah pada kumpulan data yang dihasilkan yang digunakan untuk melatih model dan model. Jika disetel ke OFF, kumpulan data yang dihasilkan digunakan untuk melatih model dan model tetap ada di Amazon S3 dan dapat digunakan untuk tujuan lain. Jika

disetel ke AKTIF, Amazon Redshift menghapus artefak di Amazon S3 setelah pelatihan selesai. Defaultnya ON.

(Opsional) MAX_CELLS menentukan jumlah sel dalam data pelatihan. Nilai ini adalah produk dari jumlah catatan (dalam kueri atau tabel pelatihan) dikalikan jumlah kolom. Default adalah 1.000.000.

(Opsional) MAX_RUNTIME menentukan jumlah waktu maksimum untuk berlatih. Pekerjaan pelatihan sering selesai lebih cepat tergantung pada ukuran dataset. Ini menentukan jumlah waktu maksimum yang harus dilakukan pelatihan. Defaultnya adalah 5.400 (90 menit).

HORIZON menentukan jumlah prediksi maksimum yang dapat dikembalikan model perkiraan. Setelah model dilatih, Anda tidak dapat mengubah bilangan bulat ini. Parameter ini diperlukan jika melatih model perkiraan.

FREQUENCY menentukan seberapa terperinci dalam satuan waktu yang Anda inginkan dari perkiraan. Opsi yang tersedia adalah`Y | M | W | D | H | 30min | 15min | 10min | 5min | 1min`. Parameter ini diperlukan jika melatih model perkiraan.

(Opsional) PERCENTILES adalah string yang dipisahkan koma yang menentukan jenis perkiraan yang digunakan untuk melatih prediktor. Jenis prakiraan dapat berupa kuantil dari 0,01 hingga 0,99, dengan penambahan 0,01 atau lebih tinggi. Anda juga dapat menentukan perkiraan rata-rata dengan mean. Anda dapat menentukan maksimal lima jenis perkiraan.

MAX_BATCH_ROWS bilangan bulat

(Opsional) Jumlah maksimum baris yang dikirim Amazon Redshift dalam permintaan batch tunggal untuk satu pemanggilan SageMaker AI. Ini didukung hanya untuk BYOM dengan inferensi jarak jauh. Nilai minimum parameter ini adalah 1. Nilai maksimumnya adalah`INT_MAX`, atau 2.147.483.647. Parameter ini diperlukan hanya jika tipe data input dan yang dikembalikan adalah SUPER. Nilai defaultnya adalah`INT_MAX`, atau 2.147.483.647.

Catatan penggunaan

Saat menggunakan CREATE MODEL, pertimbangkan hal berikut:

- Pernyataan CREATE MODEL beroperasi dalam mode asinkron dan kembali setelah ekspor data pelatihan ke Amazon S3. Langkah-langkah pelatihan yang tersisa di Amazon SageMaker AI terjadi di latar belakang. Saat pelatihan sedang berlangsung, fungsi inferensi yang sesuai terlihat tetapi

tidak dapat dijalankan. Anda dapat STV_ML_MODEL_INFO menanyakan untuk melihat keadaan pelatihan.

- Pelatihan dapat berjalan hingga 90 menit di latar belakang, secara default dalam model Otomatis dan dapat diperpanjang. Untuk membatalkan pelatihan, cukup jalankan MODEL JATUHKAN perintah.
- Cluster Amazon Redshift yang Anda gunakan untuk membuat model dan bucket Amazon S3 yang digunakan untuk mementaskan data pelatihan dan artefak model harus berada di Wilayah yang sama AWS .
- Selama pelatihan model, Amazon Redshift dan SageMaker AI menyimpan artefak perantara di bucket Amazon S3 yang Anda berikan. Secara default, Amazon Redshift melakukan pengumpulan sampah di akhir operasi CREATE MODEL. Amazon Redshift menghapus objek tersebut dari Amazon S3. Untuk mempertahankan artefak tersebut di Amazon S3, setel opsi S3_GARBAGE COLLECT OFF.
- Anda harus menggunakan setidaknya 500 baris dalam data pelatihan yang disediakan dalam klausa FROM.
- Anda hanya dapat menentukan hingga 256 kolom fitur (input) dalam klausa FROM {table_name | (select_query)} saat menggunakan pernyataan CREATE MODEL.
- Untuk AUTO ON, tipe kolom yang dapat Anda gunakan sebagai set pelatihan adalah SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE, BOOLEAN, CHAR, VARCHAR, DATE, TIME, TIMETZ, TIMESTAMP, dan TIMESTAMPTZ. Untuk AUTO OFF, tipe kolom yang dapat Anda gunakan sebagai set pelatihan adalah SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE, dan BOOLEAN.
- Anda tidak dapat menggunakan DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, TIMESTAMPTZ, GEOMETRY, GEOGRAPHY, HLLSKETCH, SUPER, atau VARBYTE sebagai tipe kolom target.
- Untuk meningkatkan akurasi model, lakukan salah satu hal berikut:
  - Tambahkan sebanyak mungkin kolom yang relevan dalam perintah CREATE MODEL saat Anda menentukan data pelatihan dalam klausa FROM.
  - Gunakan nilai yang lebih besar untuk MAX_RUNTIME dan MAX_CELLS. Nilai yang lebih besar untuk parameter ini meningkatkan biaya pelatihan model.
- Eksekusi pernyataan CREATE MODEL dikembalikan segera setelah data pelatihan dihitung dan diekspor ke bucket Amazon S3. Setelah itu, Anda dapat memeriksa status pelatihan menggunakan perintah SHOW MODEL. Ketika model yang dilatih di latar belakang gagal, Anda dapat memeriksa kesalahan menggunakan SHOW MODEL. Anda tidak dapat mencoba lagi model yang gagal.

Gunakan DROP MODEL untuk menghapus model yang gagal dan membuat ulang model baru. Untuk informasi selengkapnya tentang SHOW MODEL, lihatTUNJUKKAN MODEL.

- BYOM lokal mendukung jenis model yang sama dengan yang didukung Amazon Redshift ML untuk kasus non-BYOM. Amazon Redshift mendukung XGBoost biasa (menggunakan XGBoost versi 1.0 atau lebih baru), model KMEANS tanpa preprosesor, dan model Pembelajar yang dilatih oleh Amazon AI Aut XGBOOST/MLP/Linear opilot. SageMaker Ini mendukung yang terakhir dengan preprosesor yang telah ditentukan Autopilot yang juga didukung oleh Amazon SageMaker AI Neo.
- Jika cluster Amazon Redshift Anda telah mengaktifkan perutean yang disempurnakan untuk cloud pribadi virtual (VPC) Anda, pastikan untuk membuat titik akhir Amazon S3 VPC dan titik akhir VPC SageMaker AI untuk VPC tempat cluster Anda berada. Melakukan hal ini memungkinkan lalu lintas berjalan melalui VPC Anda antara layanan ini selama CREATE MODEL. Untuk informasi selengkapnya, lihat SageMaker AI Clarify Job Amazon VPC Subnet dan Grup Keamanan.

Kasus penggunaan

Kasus penggunaan berikut menunjukkan cara menggunakan CREATE MODEL agar sesuai dengan kebutuhan Anda.

CIPTAKAN MODEL Sederhana

Berikut ini merangkum opsi dasar sintaks CREATE MODEL.

Sintaks CREATE MODEL sederhana

```
CREATE MODEL model_name
FROM { table_name | ( select_query ) }
TARGET column_name
FUNCTION prediction_function_name
IAM_ROLE { default }
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  [ MAX_CELLS integer ]
)
```

Parameter CREATE MODEL sederhana

nama_model

Nama modul. Nama model dalam skema harus unik.

DARI {table_name | (sel ect_query)}

Table_name atau kueri yang menentukan data pelatihan. Mereka dapat berupa tabel yang ada di sistem, atau kueri Amazon SELECT yang Redshift-compatible dilampirkan dengan tanda kurung, yaitu (). Setidaknya harus ada dua kolom dalam hasil kueri.

Nama kolom TARGET

Nama kolom yang menjadi target prediksi. Kolom harus ada dalam klausa FROM.

P rediksi_fungsi_nama_fungsi FUNGSI

Nilai yang menentukan nama fungsi pembelajaran mesin Amazon Redshift yang akan dihasilkan oleh CREATE MODEL dan digunakan untuk membuat prediksi menggunakan model ini. Fungsi ini dibuat dalam skema yang sama dengan objek model dan dapat kelebihan beban.

Pembelajaran mesin Amazon Redshift mendukung model, seperti model Xtreme Gradient Boosted tree (XGBoost) untuk regresi dan klasifikasi.

<account-id><role-name>IAM_ROLE {default | 'arn:aws:iam: :rol/ '}

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREAT MODEL berjalan. Atau, Anda dapat menentukan ARN dari peran IAM untuk menggunakan peran itu.

S3_BUCKET 'amzn-s3-demo-bucket'

Nama bucket Amazon S3 yang Anda buat sebelumnya digunakan untuk berbagi data pelatihan dan artefak antara Amazon Redshift dan SageMaker AI. Amazon Redshift membuat subfolder di bucket ini sebelum membongkar data pelatihan. Saat pelatihan selesai, Amazon Redshift menghapus subfolder yang dibuat dan isinya.

MAX_CELLS bilangan bulat

Jumlah maksimum sel untuk diekspor dari klausa FROM. Default adalah 1.000.000.

Jumlah sel adalah produk dari jumlah baris dalam data pelatihan (dihasilkan oleh tabel klausa FROM atau kueri) dikalikan jumlah kolom. Jika jumlah sel dalam data pelatihan lebih dari yang ditentukan oleh parameter max_cells, CREATE MODEL menurunkan sampel data pelatihan klausa FROM untuk mengurangi ukuran set pelatihan di bawah MAX_CELLS. Mengizinkan kumpulan data pelatihan yang lebih besar dapat menghasilkan akurasi yang lebih tinggi tetapi juga dapat berarti model membutuhkan waktu lebih lama untuk dilatih dan harganya lebih mahal.

Untuk informasi tentang biaya penggunaan Amazon Redshift, lihatBiaya untuk menggunakan Amazon Redshift ML.

Untuk informasi selengkapnya tentang biaya yang terkait dengan berbagai nomor ponsel dan detail uji coba gratis, lihat harga https://aws.amazon.com/redshift/pricing Amazon Redshift.

BUAT MODEL dengan panduan pengguna

Berikut ini, Anda dapat menemukan deskripsi opsi untuk CREATE MODEL selain opsi yang dijelaskan diCIPTAKAN MODEL Sederhana.

Secara default, CREATE MODEL mencari kombinasi pra-pemrosesan dan model terbaik untuk dataset spesifik Anda. Anda mungkin ingin kontrol tambahan atau memperkenalkan pengetahuan domain tambahan (seperti jenis masalah atau tujuan) atas model Anda. Dalam skenario churn pelanggan, jika hasil “pelanggan tidak aktif” jarang terjadi, maka tujuan F1 sering lebih disukai daripada tujuan akurasi. Karena model akurasi tinggi dapat memprediksi “pelanggan aktif” sepanjang waktu, ini menghasilkan akurasi tinggi tetapi nilai bisnis kecil. Untuk informasi tentang tujuan F1, lihat AutoMLJobObjective di Referensi API Amazon SageMaker AI.

Kemudian CREATE MODEL mengikuti saran Anda pada aspek yang ditentukan, seperti tujuan. Pada saat yang sama, CREATE MODEL secara otomatis menemukan preprosesor terbaik dan hyperparameter terbaik.

BUAT MODEL dengan sintaks panduan pengguna

CREATE MODEL menawarkan lebih banyak fleksibilitas pada aspek yang dapat Anda tentukan dan aspek yang ditemukan Amazon Redshift secara otomatis.

```
CREATE MODEL model_name
FROM { table_name | ( select_statement ) }
TARGET column_name
FUNCTION function_name
IAM_ROLE { default }
[ MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER} ]
[ PROBLEM_TYPE ( REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION ) ]
[ OBJECTIVE ( 'MSE' | 'Accuracy' | 'F1' | 'F1Macro' | 'AUC') ]
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket', |
  S3_GARBAGE_COLLECT { ON | OFF }, |
  KMS_KEY_ID 'kms_key_id', |
  MAX_CELLS integer, |
  MAX_RUNTIME integer (, ...)
)
```

BUAT MODEL dengan parameter panduan pengguna

TIPE_MODEL {XGBOOST | MLP | PEMBELAJAR_LINIER}

(Opsional) Menentukan jenis model. Anda dapat menentukan apakah Anda ingin melatih model tipe model tertentu, seperti XGBoost, multilayer perceptron (MLP), atau Linear Learner, yang semuanya merupakan algoritma yang didukung Amazon AI Autopilot. SageMaker Jika Anda tidak menentukan parameter, maka semua jenis model yang didukung dicari selama pelatihan untuk model terbaik.

TIPE_MASALAH (REGRESI | KLASIFIKASI_BINER | KLASASI_MULTIKLASIFIKASI)

(Opsional) Menentukan jenis masalah. Jika Anda mengetahui jenis masalahnya, Anda dapat membatasi Amazon Redshift untuk hanya mencari model terbaik dari jenis model tertentu. Jika Anda tidak menentukan parameter ini, jenis masalah akan ditemukan selama pelatihan, berdasarkan data Anda.

TUJUAN ('MSE' | 'Akurasi' | 'F1' | 'F1Makro' | 'AUC')

(Opsional) Menentukan nama metrik objektif yang digunakan untuk mengukur kualitas prediktif sistem pembelajaran mesin. Metrik ini dioptimalkan selama pelatihan untuk memberikan perkiraan terbaik untuk nilai parameter model dari data. Jika Anda tidak menentukan metrik secara eksplisit, perilaku defaultnya adalah secara otomatis menggunakan MSE: untuk regresi, F1: untuk klasifikasi biner, Akurasi: untuk klasifikasi multiclass. Untuk informasi selengkapnya tentang tujuan, lihat AutoMLJobObjective di Referensi API Amazon SageMaker AI.

MAX_CELLS bilangan bulat

(Opsional) Menentukan jumlah sel dalam data pelatihan. Nilai ini adalah produk dari jumlah catatan (dalam kueri atau tabel pelatihan) dikalikan jumlah kolom. Default adalah 1.000.000.

MAX_RUNTIME bilangan bulat

(Opsional) Menentukan jumlah waktu maksimum untuk berlatih. Pekerjaan pelatihan sering selesai lebih cepat tergantung pada ukuran dataset. Ini menentukan jumlah waktu maksimum yang harus dilakukan pelatihan. Defaultnya adalah 5.400 (90 menit).

S3_GARBAGE_COLLECT {AKTIF | MATI}

(Opsional) Menentukan apakah Amazon Redshift melakukan pengumpulan sampah pada kumpulan data yang dihasilkan yang digunakan untuk melatih model dan model. Jika disetel ke OFF, kumpulan data yang dihasilkan digunakan untuk melatih model dan model tetap ada di Amazon S3 dan dapat digunakan untuk tujuan lain. Jika disetel ke AKTIF, Amazon Redshift menghapus artefak di Amazon S3 setelah pelatihan selesai. Defaultnya ON.

KMS_KEY_ID 'kms_key_id'

(Opsional) Menentukan apakah Amazon Redshift menggunakan enkripsi sisi server dengan AWS KMS kunci untuk melindungi data saat diam. Data yang sedang transit dilindungi dengan Secure Sockets Layer (SSL).

PREPROCESSOR 'string'

(Opsional) Menentukan kombinasi tertentu dari preprocessor ke kumpulan kolom tertentu. Format adalah daftar ColumnSets, dan transformasi yang sesuai untuk diterapkan pada setiap set kolom. Amazon Redshift menerapkan semua transformator dalam daftar transformer tertentu ke semua kolom yang sesuai ColumnSet. Misalnya, untuk menerapkan OneHotEncoder dengan Imputer ke kolom t1 dan t2, gunakan perintah sampel berikut.

```
CREATE MODEL customer_churn
FROM customer_data
TARGET 'Churn'
FUNCTION predict_churn
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
PROBLEM_TYPE BINARY_CLASSIFICATION
OBJECTIVE 'F1'
PREPROCESSORS '[
...
{"ColumnSet": [
    "t1",
    "t2"
  ],
  "Transformers": [
    "OneHotEncoder",
    "Imputer"
  ]
},
{"ColumnSet": [
    "t3"
  ],
  "Transformers": [
    "OneHotEncoder"
  ]
},
{"ColumnSet": [
    "temp"
  ],
  "Transformers": [
```

```
    "Imputer",
    "NumericPassthrough"
  ]
}
]'
SETTINGS (
S3_BUCKET 'amzn-s3-demo-bucket'
)
```

Amazon Redshift mendukung transformer berikut:

- OneHotEncoder — Biasanya digunakan untuk mengkodekan nilai diskrit ke dalam vektor biner dengan satu nilai bukan nol. Transformator ini cocok untuk banyak model pembelajaran mesin.
- OrdinalEncoder — Mengkodekan nilai-nilai diskrit menjadi satu bilangan bulat. Transformator ini cocok untuk model pembelajaran mesin tertentu, seperti MLP dan Linear Learner.
- NumericPassthrough — Meneruskan input apa adanya ke dalam model.
- Impputer — Mengisi nilai yang hilang dan bukan nilai angka (NaN).
- ImputerWithIndicator — Mengisi nilai yang hilang dan nilai NaN. Transformator ini juga menciptakan indikator apakah ada nilai yang hilang dan diisi.
- Normalizer - Menormalkan nilai, yang dapat meningkatkan kinerja banyak algoritma pembelajaran mesin.
- DateTimeVectorizer - Membuat penyematan vektor, mewakili kolom tipe data datetime yang dapat digunakan dalam model pembelajaran mesin.
- PCA — Memproyeksikan data ke ruang dimensi yang lebih rendah untuk mengurangi jumlah fitur sambil menyimpan informasi sebanyak mungkin.
- StandardScaler — Menstandardisasi fitur dengan menghapus rata-rata dan penskalaan ke varians unit.
- MinMax — Mengubah fitur dengan menskalakan setiap fitur ke rentang tertentu.

Amazon Redshift ML menyimpan transformator terlatih, dan secara otomatis menerapkannya sebagai bagian dari kueri prediksi. Anda tidak perlu menentukannya saat menghasilkan prediksi dari model Anda.

BUAT model XGBoost dengan AUTO OFF

AUTO OFF CREATE MODEL umumnya memiliki tujuan yang berbeda dari CREATE MODEL default.

Sebagai pengguna tingkat lanjut yang sudah mengetahui jenis model yang Anda inginkan dan hyperparameter untuk digunakan saat melatih model ini, Anda dapat menggunakan CREATE MODEL dengan AUTO OFF untuk mematikan penemuan otomatis CREATE MODEL dari preprocessor dan hyperparameter. Untuk melakukannya, Anda secara eksplisit menentukan jenis model. XGBoost saat ini merupakan satu-satunya jenis model yang didukung saat AUTO disetel ke OFF. Anda dapat menentukan hyperparameter. Amazon Redshift menggunakan nilai default untuk hyperparameter apa pun yang Anda tentukan.

BUAT model XGBoost dengan sintaks AUTO OFF

```
CREATE MODEL model_name
FROM { table_name | (select_statement ) }
TARGET column_name
FUNCTION function_name
IAM_ROLE { default }
AUTO OFF
MODEL_TYPE XGBOOST
OBJECTIVE { 'reg:squarederror' | 'reg:squaredlogerror' | 'reg:logistic' |
            'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' | 'binary:hinge'
 |
            'multi:softmax' | 'rank:pairwise' | 'rank:ndcg' }
HYPERPARAMETERS DEFAULT EXCEPT (
    NUM_ROUND '10',
    ETA '0.2',
    NUM_CLASS '10',
    (, ...)
)
PREPROCESSORS 'none'
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket', |
  S3_GARBAGE_COLLECT { ON | OFF }, |
  KMS_KEY_ID 'kms_key_id', |
  MAX_CELLS integer, |
  MAX_RUNTIME integer (, ...)
)
```

BUAT model XGBoost dengan parameter AUTO OFF

MATI OTOMATIS

Mematikan penemuan otomatis CREATE MODEL dari pemilihan preprosesor, algoritma, dan hiper-parameter.

MODEL_TYPE XGBOOST

Menentukan untuk menggunakan XGBOOST untuk melatih model.

TUJUAN str

Menentukan tujuan yang dikenali oleh algoritma. Amazon Redshift mendukung reg:squarederror, reg:squaredlogerror, reg:logistic, reg:pseudohubererror, reg:tweedie, binary:logistic, binary:engsel, multi:softmax. Untuk informasi selengkapnya tentang tujuan ini, lihat Param eter tugas belajar di dokumentasi XGBoost.

HYPERPARAMETER {DEFAULT | DEFAULT KECUALI (kunci 'nilai' (,..))}

Menentukan apakah parameter XGBoost default digunakan atau diganti oleh nilai yang ditentukan pengguna. Nilai-nilai harus ditutup dengan tanda kutip tunggal. Berikut ini adalah contoh parameter untuk XGBoost dan defaultnya.

Nama parameter	Nilai parameter	Nilai default	Catatan
nom_class	Bilangan Bulat	Diperlu n untuk klasifik si Multicl s.	N/A
num_putar an	Bilangan Bulat	100	N/A
metode pohon	String	Otoma	N/A
kedalaman maksimum	Bilangan Bulat	6	[0, 10]
berat min_anak	Desimal	1	MinValue: 0, MaxValue: 120
subsampel	Desimal	1	MinValue: 0,5, MaxValue: 1

Nama parameter	Nilai parameter	Nilai default	Catatan
gama	Desimal	0	MinValue: 0, MaxValue: 5
alfa	Desimal	0	MinValue: 0, MaxValue: 1000
eTea	Desimal	0.3	MinValue: 0,1, MaxValue: 0,5
colsample_byleve	Desimal	1	MinValue: 0,1, MaxValue: 1
colsample_bynode	Desimal	1	MinValue: 0,1, MaxValue: 1
colsample_bytree	Desimal	1	MinValue: 0,5, MaxValue: 1
Lambda	Desimal	1	MinValue: 0, MaxValue: 1000
max_delta_step	Bilangan Bulat	0	[0, 10]

Contoh berikut menyiapkan data untuk XGBoost.

```
DROP TABLE IF EXISTS abalone_xgb;

CREATE TABLE abalone_xgb (
length_val float,
diameter float,
height float,
whole_weight float,
shucked_weight float,
viscera_weight float,
shell_weight float,
rings int,
record_number int);

COPY abalone_xgb
FROM 's3://redshift-downloads/redshift-ml/abalone_xg/'
REGION 'us-east-1'
```

```
IAM_ROLE default
IGNOREHEADER 1 CSV;
```

Contoh berikut membuat model XGBoost dengan opsi lanjutan tertentu, seperti MODEL_TYPE, OBJECTIVE, dan PREPROCESSORS.

```
DROP MODEL abalone_xgboost_multi_predict_age;

CREATE MODEL abalone_xgboost_multi_predict_age
FROM ( SELECT length_val,
              diameter,
              height,
              whole_weight,
              shucked_weight,
              viscera_weight,
              shell_weight,
              rings
   FROM abalone_xgb WHERE record_number < 2500 )
TARGET rings FUNCTION ml_fn_abalone_xgboost_multi_predict_age
IAM_ROLE default
AUTO OFF
MODEL_TYPE XGBOOST
OBJECTIVE 'multi:softmax'
PREPROCESSORS 'none'
HYPERPARAMETERS DEFAULT EXCEPT (NUM_ROUND '100', NUM_CLASS '30')
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');
```

Contoh berikut menggunakan kueri inferensi untuk memprediksi usia ikan dengan angka rekor lebih besar dari 2500. Ini menggunakan fungsi ml_fn_abalone_xgboost_multi_predict_age yang dibuat dari perintah di atas.

```
select ml_fn_abalone_xgboost_multi_predict_age(length_val,
                                               diameter,
                                               height,
                                               whole_weight,
                                               shucked_weight,
                                               viscera_weight,
                                               shell_weight)+1.5 as age
from abalone_xgb where record_number > 2500;
```

Bawa model Anda sendiri (BYOM) - inferensi lokal

Amazon Redshift ML mendukung penggunaan bring your own model (BYOM) untuk inferensi lokal.

Berikut ini merangkum opsi untuk sintaks CREATE MODEL untuk BYOM. Anda dapat menggunakan model yang dilatih di luar Amazon Redshift dengan Amazon SageMaker AI untuk inferensi dalam database secara lokal di Amazon Redshift.

BUAT sintaks MODEL untuk inferensi lokal

Berikut ini menjelaskan sintaks CREATE MODEL untuk inferensi lokal.

```
CREATE MODEL model_name
FROM ('job_name' | 's3_path' )
FUNCTION function_name ( data_type [, ...] )
RETURNS data_type
IAM_ROLE { default }
[ SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket', | --required
  KMS_KEY_ID 'kms_string') --optional
];
```

Amazon Redshift saat ini hanya mendukung model XGBoost, MLP, dan Linear Learner yang telah dilatih sebelumnya untuk BYOM. Anda dapat mengimpor SageMaker AI Autopilot dan model yang dilatih langsung di Amazon SageMaker AI untuk inferensi lokal menggunakan jalur ini.

BUAT parameter MODEL untuk inferensi lokal

nama_model

Nama modul. Nama model dalam skema harus unik.

DARI ('job_name' | 's3_ path')

Job_name menggunakan nama pekerjaan Amazon SageMaker AI sebagai input. Nama pekerjaan dapat berupa nama pekerjaan pelatihan Amazon SageMaker AI atau nama pekerjaan Amazon SageMaker AI Autopilot. Pekerjaan harus dibuat di AWS akun yang sama yang memiliki cluster Amazon Redshift. Untuk menemukan nama pekerjaan, luncurkan Amazon SageMaker AI. Di menu tar ik -turun Pelatihan, pilih Pekerjaan pelatihan.

's3_path' menentukan lokasi S3 dari file artefak model.tar.gz yang akan digunakan saat membuat model.

Fungsi nam a_fungsi (data_type [,...])

Nama fungsi yang akan dibuat dan tipe data dari argumen input. Anda dapat memberikan nama skema.

RETURNS data_type

Jenis data dari nilai yang dikembalikan oleh fungsi.

<account-id><role-name>IAM_ROLE {default | 'arn:aws:iam: :rol/ '}

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREATE MODEL dijalankan.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi.

PENGATURAN (S3_BUCKET 'amzn-s3-demo-bucket', | KMS_KEY_ID 'kms_string')

Klausa S3_BUCKET menentukan lokasi Amazon S3 yang digunakan untuk menyimpan hasil perantara.

(Opsional) Klausa KMS_KEY_ID menentukan apakah Amazon Redshift menggunakan enkripsi sisi server dengan kunci untuk melindungi data saat diam. AWS KMS Data yang sedang transit dilindungi dengan Secure Sockets Layer (SSL).

Untuk informasi selengkapnya, lihat BUAT MODEL dengan panduan pengguna.

BUAT MODEL untuk contoh inferensi lokal

Contoh berikut membuat model yang sebelumnya telah dilatih di Amazon SageMaker AI, di luar Amazon Redshift. Karena tipe model didukung oleh Amazon Redshift ML untuk inferensi lokal, CREATE MODEL berikut membuat fungsi yang dapat digunakan secara lokal di Amazon Redshift. Anda dapat memberikan nama pekerjaan pelatihan SageMaker AI.

```
CREATE MODEL customer_churn
FROM 'training-job-customer-churn-v4'
FUNCTION customer_churn_predict (varchar, int, float, float)
RETURNS int
IAM_ROLE default
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');
```

Setelah model dibuat, Anda dapat menggunakan fungsi customer_churn_predictive dengan tipe argumen yang ditentukan untuk membuat prediksi.

Bawa model Anda sendiri (BYOM) - inferensi jarak jauh

Amazon Redshift ML juga mendukung penggunaan bring your own model (BYOM) untuk inferensi jarak jauh.

Berikut ini merangkum opsi untuk sintaks CREATE MODEL untuk BYOM.

BUAT sintaks MODEL untuk inferensi jarak jauh

Berikut ini menjelaskan sintaks CREATE MODEL untuk inferensi jarak jauh.

```
CREATE MODEL model_name
FUNCTION function_name ( data_type [, ...] )
RETURNS data_type
SAGEMAKER 'endpoint_name'[:'model_name']
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
[SETTINGS (MAX_BATCH_ROWS integer)];
```

BUAT parameter MODEL untuk inferensi jarak jauh

nama_model

Nama modul. Nama model dalam skema harus unik.

FUNGSI fn_name ([data_type] [,...])

Nama fungsi dan tipe data dari argumen input. Lihat Jenis data untuk semua tipe data yang didukung. `Geography`,`geometry`, `hllsketch` dan tidak didukung.

Anda juga dapat memberikan nama fungsi di dalam skema menggunakan notasi dua bagian, seperti`myschema.myfunction`.

RETURNS data_type

Jenis data dari nilai yang dikembalikan oleh fungsi. Lihat Jenis data untuk semua tipe data yang didukung. `Geography`,`geometry`, `hllsketch` dan tidak didukung.

SAGEMAKER 'nama titik akhir' [: 'nama_model']

Nama titik akhir Amazon SageMaker AI. Jika nama titik akhir menunjuk ke titik akhir multimodel, tambahkan nama model yang akan digunakan. Titik akhir harus di-host di Wil AWS ayah yang

Akun AWS sama dan dengan cluster Amazon Redshift. Untuk menemukan titik akhir Anda, luncurkan Amazon SageMaker AI. Di menu tarik -turun Infer ensi, pilih Titik Akhir.

<account-id><role-name>IAM_ROLE {default | 'arn:aws:iam: :rol/ '}

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREATE MODEL dijalankan. Atau, Anda dapat menentukan ARN dari peran IAM untuk menggunakan peran itu.

MAX_BATCH_ROWS bilangan bulat

Jumlah maksimum baris yang dikirim Amazon Redshift dalam permintaan batch tunggal untuk satu pemanggilan SageMaker AI. Ini didukung hanya untuk BYOM dengan inferensi jarak jauh. Jumlah sebenarnya dari baris dalam batch juga tergantung pada ukuran input, tetapi kurang dari atau sama dengan nilai ini. Nilai minimum parameter ini adalah 1. Nilai maksimumnya adalah`INT_MAX`, atau 2.147.483.647. Parameter ini diperlukan hanya jika tipe data input dan dikembalikan`SUPER`. Nilai defaultnya adalah`INT_MAX`, atau 2.147.483.647.

Saat model digunakan ke titik akhir SageMaker AI, SageMaker AI membuat informasi model di Amazon Redshift. Kemudian melakukan inferensi melalui fungsi eksternal. Anda dapat menggunakan perintah SHOW MODEL untuk melihat informasi model di cluster Amazon Redshift Anda.

BUAT MODEL untuk catatan penggunaan inferensi jarak jauh

Sebelum menggunakan CREATE MODEL untuk inferensi jarak jauh, pertimbangkan hal berikut:

- Titik akhir harus di-host oleh AWS akun yang sama yang memiliki cluster Amazon Redshift.
- Pastikan titik akhir Amazon SageMaker AI memiliki sumber daya yang cukup untuk mengakomodasi panggilan inferensi dari Amazon Redshift atau titik akhir Amazon SageMaker AI dapat diskalakan secara otomatis.
- Jika Anda tidak menggunakan tipe `SUPER` data sebagai input, model hanya menerima input dalam format nilai yang dipisahkan koma (CSV) yang sesuai dengan jenis konten `text/CSV` di SageMaker AI.
- Jika Anda tidak menggunakan tipe `SUPER` data sebagai input, output model adalah nilai tunggal dari jenis yang ditentukan saat Anda membuat fungsi. Outputnya dalam format nilai yang dipisahkan koma (CSV) melalui jenis konten `text/CSV` di SageMaker AI. `VARCHAR`tipe data tidak boleh dalam tanda kutip dan tidak dapat berisi baris baru, dan setiap output harus berada di baris baru.
- Model menerima nol sebagai string kosong.

- Ketika tipe data input adalahSUPER, hanya satu argumen input yang didukung.
- Ketika tipe data input adalahSUPER, tipe data yang dikembalikan juga harusSUPER.
- MAX_BATCH_ROWS diperlukan ketika tipe data input dan yang dikembalikan adalah SUPER.
- Ketika tipe data input adalah tipe konten dari SUPER pemanggilan titik akhir adalah `application/json` ketika MAX_BATCH_ROWS adalah 1 atau dalam semua kasus lainnya. `application/jsonlines`
- Ketika tipe data yang dikembalikan adalah tipe SUPER terima dari pemanggilan titik akhir adalah `application/json` ketika MAX_BATCH_ROWS adalah 1 atau dalam semua kasus lainnya. `application/jsonlines`

BUAT MODEL untuk contoh inferensi jarak jauh

Contoh berikut membuat model yang menggunakan titik akhir SageMaker AI untuk membuat prediksi. Pastikan titik akhir berjalan untuk membuat prediksi dan menentukan namanya dalam perintah CREATE MODEL.

```
CREATE MODEL remote_customer_churn
FUNCTION remote_fn_customer_churn_predict (varchar, int, float, float)
RETURNS int
SAGEMAKER 'customer-churn-endpoint'
IAM_ROLE default;
```

Contoh berikut membuat BYOM dengan inferensi jarak jauh dengan model model bahasa besar (LLM). LLM yang dihosting di Amazon SageMaker AI Jumpstart menerima dan mengembalikan jenis `application/json` konten dan mendukung satu JSON per pemanggilan. Jenis data input dan yang dikembalikan harus SUPER dan MAX_BATCH_ROWS harus disetel ke 1.

```
CREATE MODEL sample_super_data_model
FUNCTION sample_super_data_model_predict(super)
RETURNS super
SAGEMAKER 'sample_super_data_model_endpoint'
IAM_ROLE default
SETTINGS (MAX_BATCH_ROWS 1);
```

BUAT MODEL dengan K-MEANS

Amazon Redshift mendukung K-Means algoritma yang mengelompokkan data yang tidak diberi label. Algoritma ini memecahkan masalah pengelompokan di mana Anda ingin menemukan

pengelompokan dalam data. Data yang tidak diklasifikasikan dikelompokkan dan dipartisi berdasarkan persamaan dan perbedaannya.

BUAT MODEL dengan K-MEANS sintaks

```
CREATE MODEL model_name
FROM { table_name | ( select_statement ) }
FUNCTION function_name
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS 'string'
HYPERPARAMETERS DEFAULT EXCEPT ( K 'val' [, ...] )
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  KMS_KEY_ID 'kms_string', |
    -- optional
  S3_GARBAGE_COLLECT on / off, |
    -- optional
  MAX_CELLS integer, |
    -- optional
  MAX_RUNTIME integer
    -- optional);
```

BUAT MODEL dengan K-MEANS parameter

MATI OTOMATIS

Mematikan penemuan otomatis CREATE MODEL dari pemilihan preprosesor, algoritma, dan hiper-parameter.

MODEL_TYPE KMEAN

Menentukan untuk menggunakan KMEANS untuk melatih model.

PREPROCESSOR 'string'

Menentukan kombinasi tertentu dari preprocessor ke kumpulan kolom tertentu. Format adalah daftar ColumnSets, dan transformasi yang sesuai untuk diterapkan pada setiap set kolom. Amazon Redshift mendukung 3 K-Means preprosesor, yaitu, StandardScaler MinMax, dan. NumericPassthrough Jika Anda tidak ingin menerapkan prapemrosesan apa pun K-Means, pilih NumericPassthrough secara eksplisit sebagai transformator. Untuk informasi selengkapnya tentang transformer yang didukung, lihatBUAT MODEL dengan parameter panduan pengguna.

K-Means Algoritma menggunakan jarak Euclidean untuk menghitung kesamaan. Pra-pemrosesan data memastikan bahwa fitur model tetap pada skala yang sama dan menghasilkan hasil yang dapat diandalkan.

HYPERPARAMETER DEFAULT KECUALI (K 'val' [,...])

Menentukan apakah K-Means parameter digunakan. Anda harus menentukan K parameter saat menggunakan K-Means algoritma. Untuk informasi selengkapnya, lihat K-Means Hyperparameter di Panduan Pengembang Amazon SageMaker AI

Contoh berikut menyiapkan data untuk K-Means.

```
CREATE MODEL customers_clusters
FROM customers
FUNCTION customers_cluster
IAM_ROLE default
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS '[
{
  "ColumnSet": [ "*" ],
  "Transformers": [ "NumericPassthrough" ]
}
]'
HYPERPARAMETERS DEFAULT EXCEPT ( K '5' )
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');

select customer_id, customers_cluster(...) from customers;
customer_id | customers_cluster
--------------------
12345            1
12346            2
12347            4
12348
```

BUAT MODEL dengan Prakiraan

Model perkiraan di Redshift ML menggunakan Amazon Forecast untuk membuat perkiraan deret waktu yang akurat. Dengan demikian, Anda dapat menggunakan data historis selama periode waktu untuk membuat prediksi tentang peristiwa masa depan. Kasus penggunaan umum Amazon Forecast termasuk menggunakan data produk ritel untuk memutuskan bagaimana menentukan

harga persediaan, data kuantitas manufaktur untuk memprediksi berapa banyak dari satu item untuk dipesan, dan data lalu lintas web untuk memperkirakan berapa banyak lalu lintas yang mungkin diterima server web.

Batas kuota dari Amazon Fo recast diberlakukan dalam model perkiraan Amazon Redshift. Misalnya, jumlah perkiraan maksimum adalah 100, tetapi dapat disesuaikan. Menghapus model perkiraan tidak secara otomatis menghapus sumber daya terkait di Amazon Forecast. Jika Anda menghapus cluster Redshift, semua model terkait juga dihapus.

Perhatikan bahwa model Prakiraan saat ini hanya tersedia di Wilayah berikut:

- AS Timur (Ohio) (us-east-2)
- US East (N. Virginia) (us-east-1)
- US West (Oregon) (us-west-2)
- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pasifik (Singapura) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Eropa (Frankfurt) (eu-central-1)
- Eropa (Irlandia) (eu-west-1)

BUAT MODEL dengan sintaks Prakiraan

```
CREATE [ OR REPLACE ] MODEL forecast_model_name
FROM { table_name | ( select_query ) }
TARGET column_name
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>'}
AUTO ON
MODEL_TYPE FORECAST
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  HORIZON integer,
  FREQUENCY forecast_frequency
  [PERCENTILES '0.1', '0.5', '0.9']
  )
```

BUAT MODEL dengan parameter Prakiraan

nama_model_ramalan

Nama modul. Nama model harus unik.

DARI {table_name | (select_query)}

Table_name atau kueri yang menentukan data pelatihan. Ini bisa berupa tabel yang ada di sistem, atau kueri SELECT yang kompatibel dengan Amazon Redshift yang dilampirkan dengan tanda kurung. Tabel atau hasil kueri harus memiliki setidaknya tiga kolom: (1) kolom varchar yang menentukan nama deret waktu. Setiap dataset dapat memiliki beberapa deret waktu; (2) kolom datetime; dan (3) kolom target untuk diprediksi. Kolom target ini harus berupa int atau float. Jika Anda menyediakan dataset yang memiliki lebih dari tiga kolom, Amazon Redshift mengasumsikan bahwa semua kolom tambahan adalah bagian dari deret waktu terkait. Perhatikan bahwa deret waktu terkait harus bertipe int atau float. Untuk informasi selengkapnya tentang deret waktu terkait, lihat Menggunakan Kumpulan Data Seri Waktu Terkait.

Nama kolom TARGET

Nama kolom yang menjadi target prediksi. Kolom harus ada dalam klausa FROM.

<account-id><role-name>IAM_ROLE {default | 'arn:aws:iam: :rol/ '}

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah CREAT MODEL berjalan. Atau, Anda dapat menentukan ARN dari peran IAM untuk menggunakan peran itu.

OTOMATIS MENYALA

Menghidupkan penemuan otomatis CREATE MODEL algoritma dan pemilihan hiper-parameter. Menentukan saat membuat model Prakiraan menunjukkan untuk menggunakan Prakiraan AutoPredictor, di mana Amazon Forecast menerapkan kombinasi algoritma yang optimal untuk setiap deret waktu dalam kumpulan data Anda.

PRAKIRAAN MODEL_TYPE

Menentukan untuk menggunakan FORECAST untuk melatih model.

S3_BUCKET 'amzn-s3-demo-bucket'

Nama bucket Amazon Simple Storage Service yang Anda buat sebelumnya dan yang digunakan untuk berbagi data pelatihan dan artefak antara Amazon Redshift dan Amazon Forecast. Amazon

Redshift membuat subfolder di bucket ini sebelum menurunkan data pelatihan. Saat pelatihan selesai, Amazon Redshift menghapus subfolder yang dibuat dan isinya.

BILANGAN BULAT HORIZON

Jumlah maksimum prediksi yang dapat dikembalikan oleh model perkiraan. Setelah model dilatih, Anda tidak dapat mengubah bilangan bulat ini.

Frekuensi prakiraan_frekuensi

Menentukan seberapa terperinci perkiraan yang Anda inginkan. Opsi yang tersedia adalah`Y | M | W | D | H | 30min | 15min | 10min | 5min | 1min`. Diperlukan jika Anda melatih model perkiraan.

String PERCENTILES

String yang dipisahkan koma yang menentukan jenis perkiraan yang digunakan untuk melatih prediktor. Jenis prakiraan dapat berupa kuantil dari 0,01 hingga 0,99, dengan kenaikan 0,01 atau lebih tinggi. Anda juga dapat menentukan perkiraan rata-rata dengan mean. Anda dapat menentukan maksimal lima jenis perkiraan.

Contoh berikut menunjukkan cara membuat model perkiraan sederhana.

```
CREATE MODEL forecast_example
FROM forecast_electricity_
TARGET target
IAM_ROLE 'arn:aws:iam::<account-id>:role/<role-name>'
AUTO ON
MODEL_TYPE FORECAST
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket',
          HORIZON 24,
          FREQUENCY 'H',
          PERCENTILES '0.25,0.50,0.75,mean',
          S3_GARBAGE_COLLECT OFF);
```

Setelah Anda membuat model perkiraan, Anda dapat membuat tabel baru dengan data prediksi.

```
CREATE TABLE forecast_model_results as SELECT Forecast(forecast_example)
```

Anda kemudian dapat menanyakan tabel baru untuk mendapatkan prediksi.

```
SELECT * FROM forecast_model_results
```

BUAT PROSEDUR

Membuat prosedur tersimpan baru atau mengganti prosedur yang ada untuk database saat ini.

Untuk informasi selengkapnya dan contoh tambahan, lihat Membuat prosedur tersimpan di Amazon Redshift.

Hak istimewa yang diperlukan

Anda harus memiliki izin dengan salah satu cara berikut untuk menjalankan PROSEDUR CREATE OR REPLACE:

- Untuk MEMBUAT PROSEDUR:
  - Superuser
  - Pengguna dengan hak istimewa CREATE dan USAGE pada skema tempat prosedur tersimpan dibuat
- Untuk PROSEDUR PENGGANTI:
  - Superuser
  - Pemilik prosedur

Sintaksis

```
CREATE [ OR REPLACE ] PROCEDURE sp_procedure_name
  ( [ [ argname ] [ argmode ] argtype [, ...] ] )
[ NONATOMIC ]
AS $$
  procedure_body
$$ LANGUAGE plpgsql
[ { SECURITY INVOKER | SECURITY DEFINER } ]
[ SET configuration_parameter { TO value | = value } ]
```

Parameter

ATAU GANTI

Klausa yang menentukan bahwa jika prosedur dengan nama dan tipe data argumen input yang sama, atau tanda tangan, seperti yang ini sudah ada, prosedur yang ada diganti. Anda hanya dapat mengganti prosedur dengan prosedur baru yang mendefinisikan kumpulan tipe data yang identik.

Jika Anda menentukan prosedur dengan nama yang sama dengan prosedur yang ada, tetapi tanda tangan yang berbeda, Anda membuat prosedur baru. Dengan kata lain, nama prosedur kelebihan beban. Untuk informasi selengkapnya, lihat Membebani nama prosedur.

sp_nama_prosedur

Nama prosedurnya. Jika Anda menentukan nama skema (seperti**`myschema.myprocedure`**), prosedur dibuat dalam skema yang ditentukan. Jika tidak, prosedur dibuat dalam skema saat ini. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

Sebaiknya awali semua nama prosedur tersimpan dengan`sp_`. Amazon Redshift menyimpan `sp_` awalan untuk nama prosedur yang disimpan. Dengan menggunakan `sp_` awalan, Anda memastikan bahwa nama prosedur yang disimpan tidak bertentangan dengan nama prosedur atau fungsi tersimpan bawaan Amazon Redshift yang ada atau yang akan datang. Untuk informasi selengkapnya, lihat Penamaan prosedur yang disimpan.

Anda dapat menentukan lebih dari satu prosedur dengan nama yang sama jika tipe data untuk argumen input, atau tanda tangan, berbeda. Dengan kata lain, dalam hal ini nama prosedur kelebihan beban. Untuk informasi selengkapnya, lihat Membebani nama prosedur

[argname] [argmode] argtype

Daftar nama argumen, mode argumen, dan tipe data. Hanya tipe data yang diperlukan. Nama dan mode bersifat opsional dan posisinya dapat dialihkan.

Mode argumen bisa IN, OUT, atau INOUT. Defaultnya adalah IN.

Anda dapat menggunakan argumen OUT dan INOUT untuk mengembalikan satu atau lebih nilai dari panggilan prosedur. Ketika ada argumen OUT atau INOUT, panggilan prosedur mengembalikan satu baris hasil yang berisi n kolom, di mana n adalah jumlah total argumen OUT atau INOUT.

Argumen INOUT adalah argumen input dan output pada saat yang bersamaan. Argumen input mencakup argumen IN dan INOUT, dan argum en output mencakup argumen OUT dan INOUT.

Argumen OUT tidak ditentukan sebagai bagian dari pernyataan CALL. Tentukan argumen INOUT dalam pernyataan CALL prosedur tersimpan. Argumen INOUT dapat berguna saat meneruskan dan mengembalikan nilai dari panggilan bersarang, dan juga saat mengembalikan a. `refcursor` Untuk informasi selengkapnya tentang `refcursor` jenis, lihatCursors.

Jenis data argumen dapat berupa tipe data Amazon Redshift standar apa pun. Selain itu, tipe data argumen bisa`refcursor`.

Anda dapat menentukan maksimum 32 argumen input dan 32 argumen output.

AS $$ prosedur _badan $$

Konstruksi yang menyertakan prosedur yang akan dijalankan. Kata kunci literal AS $$ dan $$ diperlukan.

Amazon Redshift mengharuskan Anda untuk melampirkan pernyataan dalam prosedur Anda dengan menggunakan format yang disebut kuotasi dolar. Apa pun di dalam kandang dilewatkan persis apa adanya. Anda tidak perlu melarikan diri dari karakter khusus karena isi string ditulis secara harfiah.

Dengan kutipan dolar, Anda menggunakan sepasang tanda dolar ($$) untuk menandakan awal dan akhir pernyataan yang akan dijalankan, seperti yang ditunjukkan pada contoh berikut.

```
$$ my statement $$
```

Secara opsional, di antara tanda-tanda dolar di setiap pasangan, Anda dapat menentukan string untuk membantu mengidentifikasi pernyataan. String yang Anda gunakan harus sama di awal dan akhir pasangan penutup. String ini peka huruf besar-kecil, dan mengikuti batasan yang sama dengan pengidentifikasi yang tidak dikutip kecuali bahwa itu tidak dapat berisi tanda dolar. Contoh berikut menggunakan uji string.

```
$test$ my statement $test$
```

Sintaks ini juga berguna untuk kutipan dolar bersarang. Untuk informasi lebih lanjut tentang kutipan dolar, lihat "Konstanta Dollar-quoted String” di bawah Struktur Lek sikal dalam dokumentasi PostgreSQL.

badan prosedur

Satu set PL/pgSQL pernyataan yang valid. PL/pgSQL pernyataan menambah perintah SQL dengan konstruksi prosedural, termasuk ekspresi perulangan dan kondisional, untuk mengontrol aliran logis. Sebagian besar perintah SQL dapat digunakan dalam badan prosedur, termasuk bahasa modifikasi data (DML) seperti COPY, UNLOAD dan INSERT, dan bahasa definisi data (DDL) seperti CREATE TABLE. Untuk informasi selengkapnya, lihat PL/pgSQL referensi bahasa.

BAHASA plpgsql

Nilai bahasa. Tentukan `plpgsql`. Anda harus memiliki izin untuk penggunaan bahasa untuk digunakan`plpgsql`. Untuk informasi selengkapnya, lihat HIBAH.

NONATOMIK

Membuat prosedur tersimpan dalam mode transaksi nonatomik. Mode NONATOMIC secara otomatis mengkomit pernyataan di dalam prosedur. Selain itu, ketika kesalahan terjadi di dalam prosedur NONATOMIC, kesalahan tidak dilemparkan kembali jika ditangani oleh blok pengecualian. Untuk informasi selengkapnya, lihat Mengelola transaksi dan MENAIKKAN.

Saat Anda mendefinisikan prosedur tersimpan sebagai`NONATOMIC`, pertimbangkan hal berikut:

- Saat Anda memasukkan panggilan prosedur tersimpan, semua prosedur harus dibuat dalam mode transaksi yang sama.
- `SECURITY DEFINER`Opsi dan `SET configuration_parameter` opsi tidak didukung saat membuat prosedur dalam mode NONATOMIC.
- Setiap kursor yang dibuka (eksplisit atau implisit) ditutup secara otomatis ketika komit implisit diproses. Oleh karena itu, Anda harus membuka transaksi eksplisit sebelum memulai loop kursor untuk memastikan bahwa setiap SQL dalam iterasi loop tidak terikat secara implisit.

PEMANGGIL KEAMANAN | PENENTU KEAMANAN

`SECURITY DEFINER`Opsi ini tidak didukung ketika `NONATOMIC` ditentukan.

Mode keamanan untuk prosedur menentukan hak akses prosedur saat runtime. Prosedur harus memiliki izin untuk mengakses objek database yang mendasarinya.

Untuk mode SECURITY INVOKER, prosedur menggunakan hak istimewa pengguna yang memanggil prosedur. Pengguna harus memiliki izin eksplisit pada objek database yang mendasarinya. Standarnya adalah SECURITY INVOKER.

Untuk mode SECURITY DEFINER, prosedur menggunakan hak istimewa pemilik prosedur. Pemilik prosedur didefinisikan sebagai pengguna yang memiliki prosedur pada waktu berjalan, belum tentu pengguna yang awalnya mendefinisikan prosedur. Pengguna yang memanggil prosedur memerlukan hak istimewa eksekusi pada prosedur, tetapi tidak memerlukan hak istimewa apa pun pada objek yang mendasarinya.

SETEL configuration_parameter {KE nilai | = nilai}

Opsi ini tidak didukung ketika `NONATOMIC` ditentukan.

Klausa SET menyebabkan yang ditentukan diatur `configuration_parameter` ke nilai yang ditentukan saat prosedur dimasukkan. Klausa ini kemudian mengembalikan `configuration_parameter` ke nilai sebelumnya ketika prosedur keluar.

Catatan penggunaan

Jika prosedur tersimpan dibuat menggunakan opsi SECURITY DEFINER, saat memanggil fungsi CURRENT_USER dari dalam prosedur tersimpan, Amazon Redshift mengembalikan nama pengguna pemilik prosedur tersimpan.

Contoh

> **Note**
>
> Jika saat menjalankan contoh ini Anda mengalami kesalahan yang mirip dengan:
>
> ```
> ERROR: 42601: [Amazon](500310) unterminated dollar-quoted string at or near "$$
> ```
>
> Lihat Ikhtisar prosedur tersimpan di Amazon Redshift.

Contoh berikut membuat prosedur dengan dua parameter input.

```
CREATE OR REPLACE PROCEDURE test_sp1(f1 int, f2 varchar(20))
AS $$
DECLARE
  min_val int;
BEGIN
  DROP TABLE IF EXISTS tmp_tbl;
  CREATE TEMP TABLE tmp_tbl(id int);
  INSERT INTO tmp_tbl values (f1),(10001),(10002);
  SELECT min_val MIN(id) FROM tmp_tbl;
  RAISE INFO 'min_val = %, f2 = %', min_val, f2;
END;
$$ LANGUAGE plpgsql;
```

> **Note**
>
> Saat Anda menulis prosedur tersimpan, kami merekomendasikan praktik terbaik untuk mengamankan nilai-nilai sensitif:
> Jangan membuat kode keras informasi sensitif apa pun dalam logika prosedur yang disimpan. Misalnya, jangan menetapkan kata sandi pengguna dalam pernyataan CREATE USER di badan prosedur yang disimpan. Ini menimbulkan risiko keamanan, karena nilai hard-code dapat direkam sebagai metadata skema dalam tabel katalog. Sebagai gantinya,

berikan nilai sensitif, seperti kata sandi, sebagai argumen ke prosedur yang disimpan, melalui parameter.
Untuk informasi selengkapnya tentang prosedur tersimpan, lihat MEMBUAT PROSEDUR dan Membuat prosedur tersimpan di Amazon Redshift. Untuk informasi selengkapnya tentang tabel katalog, lihat Tabel katalog sistem.

Contoh berikut membuat prosedur dengan satu parameter IN, satu parameter OUT, dan satu parameter INOUT.

```
CREATE OR REPLACE PROCEDURE test_sp2(f1 IN int, f2 INOUT varchar(256), out_var OUT
 varchar(256))
AS $$
DECLARE
  loop_var int;
BEGIN
  IF f1 is null OR f2 is null THEN
    RAISE EXCEPTION 'input cannot be null';
  END IF;
  DROP TABLE if exists my_etl;
  CREATE TEMP TABLE my_etl(a int, b varchar);
    FOR loop_var IN 1..f1 LOOP
        insert into my_etl values (loop_var, f2);
        f2 := f2 || '+' || f2;
    END LOOP;
  SELECT INTO out_var count(*) from my_etl;
END;
$$ LANGUAGE plpgsql;
```

Contoh berikut membuat prosedur yang menggunakan `SECURITY DEFINER` parameter. Prosedur ini berjalan menggunakan hak istimewa pengguna yang memiliki prosedur.

```
CREATE OR REPLACE PROCEDURE sp_get_current_user_definer()
AS $$
DECLARE curr_user varchar(250);
BEGIN
  SELECT current_user INTO curr_user;
  RAISE INFO '%', curr_user;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER;
```

Contoh berikut membuat prosedur yang menggunakan `SECURITY INVOKER` parameter. Prosedur ini berjalan menggunakan hak istimewa pengguna yang menjalankan prosedur.

```
CREATE OR REPLACE PROCEDURE sp_get_current_user_invoker()
AS $$
DECLARE curr_user varchar(250);
BEGIN
  SELECT current_user INTO curr_user;
  RAISE INFO '%', curr_user;
END;
$$ LANGUAGE plpgsql
SECURITY INVOKER;
```

BUAT KEBIJAKAN RLS

Membuat kebijakan keamanan tingkat baris baru untuk menyediakan akses granular ke objek database.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat membuat kebijakan.

Sintaksis

```
CREATE RLS POLICY { policy_name | database_name.policy_name }
[ WITH (column_name data_type [, ...]) [ [AS] relation_alias ] ]
USING ( using_predicate_exp )
```

Parameter

nama_kebijakan

Nama kebijakan .

database_name

Nama database tempat kebijakan akan dibuat. Kebijakan dapat dibuat pada database yang terhubung atau pada database yang mendukung izin federasi Amazon Redshift.

DENGAN (column_name data_type [,...])

Menentukan column_name dan data_type yang di referensikan ke kolom tabel yang dilampirkan kebijakan.

Anda dapat menghilangkan klausa WITH hanya jika kebijakan RLS tidak mereferensikan kolom tabel mana pun yang dilampirkan kebijakan.

AS relation_alias

Menentukan alias opsional untuk tabel yang akan dilampirkan kebijakan RLS.

MENGGUNAKAN (menggunakan_predic ate_exp)

Menentukan filter yang diterapkan pada klausa WHERE dari query. Amazon Redshift menerapkan predikat kebijakan sebelum predikat pengguna tingkat kueri. Misalnya, **`current_user = 'joe' and price > 10`** membatasi Joe untuk hanya melihat catatan dengan harga lebih besar dari $10.

Untuk penggunaan KEBIJAKAN BUAT RLS di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

Catatan penggunaan

Saat bekerja dengan pernyataan CREATE RLS POLICY, perhatikan hal berikut:

- Amazon Redshift mendukung filter yang dapat menjadi bagian dari klausa WHERE dari kueri.
- Semua kebijakan yang dilampirkan ke tabel harus dibuat dengan alias tabel yang sama.
- Anda harus menggunakan pernyataan GRANT dan REVOKE untuk secara eksplisit memberikan dan mencabut izin SELECT ke kebijakan RLS yang mereferensikan tabel pencarian. Tabel pencarian adalah objek tabel yang digunakan dalam definisi kebijakan. Untuk informasi selengkapnya, lihat the section called “HIBAH” dan the section called “MENCABUT”.
- Keamanan tingkat baris Amazon Redshift tidak mendukung tipe objek berikut di dalam definisi kebijakan: tabel katalog, relasi lintas database, tabel eksternal, tampilan reguler, tampilan pengikatan terlambat, tabel dengan kebijakan RLS diaktifkan, dan tabel sementara.

Contoh

Contoh berikut membuat kebijakan RLS yang disebut policy_concerts. Kebijakan ini berlaku untuk kolom VARCHAR (10) yang disebut catgroup dan menyetel filter USING untuk hanya mengembalikan baris di mana nilai catgroup berada. `'Concerts'`

```
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');
```

Untuk contoh end-to-end menggunakan kebijakan RLS, lihatRow-level contoh keamanan ujung ke ujung.

CREATE ROLE

Membuat peran kustom baru yang merupakan kumpulan izin. Untuk daftar peran yang ditentukan sistem Amazon Redshift, lihat. the section called “Peran yang ditentukan sistem Amazon Redshift” Kueri SVV_ROLE untuk melihat peran yang saat ini dibuat di cluster atau kelompok kerja Anda.

Ada kuota jumlah peran yang dapat dibuat. Untuk informasi selengkapnya, lihat Kuota dan batasan di Amazon Redshift dalam Panduan Manajemen Amazon Redshift.

Izin yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE ROLE.

- Superuser
- Pengguna dengan hak istimewa CREATE ROLE

Sintaksis

```
CREATE ROLE role_name
[ EXTERNALID external_id ]
```

Parameter

role_name

Nama perannya. Nama peran harus unik dan tidak boleh sama dengan nama pengguna mana pun. Nama peran tidak boleh menjadi kata yang dicadangkan.

Pengguna super atau pengguna biasa dengan hak istimewa CREATE ROLE dapat membuat peran. Pengguna yang bukan superuser tetapi telah diberikan USE ke peran WITH GRANT OPTION dan ALTER hak istimewa dapat memberikan peran ini kepada siapa pun.

EXTERNALID_id eksternal

Pengidentifikasi untuk peran, yang terkait dengan penyedia identitas. Untuk informasi selengkapnya, lihat Federasi penyedia identitas asli (IdP) untuk Amazon Red shift.

Contoh

Contoh berikut menciptakan peran`sample_role1`.

```
CREATE ROLE sample_role1;
```

Contoh berikut membuat peran`sample_role1`, dengan ID eksternal yang terkait dengan penyedia identitas.

```
CREATE ROLE sample_role1 EXTERNALID "ABC123";
```

BUAT SKEMA

Mendefinisikan skema baru untuk database saat ini.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE SCHEMA:

- Superuser
- Pengguna dengan hak istimewa CREATE SCHEMA

Sintaksis

```
CREATE SCHEMA [ IF NOT EXISTS ] schema_name [ AUTHORIZATION username ]
           [ QUOTA {quota [MB | GB | TB] | UNLIMITED} ] [ schema_element [ ... ]

CREATE SCHEMA AUTHORIZATION username[ QUOTA {quota [MB | GB | TB] | UNLIMITED} ]
 [ schema_element [ ... ] ]
```

Parameter

JIKA TIDAK ADA

Klausa yang menunjukkan bahwa jika skema yang ditentukan sudah ada, perintah tidak boleh membuat perubahan dan mengembalikan pesan bahwa skema itu ada, daripada diakhiri dengan kesalahan.

Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika CREATE SCHEMA mencoba membuat skema yang sudah ada.

nama skema

Nama skema baru. Nama skema tidak bisa`PUBLIC`. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

> **Note**
>
> Daftar skema dalam parameter search_path konfigurasi menentukan prioritas objek dengan nama identik ketika mereka direferensikan tanpa nama skema.

OTORISASI

Klausul yang memberikan kepemilikan kepada pengguna tertentu.

username

Nama pemilik skema.

element_skema

Definisi untuk satu atau lebih objek yang akan dibuat dalam skema.

KUOTA

Jumlah maksimum ruang disk yang dapat digunakan skema tertentu. Ruang ini adalah penggunaan disk kolektif. Ini mencakup semua tabel permanen, tampilan terwujud di bawah skema yang ditentukan, dan salinan duplikat dari semua tabel dengan distribusi SEMUA pada setiap node komputasi. Kuota skema tidak memperhitungkan tabel sementara yang dibuat sebagai bagian dari namespace atau skema sementara.

Untuk melihat kuota skema yang dikonfigurasi, lihatSVV_SCHEMA_KUOTA_STATUS.

Untuk melihat catatan di mana kuota skema terlampaui, lihatPELANGGARAN STL_SCHEMA_KUOTA_.

Amazon Redshift mengonversi nilai yang dipilih menjadi megabyte. Gigabyte adalah unit pengukuran default ketika Anda tidak menentukan nilai.

Anda harus menjadi superuser database untuk mengatur dan mengubah kuota skema. Pengguna yang bukan superuser tetapi memiliki izin CREATE SCHEMA dapat membuat skema dengan kuota yang ditentukan. Saat Anda membuat skema tanpa menentukan kuota, skema memiliki kuota tak terbatas. Saat Anda menetapkan kuota di bawah nilai saat ini yang digunakan oleh skema, Amazon Redshift tidak mengizinkan penyerapan lebih lanjut sampai Anda mengosongkan ruang disk. Pernyataan DELETE menghapus data dari tabel dan ruang disk dibebaskan hanya ketika VACUUM berjalan.

Amazon Redshift memeriksa setiap transaksi untuk pelanggaran kuota sebelum melakukan transaksi. Amazon Redshift memeriksa ukuran (ruang disk yang digunakan oleh semua tabel dalam skema) dari setiap skema yang dimodifikasi terhadap kuota yang ditetapkan. Karena pemeriksaan pelanggaran kuota terjadi di akhir transaksi, batas ukuran dapat melebihi kuota sementara dalam transaksi sebelum dilakukan. Ketika transaksi melebihi kuota, Amazon Redshift menghentikan transaksi, melarang konsumsi berikutnya, dan mengembalikan semua perubahan sampai Anda mengosongkan ruang disk. Karena VACUUM latar belakang dan pembersihan internal, ada kemungkinan bahwa skema tidak penuh pada saat Anda memeriksa skema setelah transaksi dibatalkan.

Sebagai pengecualian, Amazon Redshift mengabaikan pelanggaran kuota dan melakukan transaksi dalam kasus-kasus tertentu. Amazon Redshift melakukan ini untuk transaksi yang hanya terdiri dari satu atau lebih pernyataan berikut di mana tidak ada pernyataan penyerapan INSERT atau COPY dalam transaksi yang sama:

- DELETE
- MEMOTONG
- VAKUM
- MEJA DROP
- ALTER TABLE APPEND hanya saat memindahkan data dari skema lengkap ke skema non-lengkap lainnya

TAK TERBATAS

Amazon Redshift tidak membatasi pertumbuhan ukuran total skema.

Batas

Amazon Redshift memberlakukan batasan berikut untuk skema.

- Ada maksimal 9900 skema per database.

Contoh

Contoh berikut membuat skema bernama US_SALES dan memberikan kepemilikan kepada pengguna DWUSER.

```
create schema us_sales authorization dwuser;
```

Contoh berikut membuat skema bernama US_SALES, memberikan kepemilikan kepada pengguna DWUSER, dan menetapkan kuota ke 50 GB.

```
create schema us_sales authorization dwuser QUOTA 50 GB;
```

Untuk melihat skema baru, kueri tabel katalog PG_NAMESPACE seperti yang ditunjukkan berikut.

```
select nspname as schema, usename as owner
from pg_namespace, pg_user
where pg_namespace.nspowner = pg_user.usesysid
and pg_user.usename ='dwuser';

   schema |  owner
----------+----------
 us_sales | dwuser
(1 row)
```

Contoh berikut membuat skema US_SALES, atau tidak melakukan apa-apa dan mengembalikan pesan jika sudah ada.

```
create schema if not exists us_sales;
```

CREATE TABLE

Membuat tabel baru di database saat ini. Anda menentukan daftar kolom, yang masing-masing menyimpan data dari jenis yang berbeda. Pemilik tabel adalah penerbit perintah CREATE TABLE.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE TABLE:

- Superuser
- Pengguna dengan hak istimewa CREATE TABLE

Sintaksis

```
CREATE [ [LOCAL ] { TEMPORARY | TEMP } ] TABLE
[ IF NOT EXISTS ] table_name
( { column_name data_type [column_attributes] [ column_constraints ]
  | table_constraints
  | LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ] }
  [, ... ]  )
[ BACKUP { YES | NO } ]
[table_attributes]

where column_attributes are:
  [ DEFAULT default_expr ]
  [ IDENTITY ( seed, step ) ]
  [ GENERATED BY DEFAULT AS IDENTITY ( seed, step ) ]
  [ ENCODE encoding ]
  [ DISTKEY ]
  [ SORTKEY ]
  [ COLLATE { CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI } ]

and column_constraints are:
  [ { NOT NULL | NULL } ]
  [ { UNIQUE  |  PRIMARY KEY } ]
  [ REFERENCES reftable [ ( refcolumn ) ] ]

and table_constraints  are:
  [ UNIQUE ( column_name [, ... ] ) ]
  [ PRIMARY KEY ( column_name [, ... ] )  ]
  [ FOREIGN KEY (column_name [, ... ] ) REFERENCES reftable [ ( refcolumn ) ]


and table_attributes are:
  [ DISTSTYLE { AUTO | EVEN | KEY | ALL } ]
  [ DISTKEY ( column_name ) ]
  [ [COMPOUND | INTERLEAVED ] SORTKEY ( column_name [,...]) |  [ SORTKEY AUTO ] ]
```

```
  [ ENCODE AUTO ]
```

Parameter

LOKAL

Tidak wajib. Meskipun kata kunci ini diterima dalam pernyataan, kata kunci ini tidak berpengaruh di Amazon Redshift.

SEMENTARA | TEMP

Kata kunci yang membuat tabel sementara yang hanya terlihat dalam sesi saat ini. Tabel secara otomatis dijatuhkan pada akhir sesi di mana ia dibuat. Tabel sementara dapat memiliki nama yang sama dengan tabel permanen. Tabel sementara dibuat dalam skema khusus sesi yang terpisah. (Anda tidak dapat menentukan nama untuk skema ini.) Skema sementara ini menjadi skema pertama di jalur pencarian, sehingga tabel sementara lebih diutamakan daripada tabel permanen kecuali Anda memenuhi syarat nama tabel dengan nama skema untuk mengakses tabel permanen. Untuk informasi selengkapnya tentang skema dan prioritas, lihat. search_path

> **Note**
>
> Secara default, pengguna database memiliki izin untuk membuat tabel sementara dengan keanggotaan otomatis mereka di grup PUBLIK. Untuk menolak hak istimewa ini kepada pengguna, cabut hak istimewa TEMP dari grup PUBLIK, dan kemudian secara eksplisit berikan hak istimewa TEMP hanya kepada pengguna atau grup pengguna tertentu.

JIKA TIDAK ADA

Klausa yang menunjukkan bahwa jika tabel yang ditentukan sudah ada, perintah tidak boleh membuat perubahan dan mengembalikan pesan bahwa tabel ada, daripada berhenti dengan kesalahan. Perhatikan bahwa tabel yang ada mungkin tidak seperti yang akan dibuat; hanya nama tabel yang dibandingkan.

Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika CREATE TABLE mencoba membuat tabel yang sudah ada.

table_name

Nama tabel yang akan dibuat.

> **Important**
>
> Jika Anda menentukan nama tabel yang dimulai dengan '# ', tabel dibuat sebagai tabel sementara. Berikut ini adalah contohnya:
>
> ```
> create table #newtable (id int);
> ```
>
> Anda juga mereferensikan tabel dengan '# '. Contoh:
>
> ```
> select * from #newtable;
> ```

Panjang maksimum untuk nama tabel adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Anda dapat menggunakan karakter UTF-8 multibyte hingga maksimal empat byte. Amazon Redshift memberlakukan kuota jumlah tabel per cluster berdasarkan jenis node, termasuk tabel sementara yang ditentukan pengguna dan tabel sementara yang dibuat oleh Amazon Redshift selama pemrosesan kueri atau pemeliharaan sistem. Secara opsional, nama tabel dapat memenuhi syarat dengan database dan nama skema. Dalam contoh berikut, nama database adalah`tickit`, nama skema adalah`public`, dan nama tabel adalah`test`.

```
create table tickit.public.test (c1 int);
```

Jika database atau skema tidak ada, tabel tidak dibuat, dan pernyataan mengembalikan kesalahan. Anda tidak dapat membuat tabel atau tampilan di database sistem`template0`,`template1`,`padb_harvest`, atau`sys:internal`.

Jika nama skema diberikan, tabel baru dibuat dalam skema itu (dengan asumsi pembuat memiliki akses ke skema). Nama tabel harus berupa nama unik untuk skema itu. Jika tidak ada skema yang ditentukan, tabel dibuat dengan menggunakan skema database saat ini. Jika Anda membuat tabel sementara, Anda tidak dapat menentukan nama skema, karena tabel sementara ada dalam skema khusus.

Beberapa tabel sementara dengan nama yang sama dapat ada pada waktu yang sama dalam database yang sama jika mereka dibuat dalam sesi terpisah karena tabel ditetapkan ke skema yang berbeda. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

nama_kolom

Nama kolom yang akan dibuat di tabel baru. Panjang maksimum untuk nama kolom adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Anda dapat menggunakan karakter UTF-8 multibyte hingga maksimal empat byte. Jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.600. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

> **Note**
>
> Jika Anda membuat “tabel lebar”, berhati-hatilah agar daftar kolom Anda tidak melebihi batas lebar baris untuk hasil perantara selama pemuatan dan pemrosesan kueri. Untuk informasi selengkapnya, lihat Catatan penggunaan.

tipe_data

Jenis data kolom yang sedang dibuat. Untuk kolom CHAR dan VARCHAR, Anda dapat menggunakan kata kunci MAX alih-alih mendeklarasikan panjang maksimum. MAX menetapkan panjang maksimum ke 4.096 byte untuk CHAR atau 65535 byte untuk VARCHAR. Ukuran maksimum objek GEOMETRY adalah 1.048.447 byte.

Untuk informasi tentang tipe data yang didukung Amazon Redshift, lihatJenis Data.

DEFAULT default_expr

Klausa yang menetapkan nilai data default untuk kolom. Tipe data default_expr harus sesuai dengan tipe data kolom. Nilai DEFAULT harus berupa ekspresi bebas variabel. Subquery, referensi silang ke kolom lain dalam tabel saat ini, dan fungsi yang ditentukan pengguna tidak diperbolehkan.

Ek spresi default_expr digunakan dalam operasi INSERT apa pun yang tidak menentukan nilai untuk kolom. Jika tidak ada nilai default yang ditentukan, nilai default untuk kolom adalah null.

Jika operasi COPY dengan daftar kolom yang ditentukan menghilangkan kolom yang memiliki nilai DEFAULT, perintah COPY menyisipkan nilai default _expr.

IDENTITAS (benih, langkah)

Klausa yang menentukan bahwa kolom adalah kolom IDENTITAS. Kolom IDENTITY berisi nilai unik yang dibuat secara otomatis. Tipe data untuk kolom IDENTITY harus berupa INT atau BIGINT.

Saat Anda menambahkan baris menggunakan `INSERT INTO [tablename] VALUES()` pernyataan `INSERT` atau, nilai-nilai ini dimulai dengan nilai yang ditentukan sebagai benih dan bertambah dengan angka yang ditentukan sebagai langkah.

Ketika Anda memuat tabel menggunakan `COPY` pernyataan `INSERT INTO [tablename] SELECT * FROM` atau, data dimuat secara paralel dan didistribusikan ke irisan node. Untuk memastikan bahwa nilai identitas unik, Amazon Redshift melewatkan sejumlah nilai saat membuat nilai identitas. Nilai identitas unik, tetapi urutannya mungkin tidak cocok dengan urutan dalam file sumber.

DIHASILKAN SECARA DEFAULT SEBAGAI IDENTITAS (benih, langkah)

Klausa yang menentukan bahwa kolom adalah kolom IDENTITAS default, sehingga Anda dapat secara otomatis menetapkan nilai unik ke kolom. Tipe data untuk kolom IDENTITY harus berupa INT atau BIGINT. Ketika Anda menambahkan baris tanpa nilai, nilai-nilai ini dimulai dengan nilai yang ditentukan sebagai benih dan bertambah dengan angka yang ditentukan sebagai langkah. Untuk informasi tentang bagaimana nilai dihasilkan, lihatIDENTITY.

Juga, selama INSERT, UPDATE, atau COPY Anda dapat memberikan nilai tanpa EXPLICIT_IDS. Amazon Redshift menggunakan nilai itu untuk menyisipkan ke kolom identitas alih-alih menggunakan nilai yang dihasilkan sistem. Nilai dapat berupa duplikat, nilai kurang dari benih, atau nilai antara nilai-nilai langkah. Amazon Redshift tidak memeriksa keunikan nilai di kolom. Memberikan nilai tidak mempengaruhi nilai yang dihasilkan sistem berikutnya.

> **Note**
>
> Jika Anda memerlukan keunikan di kolom, jangan tambahkan nilai duplikat. Sebagai gantinya, tambahkan nilai unik yang kurang dari benih atau di antara nilai langkah.

Ingatlah hal berikut tentang kolom identitas default:

- Kolom identitas default BUKAN NULL. NULL tidak dapat dimasukkan.
- Untuk memasukkan nilai yang dihasilkan ke kolom identitas default, gunakan kata kunciDEFAULT.

```
INSERT INTO tablename (identity-column-name) VALUES (DEFAULT);
```

- Mengganti nilai kolom identitas default tidak mempengaruhi nilai yang dihasilkan berikutnya.

- Anda tidak dapat menambahkan kolom identitas default dengan pernyataan ALTER TABLE ADD COLUMN.
- Anda dapat menambahkan kolom identitas default dengan pernyataan ALTER TABLE APPEND.

ENCODE pengkodean

Pengkodean kompresi untuk kolom. ENCODE AUTO adalah default untuk tabel. Amazon Redshift secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel. Jika Anda menentukan pengkodean kompresi untuk kolom apa pun dalam tabel, tabel tidak lagi disetel ke ENCODE AUTO. Amazon Redshift tidak lagi secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel. Anda dapat menentukan opsi ENCODE AUTO untuk tabel untuk mengaktifkan Amazon Redshift untuk mengelola pengkodean kompresi secara otomatis untuk semua kolom dalam tabel.

Amazon Redshift secara otomatis menetapkan pengkodean kompresi awal ke kolom yang tidak Anda tentukan pengkodean kompresi sebagai berikut:

- Semua kolom dalam tabel sementara diberi kompresi RAW secara default.
- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, DOUBLE PRECISION, GEOMETRY, atau GEOGRAPHY diberi kompresi RAW.
- Kolom yang didefinisikan sebagai SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai CHAR, VARCHAR, atau VARBYTE diberi kompresi LZO.

> **Note**
>
> Jika Anda tidak ingin kolom dikompresi, tentukan pengkodean RAW secara eksplisit.

compression encodings (p. 78) berikut didukung:

- AZ64
- BYTEDICT
- DELTA
- DELTA32K

- LZO
- MAYORIT8
- MAYORIT16
- MAYORIT32
- RAW (tanpa kompresi)
- PANJANG
- TEKS255
- TEKS32K
- ZSTD

DISTKEY

Kata kunci yang menentukan bahwa kolom adalah kunci distribusi untuk tabel. Hanya satu kolom dalam tabel yang dapat menjadi kunci distribusi. Anda dapat menggunakan kata kunci DISTKEY setelah nama kolom atau sebagai bagian dari definisi tabel dengan menggunakan sintaks DISTKEY (column _name). Salah satu metode memiliki efek yang sama. Untuk informasi selengkapnya, lihat parameter DISTSTYLE nanti dalam topik ini.

Tipe data kolom kunci distribusi dapat berupa: BOOLEAN, REAL, DOUBLE PRECISION, SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ, CHAR, atau VARCHAR.

SORTKEY

Kata kunci yang menentukan bahwa kolom adalah kunci pengurutan untuk tabel. Ketika data dimuat ke dalam tabel, data diurutkan berdasarkan satu atau lebih kolom yang ditetapkan sebagai kunci pengurutan. Anda dapat menggunakan kata kunci SORTKEY setelah nama kolom untuk menentukan kunci pengurutan kolom tunggal, atau Anda dapat menentukan satu atau lebih kolom sebagai kolom kunci pengurutan untuk tabel dengan menggunakan sintaks SORTKEY (column_ name [,...]). Hanya kunci pengurutan majemuk yang dibuat dengan sintaks ini.

Anda dapat menentukan maksimum 400 kolom SORTKEY per tabel.

Tipe data kolom kunci pengurutan dapat berupa: BOOLEAN, REAL, DOUBLE PRECISION, SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ, CHAR, atau VARCHAR.

SUSUN {CASE_SENSITIVE | CS | CASE_INSENSITIVE | CI}

Klausa yang menentukan apakah pencarian string atau perbandingan pada kolom peka huruf besar/kecil atau tidak peka huruf besar/kecil. Nilai defaultnya sama dengan konfigurasi sensitivitas huruf besar saat ini dari database.

COLLATE hanya didukung pada tipe data berbasis string, termasuk CHAR, VARCHAR, dan nilai string dalam kolom SUPER. Untuk detail tentang kueri data SUPER yang tidak peka huruf besar/kecil, lihat. Case-insensitive menanyakan

Untuk memeriksa kolasi database saat ini, gunakan DB_COLLATION fungsi.

CASE_SENSITIVE dan CS dapat dipertukarkan dan menghasilkan hasil yang sama. Demikian pula, CASE_INSENSITIVE dan CI dapat dipertukarkan dan menghasilkan hasil yang sama.

BUKAN NULL | NULL

NOT NULL menentukan bahwa kolom tidak diizinkan untuk berisi nilai null. NULL, default, menentukan bahwa kolom menerima nilai null. Kolom IDENTITY dinyatakan NOT NULL secara default.

UNIK

Kata kunci yang menentukan bahwa kolom hanya dapat berisi nilai unik. Perilaku kendala tabel unik sama dengan batasan kolom, dengan kemampuan tambahan untuk menjangkau beberapa kolom. Untuk menentukan batasan tabel unik, gunakan sintaks UNIQUE (column_name [,...]).

> **Important**
>
> Kendala unik bersifat informasi dan tidak ditegakkan oleh sistem.

KUNCI UTAMA

Kata kunci yang menentukan bahwa kolom adalah kunci utama untuk tabel. Hanya satu kolom yang dapat didefinisikan sebagai kunci utama dengan menggunakan definisi kolom. Untuk menentukan batasan tabel dengan kunci primer beberapa kolom, gunakan sintaks PRIMARY KEY (column_name [,...]).

Mengidentifikasi kolom sebagai kunci utama menyediakan metadata tentang desain skema. Kunci utama menyiratkan bahwa tabel lain dapat mengandalkan kumpulan kolom ini sebagai

pengidentifikasi unik untuk baris. Satu kunci utama dapat ditentukan untuk tabel, baik sebagai batasan kolom atau batasan tabel. Kendala kunci utama harus memberi nama sekumpulan kolom yang berbeda dari kumpulan kolom lain yang dinamai oleh batasan unik apa pun yang ditentukan untuk tabel yang sama.

Kolom PRIMARY KEY juga didefinisikan sebagai NOT NULL.

> **Important**
>
> Kendala kunci utama hanya bersifat informasi. Mereka tidak ditegakkan oleh sistem, tetapi mereka digunakan oleh perencana.

Referensi dapat direvisi [(refcolumn)]

Klausa yang menentukan batasan kunci asing, yang menyiratkan bahwa kolom harus berisi hanya nilai yang cocok dengan nilai dalam kolom yang direferensikan dari beberapa baris tabel yang direferensikan. Kolom yang direferensikan harus berupa kolom dari batasan kunci unik atau primer dalam tabel yang direferensikan.

> **Important**
>
> Kendala kunci asing hanya bersifat informasi. Mereka tidak ditegakkan oleh sistem, tetapi mereka digunakan oleh perencana.

LIKE parent_table [{TERMASUK | TIDAK TERMASUK} DEFAULT]

Klausa yang menentukan tabel yang ada dari mana tabel baru secara otomatis menyalin nama kolom, tipe data, dan batasan NOT NULL. Tabel baru dan tabel induk dipisahkan, dan setiap perubahan yang dilakukan pada tabel induk tidak diterapkan ke tabel baru. Ekspresi default untuk definisi kolom yang disalin hanya disalin jika TERMASUK DEFAULTS ditentukan. Perilaku defaultnya adalah mengecualikan ekspresi default, sehingga semua kolom tabel baru memiliki default nol.

Tabel yang dibuat dengan opsi LIKE tidak mewarisi batasan kunci primer dan asing. Gaya distribusi, kunci pengurutan, BACKUP, dan properti NULL diwarisi oleh tabel LIKE, tetapi Anda tidak dapat secara eksplisit mengaturnya di CREATE TABLE... Pernyataan SEPERTI.

CADANGAN {YA | TIDAK}

Klausa yang menentukan apakah tabel harus disertakan dalam snapshot cluster otomatis dan manual.

Untuk tabel, seperti tabel pementasan, yang tidak berisi data penting, tentukan BACKUP NO untuk menghemat waktu pemrosesan saat membuat snapshot dan memulihkan dari snapshot serta mengurangi ruang penyimpanan di Amazon Simple Storage Service. Pengaturan BACKUP NO tidak berpengaruh pada replikasi otomatis data ke node lain di dalam cluster, sehingga tabel dengan BACKUP NO yang ditentukan dipulihkan jika terjadi kegagalan node. Defaultnya adalah BACKUP YES.

> **Note**
>
> No-backup tabel tidak didukung untuk cluster yang disediakan RG atau RA3 dan grup kerja Amazon Redshift Serverless. Tabel yang ditandai sebagai tanpa cadangan di cluster RG atau RA3 atau kelompok kerja tanpa server diperlakukan sebagai tabel permanen yang akan selalu dicadangkan saat mengambil snapshot, dan selalu dipulihkan saat memulihkan dari snapshot. Untuk menghindari biaya snapshot untuk tabel tanpa cadangan, potong sebelum mengambil snapshot.

DISTSTYLE {AUTO | GENAP | KUNCI | SEMUA}

Kata kunci yang mendefinisikan gaya distribusi data untuk seluruh tabel. Amazon Redshift mendistribusikan baris tabel ke node komputasi sesuai dengan gaya distribusi yang ditentukan untuk tabel. Defaultnya adalah AUTO.

Gaya distribusi yang Anda pilih untuk tabel memengaruhi kinerja keseluruhan database Anda. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri. Gaya distribusi yang mungkin adalah sebagai berikut:

- AUTO: Amazon Redshift menetapkan gaya distribusi optimal berdasarkan data tabel. Misalnya, jika gaya distribusi AUTO ditentukan, Amazon Redshift awalnya menetapkan gaya distribusi ALL ke tabel kecil. Saat tabel bertambah besar, Amazon Redshift mungkin mengubah gaya distribusi menjadi KEY, memilih kunci utama (atau kolom kunci primer komposit) sebagai DISTKEY. Jika tabel bertambah besar dan tidak ada kolom yang cocok untuk menjadi DISTKEY, Amazon Redshift mengubah gaya distribusi menjadi EVEN. Perubahan gaya distribusi terjadi di latar belakang dengan dampak minimal terhadap kueri pengguna.

Untuk melihat gaya distribusi yang diterapkan pada tabel, kueri tabel katalog sistem PG_CLASS. Untuk informasi selengkapnya, lihat Melihat gaya distribusi.

- EVEN: Data dalam tabel tersebar merata di seluruh node dalam cluster dalam distribusi round-robin. ID baris digunakan untuk menentukan distribusi, dan kira-kira jumlah baris yang sama didistribusikan ke setiap node.
- KUNCI: Data didistribusikan oleh nilai-nilai di kolom DISTKEY. Saat Anda mengatur kolom penggabungan tabel penggabungan sebagai kunci distribusi, baris penggabungan dari kedua tabel dikolokasikan pada node komputasi. Ketika data dikolokasikan, pengoptimal dapat melakukan gabungan dengan lebih efisien. Jika Anda menentukan DISTSTYLE KEY, Anda harus memberi nama kolom DISTKEY, baik untuk tabel atau sebagai bagian dari definisi kolom. Untuk informasi selengkapnya, lihat parameter DISTKEY sebelumnya dalam topik ini.
- SEMUA: Salinan seluruh tabel didistribusikan ke setiap node. Gaya distribusi ini memastikan bahwa semua baris yang diperlukan untuk gabungan apa pun tersedia di setiap node, tetapi ini melipatgandakan persyaratan penyimpanan dan meningkatkan waktu pemuatan dan pemeliharaan untuk tabel. Distribusi SEMUA dapat meningkatkan waktu eksekusi bila digunakan dengan tabel dimensi tertentu di mana distribusi KEY tidak sesuai, tetapi peningkatan kinerja harus ditimbang terhadap biaya pemeliharaan.

DISTKEY (nama_ kolom)

Batasan yang menentukan kolom yang akan digunakan sebagai kunci distribusi untuk tabel. Anda dapat menggunakan kata kunci DISTKEY setelah nama kolom atau sebagai bagian dari definisi tabel, dengan menggunakan sintaks DISTKEY (column _name). Salah satu metode memiliki efek yang sama. Untuk informasi selengkapnya, lihat parameter DISTSTYLE sebelumnya dalam topik ini.

[MAJEMUK | DISELINGI] SORTKEY (nama_kolom [,...]) | [SORTKEY OTOMATIS]

Menentukan satu atau lebih kunci pengurutan untuk tabel. Ketika data dimuat ke dalam tabel, data diurutkan berdasarkan kolom yang ditetapkan sebagai kunci pengurutan. Anda dapat menggunakan kata kunci SORTKEY setelah nama kolom untuk menentukan kunci pengurutan kolom tunggal, atau Anda dapat menentukan satu atau lebih kolom sebagai kolom kunci pengurutan untuk tabel dengan menggunakan sintaks. `SORTKEY (column_name [ , ... ] )`

Anda dapat secara opsional menentukan gaya pengurutan COMPOUND atau INTERLEAVED. Jika Anda menentukan SORTKEY dengan kolom defaultnya adalah COMPOUND. Untuk informasi selengkapnya, lihat Urutkan kunci.

Jika Anda tidak menentukan opsi kunci pengurutan apa pun, defaultnya adalah AUTO.

Anda dapat menentukan maksimum 400 kolom COMPOUND SORTKEY atau 8 kolom SORTKEY INTERLEAVED per tabel.

MOBIL

Menentukan bahwa Amazon Redshift menetapkan kunci pengurutan optimal berdasarkan data tabel. Misalnya, jika tombol pengurutan OTOMATIS ditentukan, Amazon Redshift awalnya tidak menetapkan kunci pengurutan ke tabel. Jika Amazon Redshift menentukan bahwa kunci pengurutan akan meningkatkan kinerja kueri, maka Amazon Redshift mungkin mengubah kunci pengurutan tabel Anda. Penyortiran tabel yang sebenarnya dilakukan dengan pengurutan tabel otomatis. Untuk informasi selengkapnya, lihat Pengurutan tabel otomatis.

Amazon Redshift tidak mengubah tabel yang memiliki kunci pengurutan atau distribusi yang ada. Dengan satu pengecualian, jika tabel memiliki kunci distribusi yang belum pernah digunakan dalam JOIN, maka kunci tersebut dapat diubah jika Amazon Redshift menentukan ada kunci yang lebih baik.

Untuk melihat kunci pengurutan tabel, kueri tampilan katalog sistem SVV_TABLE_INFO. Untuk informasi selengkapnya, lihat SVV_TABLE_INFO. Untuk melihat rekomendasi Amazon Redshift Advisor untuk tabel, kueri tampilan katalog sistem SVV_ALTER_TABLE_RECOMMENDATIONS. Untuk informasi selengkapnya, lihat SVV_ALTER_TABLE_REKOMENDASI. Untuk melihat tindakan yang dilakukan oleh Amazon Redshift, kueri tampilan katalog sistem SVL_AUTO_WORKER_ACTION. Untuk informasi selengkapnya, lihat SVL_AUTO_WORKER_ACTION.

SENYAWA

Menentukan bahwa data diurutkan menggunakan kunci majemuk yang terdiri dari semua kolom yang terdaftar, dalam urutan mereka terdaftar. Kunci pengurutan majemuk paling berguna ketika kueri memindai baris sesuai dengan urutan kolom pengurutan. Manfaat kinerja penyortiran dengan kunci majemuk menurun ketika kueri bergantung pada kolom pengurutan sekunder. Anda dapat menentukan maksimum 400 kolom COMPOUND SORTKEY per tabel.

TERJALIN

Menentukan bahwa data diurutkan menggunakan kunci pengurutan yang disisipkan. Maksimal delapan kolom dapat ditentukan untuk kunci pengurutan yang disisipkan.

Urutan yang disisipkan memberikan bobot yang sama untuk setiap kolom, atau subset kolom, dalam kunci pengurutan, sehingga kueri tidak bergantung pada urutan kolom dalam

kunci pengurutan. Ketika kueri menggunakan satu atau lebih kolom pengurutan sekunder, pengurutan terjalin secara signifikan meningkatkan kinerja kueri. Penyortiran interleaved membawa biaya overhead yang kecil untuk operasi pemuatan data dan penyedot debu.

> **Important**
>
> Jangan gunakan kunci pengurutan yang disisipkan pada kolom dengan atribut yang meningkat secara monoton, seperti kolom identitas, tanggal, atau stempel waktu.

MENYANDIKAN OTOMATIS

Mengaktifkan Amazon Redshift untuk secara otomatis menyesuaikan jenis pengkodean untuk semua kolom dalam tabel untuk mengoptimalkan kinerja kueri. ENCODE AUTO mempertahankan jenis encode awal yang Anda tentukan dalam membuat tabel. Kemudian, jika Amazon Redshift menentukan bahwa jenis pengkodean baru dapat meningkatkan kinerja kueri, Amazon Redshift dapat mengubah jenis pengkodean kolom tabel. ENCODE AUTO adalah default jika Anda tidak menentukan jenis pengkodean pada kolom mana pun dalam tabel.

UNIK (nama_kolom [,...])

Batasan yang menentukan bahwa sekelompok satu atau lebih kolom tabel hanya dapat berisi nilai-nilai unik. Perilaku kendala tabel unik sama dengan batasan kolom, dengan kemampuan tambahan untuk menjangkau beberapa kolom. Dalam konteks batasan unik, nilai nol tidak dianggap sama. Setiap batasan tabel unik harus memberi nama sekumpulan kolom yang berbeda dari kumpulan kolom yang dinamai oleh batasan kunci unik atau utama lainnya yang ditentukan untuk tabel.

> **Important**
>
> Kendala unik bersifat informasi dan tidak ditegakkan oleh sistem.

KUNCI UTAMA (column_name [,...])

Batasan yang menentukan bahwa kolom atau sejumlah kolom tabel hanya dapat berisi nilai unik (nonduplikat) non-null. Mengidentifikasi sekumpulan kolom sebagai kunci utama juga menyediakan metadata tentang desain skema. Kunci utama menyiratkan bahwa tabel lain dapat mengandalkan kumpulan kolom ini sebagai pengidentifikasi unik untuk baris. Satu kunci utama dapat ditentukan untuk tabel, baik sebagai batasan kolom tunggal atau batasan tabel. Kendala

kunci utama harus memberi nama sekumpulan kolom yang berbeda dari kumpulan kolom lain yang dinamai oleh batasan unik apa pun yang ditentukan untuk tabel yang sama.

> **Important**
>
> Kendala kunci utama hanya bersifat informasi. Mereka tidak ditegakkan oleh sistem, tetapi mereka digunakan oleh perencana.

KUNCI ASING (column_name [,...]) REFERENSI dapat di reftable [(refcolumn)]

Batasan yang menentukan batasan kunci asing, yang mengharuskan sekelompok satu atau lebih kolom tabel baru hanya berisi nilai-nilai yang cocok dengan nilai dalam kolom atau kolom yang direferensikan dari beberapa baris tabel yang direferensikan. Jika refcolumn dihilangkan, kunci utama reftable digunakan. Kolom yang direferensikan harus berupa kolom dari batasan kunci unik atau primer dalam tabel yang direferensikan.

> **Important**
>
> Kendala kunci asing hanya bersifat informasi. Mereka tidak ditegakkan oleh sistem, tetapi mereka digunakan oleh perencana.

Catatan penggunaan

Kendala keunikan, kunci utama, dan kunci asing hanya bersifat informasi; batasan tersebut tidak diberlakukan oleh Amazon Redshift saat Anda mengisi tabel. Misalnya, jika Anda memasukkan data ke dalam tabel dengan dependensi, penyisipan dapat berhasil meskipun melanggar batasan. Meskipun demikian, kunci utama dan kunci asing digunakan sebagai petunjuk perencanaan dan mereka harus dideklarasikan jika proses ETL Anda atau proses lain dalam aplikasi Anda menegakkan integritasnya. Untuk informasi tentang cara menjatuhkan tabel dengan dependensi, lihatMEJA DROP.

Batas dan kuota

Pertimbangkan batasan berikut saat Anda membuat tabel.

- Ada batas untuk jumlah maksimum tabel dalam cluster berdasarkan jenis node. Untuk informasi selengkapnya, lihat B atas di Panduan Manajemen Amazon Redshift.

- Jumlah maksimum karakter untuk nama tabel adalah 127.
- Jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.600.
- Jumlah maksimum kolom SORTKEY yang dapat Anda tentukan dalam satu tabel adalah 400.

Ringkasan pengaturan tingkat kolom dan pengaturan tingkat tabel

Beberapa atribut dan pengaturan dapat diatur pada tingkat kolom atau di tingkat tabel. Dalam beberapa kasus, menetapkan atribut atau batasan pada tingkat kolom atau di tingkat tabel memiliki efek yang sama. Dalam kasus lain, mereka menghasilkan hasil yang berbeda.

Daftar berikut merangkum pengaturan tingkat kolom dan tingkat tabel:

DISTKEY

Tidak ada perbedaan efek apakah diatur pada tingkat kolom atau di tingkat tabel.

Jika DISTKEY disetel, baik di tingkat kolom atau di tingkat tabel, DISTSTYLE harus disetel ke KEY atau tidak disetel sama sekali. DISTSTYLE hanya dapat diatur pada tingkat tabel.

SORTKEY

Jika diatur pada tingkat kolom, SORTKEY harus berupa kolom tunggal. Jika SORTKEY diatur pada tingkat tabel, satu atau lebih kolom dapat membentuk kunci pengurutan komposit majemuk atau disisipkan.

SUSUN CASE_SENSITIVE | SUSUN CASE_INSENSITIVE

Amazon Redshift tidak mendukung perubahan konfigurasi sensitivitas huruf besar/kecil untuk kolom. Saat Anda menambahkan kolom baru ke tabel, Amazon Redshift menggunakan nilai default untuk sensitivitas huruf besar/kecil. Amazon Redshift tidak mendukung kata kunci COLLATE saat menambahkan kolom baru.

Untuk informasi tentang cara membuat database menggunakan pengumpulan basis data, lihatBUAT BASIS DATA.

Untuk informasi tentang fungsi COLLATE, lihatFungsi COLLATE.

UNIK

Pada tingkat kolom, satu atau lebih kunci dapat diatur ke UNIQUE; batasan UNIQUE berlaku untuk setiap kolom secara individual. Jika UNIQUE diatur pada tingkat tabel, satu atau lebih kolom dapat membentuk batasan UNIK komposit.

KUNCI UTAMA

Jika diatur pada tingkat kolom, KUNCI UTAMA harus berupa satu kolom. Jika PRIMARY KEY diatur pada tingkat tabel, satu atau lebih kolom dapat membentuk kunci primer komposit.

KUNCI ASING

Tidak ada perbedaan efek apakah KUNCI ASING diatur pada tingkat kolom atau di tingkat tabel. Pada tingkat kolom, sintaks hanya dapat dire `REFERENCES visi [(refcolumn)]`.

Distribusi data yang masuk

Ketika skema distribusi hash dari data yang masuk cocok dengan tabel target, tidak ada distribusi fisik data yang benar-benar diperlukan ketika data dimuat. Misalnya, jika kunci distribusi ditetapkan untuk tabel baru dan data dimasukkan dari tabel lain yang didistribusikan pada kolom kunci yang sama, data dimuat di tempatnya, menggunakan node dan irisan yang sama. Namun, jika tabel sumber dan target keduanya diatur ke distribusi EVEN, data didistribusikan kembali ke tabel target.

Meja lebar

Anda mungkin dapat membuat tabel yang sangat lebar tetapi tidak dapat melakukan pemrosesan kueri, seperti pernyataan INSERT atau SELECT, pada tabel. Lebar maksimum tabel dengan kolom lebar tetap, seperti CHAR, adalah 64KB - 1 (atau 65535 byte). Jika tabel menyertakan kolom VARCHAR, tabel dapat memiliki lebar yang dideklarasikan lebih besar tanpa mengembalikan kesalahan karena kolom VARCHARS tidak menyumbangkan lebar penuh yang dideklarasikan ke batas pemrosesan kueri yang dihitung. Batas pemrosesan kueri yang efektif dengan kolom VARCHAR akan bervariasi berdasarkan sejumlah faktor.

Jika tabel terlalu lebar untuk dimasukkan atau dipilih, Anda menerima kesalahan berikut.

```
ERROR:  8001
DETAIL:  The combined length of columns processed in the SQL statement
exceeded the query-processing limit of 65535 characters (pid:7627)
```

Contoh

Untuk contoh yang menunjukkan cara menggunakan perintah CREATE TABLE, lihat Contoh topik.

Contoh

Contoh berikut menunjukkan berbagai atribut kolom dan tabel dalam pernyataan Amazon Redshift CREATE TABLE. Untuk informasi selengkapnya tentang CREATE TABLE, termasuk definisi parameter, lihatCREATE TABLE.

Banyak contoh menggunakan tabel dan data dari kumpulan data sampel TICKIT. Untuk informasi selengkapnya, lihat Contoh database.

Anda dapat mengawali nama tabel dengan nama database dan nama skema dalam perintah CREATE TABLE. Misalnya,`dev_database.public.sales`. Nama database harus database yang Anda sambungkan. Setiap upaya untuk membuat objek database di database lain gagal dengan kesalahan operasi yang tidak valid.

Buat tabel dengan kunci distribusi, kunci pengurutan majemuk, dan kompresi

Contoh berikut membuat tabel SALES di database TICKIT dengan kompresi yang ditentukan untuk beberapa kolom. LISTID dideklarasikan sebagai kunci distribusi, dan LISTID dan SELLERID dideklarasikan sebagai kunci pengurutan majemuk multicolumn. Kendala kunci primer dan kunci asing juga ditentukan untuk tabel. Sebelum membuat tabel dalam contoh, Anda mungkin perlu menambahkan batasan UNIK ke setiap kolom yang direferensikan oleh kunci asing, jika batasan tidak ada.

```
create table sales(
salesid integer not null,
listid integer not null,
sellerid integer not null,
buyerid integer not null,
eventid integer not null encode mostly16,
dateid smallint not null,
qtysold smallint not null encode mostly8,
pricepaid decimal(8,2) encode delta32k,
commission decimal(8,2) encode delta32k,
saletime timestamp,
primary key(salesid),
foreign key(listid) references listing(listid),
foreign key(sellerid) references users(userid),
foreign key(buyerid) references users(userid),
foreign key(dateid) references date(dateid))
distkey(listid)
compound sortkey(listid,sellerid);
```

Hasilnya berikut:

```
schemaname | tablename | column     | type                          | encoding | distkey
 | sortkey | notnull
-----------+-----------+------------+-------------------------------+----------+---------
+---------+--------
public     | sales     | salesid    | integer                       | lzo      | false
 |       0 | true
public     | sales     | listid     | integer                       | none     | true
 |       1 | true
public     | sales     | sellerid   | integer                       | none     | false
 |       2 | true
public     | sales     | buyerid    | integer                       | lzo      | false
 |       0 | true
public     | sales     | eventid    | integer                       | mostly16 | false
 |       0 | true
public     | sales     | dateid     | smallint                      | lzo      | false
 |       0 | true
public     | sales     | qtysold    | smallint                      | mostly8  | false
 |       0 | true
public     | sales     | pricepaid  | numeric(8,2)                  | delta32k | false
 |       0 | false
public     | sales     | commission | numeric(8,2)                  | delta32k | false
 |       0 | false
public     | sales     | saletime   | timestamp without time zone | lzo      | false
 |       0 | false
```

Contoh berikut membuat tabel t1 dengan kolom col1 yang tidak peka huruf besar/kecil.

```
create table T1 (
  col1 Varchar(20) collate case_insensitive
 );

insert into T1 values ('bob'), ('john'), ('Tom'), ('JOHN'), ('Bob');
```

Kueri tabel:

```
select * from T1 where col1 = 'John';

col1
------
 john
```

```
 JOHN
(2 rows)
```

Membuat tabel menggunakan kunci pengurutan yang disisipkan

Contoh berikut membuat tabel CUSTOMER dengan kunci pengurutan yang disisipkan.

```
create table customer_interleaved (
  c_custkey     integer        not null,
  c_name        varchar(25)    not null,
  c_address     varchar(25)    not null,
  c_city        varchar(10)    not null,
  c_nation      varchar(15)    not null,
  c_region      varchar(12)    not null,
  c_phone       varchar(15)    not null,
  c_mktsegment      varchar(10)    not null)
diststyle all
interleaved sortkey (c_custkey, c_city, c_mktsegment);
```

Buat tabel menggunakan IF NOT EXISTS

Contoh berikut membuat tabel CITIES, atau tidak melakukan apa-apa dan mengembalikan pesan jika sudah ada:

```
create table if not exists cities(
cityid integer not null,
city varchar(100) not null,
state char(2) not null);
```

Buat tabel dengan SEMUA distribusi

Contoh berikut membuat tabel VENUE dengan distribusi SEMUA.

```
create table venue(
venueid smallint not null,
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer,
primary key(venueid))
diststyle all;
```

Buat Tabel dengan distribusi EVEN

Contoh berikut membuat tabel bernama MYEVENT dengan tiga kolom.

```
create table myevent(
eventid int,
eventname varchar(200),
eventcity varchar(30))
diststyle even;
```

Tabel didistribusikan secara merata dan tidak diurutkan. Tabel tidak memiliki kolom DISTKEY atau SORTKEY yang dideklarasikan.

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'myevent';

  column   |          type          | encoding | distkey | sortkey
-----------+------------------------+----------+---------+---------
 eventid   | integer                | lzo      | f       |       0
 eventname | character varying(200) | lzo      | f       |       0
 eventcity | character varying(30)  | lzo      | f       |       0
(3 rows)
```

Buat tabel sementara yang SEPERTI tabel lain

Contoh berikut membuat tabel sementara yang disebut TEMPEVENT, yang mewarisi kolom dari tabel EVENT.

```
create temp table tempevent(like event);
```

Tabel ini juga mewarisi atribut DISTKEY dan SORTKEY dari tabel induknya:

```
select "column", type, encoding, distkey, sortkey
 from pg_table_def where tablename = 'tempevent';

  column   |            type             | encoding | distkey | sortkey
-----------+-----------------------------+----------+---------+---------
 eventid   | integer                     | none     | t       |       1
 venueid   | smallint                    | none     | f       |       0
 catid     | smallint                    | none     | f       |       0
 dateid    | smallint                    | none     | f       |       0
 eventname | character varying(200)      | lzo      | f       |       0
```

```
 starttime | timestamp without time zone | bytedict | f       |       0
(6 rows)
```

Buat tabel dengan kolom IDENTITAS

Contoh berikut membuat tabel bernama VENUE_IDENT, yang memiliki kolom IDENTITY bernama VENUEID. Kolom ini dimulai dengan 0 dan bertambah 1 untuk setiap catatan. VENUEID juga dinyatakan sebagai kunci utama tabel.

```
create table venue_ident(venueid bigint identity(0, 1),
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer,
primary key(venueid));
```

Buat tabel dengan kolom IDENTITAS default

Contoh berikut membuat tabel bernama`t1`. Tabel ini memiliki kolom IDENTITY bernama `hist_id` dan kolom IDENTITY default bernama`base_id`.

```
CREATE TABLE t1(
  hist_id BIGINT IDENTITY NOT NULL, /* Cannot be overridden */
  base_id BIGINT GENERATED BY DEFAULT AS IDENTITY NOT NULL, /* Can be overridden */
  business_key varchar(10) ,
  some_field varchar(10)
);
```

Memasukkan baris ke dalam tabel menunjukkan bahwa keduanya `hist_id` dan `base_id` nilai dihasilkan.

```
INSERT INTO T1 (business_key, some_field) values ('A','MM');
```

```
SELECT * FROM t1;

 hist_id | base_id | business_key | some_field
---------+---------+--------------+------------
       1 |       1 | A            | MM
```

Memasukkan baris kedua menunjukkan bahwa nilai default untuk `base_id` dihasilkan.

```
INSERT INTO T1 (base_id, business_key, some_field) values (DEFAULT, 'B','MNOP');
```

```
SELECT * FROM t1;

 hist_id | base_id | business_key | some_field
---------+---------+--------------+------------
       1 |       1 | A            | MM
       2 |       2 | B            | MNOP
```

Memasukkan baris ketiga menunjukkan bahwa nilai untuk `base_id` tidak perlu unik.

```
INSERT INTO T1 (base_id, business_key, some_field) values (2,'B','MNNN');
```

```
SELECT * FROM t1;

 hist_id | base_id | business_key | some_field
---------+---------+--------------+------------
       1 |       1 | A            | MM
       2 |       2 | B            | MNOP
       3 |       2 | B            | MNNN
```

Buat tabel dengan nilai kolom DEFAULT

Contoh berikut membuat tabel CATEGORYDEF yang menyatakan nilai default untuk setiap kolom:

```
create table categorydef(
catid smallint not null default 0,
catgroup varchar(10) default 'Special',
catname varchar(10) default 'Other',
catdesc varchar(50) default 'Special events',
primary key(catid));

insert into categorydef values(default,default,default,default);
```

```
select * from categorydef;

 catid | catgroup | catname |    catdesc
-------+----------+---------+----------------
     0 | Special  | Other   | Special events
(1 row)
```

Opsi DISTSTYLE, DISTKEY, dan SORTKEY

Contoh berikut menunjukkan cara kerja opsi DISTKEY, SORTKEY, dan DISTSTYLE. Dalam contoh ini, COL1 adalah kunci distribusi; oleh karena itu, gaya distribusi harus diatur ke KEY atau tidak disetel. Secara default, tabel tidak memiliki kunci pengurutan sehingga tidak diurutkan:

```
create table t1(col1 int distkey, col2 int) diststyle key;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't1';

column |  type   | encoding | distkey | sortkey
-------+---------+----------+---------+---------
col1   | integer | az64     | t       | 0
col2   | integer | az64     | f       | 0
```

Dalam contoh berikut, kolom yang sama didefinisikan sebagai kunci distribusi dan kunci pengurutan. Sekali lagi, gaya distribusi harus diatur ke KEY atau tidak disetel.

```
create table t2(col1 int distkey sortkey, col2 int);
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't2';

column |  type   | encoding | distkey | sortkey
-------+---------+----------+---------+---------
col1   | integer | none     | t       | 1
col2   | integer | az64     | f       | 0
```

Dalam contoh berikut, tidak ada kolom yang ditetapkan sebagai kunci distribusi, COL2 disetel sebagai kunci pengurutan, dan gaya distribusi diatur ke SEMUA:

```
create table t3(col1 int, col2 int sortkey) diststyle all;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't3';

Column |  Type   | Encoding | DistKey | SortKey
-------+---------+----------+---------+--------
col1   | integer | az64     | f       | 0
```

```
col2   | integer | none     | f       | 1
```

Dalam contoh berikut, gaya distribusi diatur ke EVEN dan tidak ada kunci pengurutan yang didefinisikan secara eksplisit; oleh karena itu tabel didistribusikan secara merata tetapi tidak diurutkan.

```
create table t4(col1 int, col2 int) diststyle even;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't4';

             column |  type   |encoding | distkey | sortkey
--------+---------+---------+---------+--------
col1    | integer | az64    | f       | 0
col2    | integer | az64    | f       | 0
```

Buat tabel dengan opsi ENCODE AUTO

Contoh berikut membuat tabel `t1` dengan pengkodean kompresi otomatis. ENCODE AUTO adalah default untuk tabel ketika Anda tidak menentukan jenis pengkodean untuk kolom apa pun.

```
create table t1(c0 int, c1 varchar);
```

Contoh berikut membuat tabel `t2` dengan pengkodean kompresi otomatis dengan menentukan ENCODE AUTO.

```
create table t2(c0 int, c1 varchar) encode auto;
```

Contoh berikut membuat tabel `t3` dengan pengkodean kompresi otomatis dengan menentukan ENCODE AUTO. Kol `c0` om didefinisikan dengan tipe pengkodean awal DELTA. Amazon Redshift dapat mengubah pengkodean jika pengkodean lain memberikan kinerja kueri yang lebih baik.

```
create table t3(c0 int encode delta, c1 varchar) encode auto;
```

Contoh berikut membuat tabel `t4` dengan pengkodean kompresi otomatis dengan menentukan ENCODE AUTO. Kol `c0` om didefinisikan dengan pengkodean awal DELTA, dan kolom `c1` didefinisikan dengan pengkodean awal LZO. Amazon Redshift dapat mengubah pengkodean ini jika pengkodean lain memberikan kinerja kueri yang lebih baik.

```
create table t4(c0 int encode delta, c1 varchar encode lzo) encode auto;
```

CREATE TABLE AS

Topik

- Sintaksis
- Parameter
- Catatan penggunaan CTAS
- Contoh CTAS

Membuat tabel baru berdasarkan query. Pemilik tabel ini adalah pengguna yang mengeluarkan perintah.

Tabel baru dimuat dengan data yang ditentukan oleh query dalam perintah. Kolom tabel memiliki nama dan tipe data yang terkait dengan kolom keluaran kueri. Perintah CREATE TABLE AS (CTAS) membuat tabel baru dan mengevaluasi kueri untuk memuat tabel baru.

Sintaksis

```
CREATE [ [ LOCAL ] { TEMPORARY | TEMP } ]
TABLE table_name
[ ( column_name [, ... ] ) ]
[ BACKUP { YES | NO } ]
[ table_attributes ]
AS query

where table_attributes are:
[ DISTSTYLE { AUTO | EVEN | ALL | KEY } ]
[ DISTKEY( distkey_identifier ) ]
[ [ COMPOUND | INTERLEAVED ] SORTKEY( column_name [, ...] ) ]
```

Parameter

LOKAL

Meskipun kata kunci opsional ini diterima dalam pernyataan, kata kunci ini tidak berpengaruh di Amazon Redshift.

SEMENTARA | TEMP

Membuat tabel sementara. Tabel sementara secara otomatis dijatuhkan pada akhir sesi di mana ia dibuat.

table_name

Nama tabel untuk dibuat.

> **Important**
>
> Jika Anda menentukan nama tabel yang dimulai dengan '# ', tabel dibuat sebagai tabel sementara. Contoh:
>
> ```
> create table #newtable (id) as select * from oldtable;
> ```

Panjang nama tabel maksimum adalah 127 byte; nama yang lebih panjang dipotong menjadi 127 byte. Amazon Redshift memberlakukan kuota jumlah tabel per cluster berdasarkan jenis node. Nama tabel dapat memenuhi syarat dengan database dan nama skema, seperti yang ditunjukkan tabel berikut.

```
create table tickit.public.test (c1) as select * from oldtable;
```

Dalam contoh ini, `tickit` adalah nama database dan `public` merupakan nama skema. Jika database atau skema tidak ada, pernyataan mengembalikan kesalahan.

Jika nama skema diberikan, tabel baru dibuat dalam skema itu (dengan asumsi pembuat memiliki akses ke skema). Nama tabel harus berupa nama unik untuk skema itu. Jika tidak ada skema yang ditentukan, tabel dibuat menggunakan skema database saat ini. Jika Anda membuat tabel sementara, Anda tidak dapat menentukan nama skema, karena tabel sementara ada dalam skema khusus.

Beberapa tabel sementara dengan nama yang sama diizinkan untuk ada pada waktu yang sama dalam database yang sama jika mereka dibuat dalam sesi terpisah. Tabel ini ditugaskan ke skema yang berbeda.

nama_kolom

Nama kolom dalam tabel baru. Jika tidak ada nama kolom yang disediakan, nama kolom diambil dari nama kolom keluaran kueri. Nama kolom default digunakan untuk ekspresi. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

CADANGAN {YA | TIDAK}

Klausa yang menentukan apakah tabel harus disertakan dalam snapshot cluster otomatis dan manual.

Untuk tabel, seperti tabel pementasan, yang tidak berisi data penting, tentukan BACKUP NO untuk menghemat waktu pemrosesan saat membuat snapshot dan memulihkan dari snapshot serta mengurangi ruang penyimpanan di Amazon Simple Storage Service. Pengaturan BACKUP NO tidak berpengaruh pada replikasi otomatis data ke node lain di dalam cluster, sehingga tabel dengan BACKUP NO yang ditentukan dipulihkan jika terjadi kegagalan node. Standarnya adalah BACKUP YES.

> **Note**
>
> No-backup tabel tidak didukung untuk cluster yang disediakan RG atau RA3 dan grup kerja Amazon Redshift Serverless. Tabel yang ditandai sebagai tanpa cadangan di cluster RG atau RA3 atau kelompok kerja tanpa server diperlakukan sebagai tabel permanen yang akan selalu dicadangkan saat mengambil snapshot, dan selalu dipulihkan saat memulihkan dari snapshot. Untuk menghindari biaya snapshot untuk tabel tanpa cadangan, potong sebelum mengambil snapshot.

DISTSTYLE {AUTO | GENAP | KUNCI | SEMUA}

Kata kunci yang mendefinisikan gaya distribusi data untuk seluruh tabel. Amazon Redshift mendistribusikan baris tabel ke node komputasi sesuai dengan gaya distribusi yang ditentukan untuk tabel. Standarnya adalah DISTSTYLE AUTO.

Gaya distribusi yang Anda pilih untuk tabel memengaruhi kinerja keseluruhan database Anda. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

- AUTO: Amazon Redshift menetapkan gaya distribusi optimal berdasarkan data tabel. Untuk melihat gaya distribusi yang diterapkan pada tabel, kueri tabel katalog sistem PG_CLASS. Untuk informasi selengkapnya, lihat Melihat gaya distribusi.

- EVEN: Data dalam tabel tersebar merata di seluruh node dalam cluster dalam distribusi round-robin. ID baris digunakan untuk menentukan distribusi, dan kira-kira jumlah baris yang sama didistribusikan ke setiap node. Ini adalah metode distribusi default.
- KUNCI: Data didistribusikan oleh nilai-nilai di kolom DISTKEY. Saat Anda mengatur kolom penggabungan tabel penggabungan sebagai kunci distribusi, baris penggabungan dari kedua tabel dikolokasikan pada node komputasi. Ketika data dikolokasikan, pengoptimal dapat melakukan gabungan dengan lebih efisien. Jika Anda menentukan DISTSTYLE KEY, Anda harus memberi nama kolom DISTKEY.
- SEMUA: Salinan seluruh tabel didistribusikan ke setiap node. Gaya distribusi ini memastikan bahwa semua baris yang diperlukan untuk gabungan apa pun tersedia di setiap node, tetapi ini melipatgandakan persyaratan penyimpanan dan meningkatkan waktu pemuatan dan pemeliharaan untuk tabel. Distribusi SEMUA dapat meningkatkan waktu eksekusi bila digunakan dengan tabel dimensi tertentu di mana distribusi KEY tidak sesuai, tetapi peningkatan kinerja harus ditimbang terhadap biaya pemeliharaan.

DISTKEY (kolom)

Menentukan nama kolom atau nomor posisi untuk kunci distribusi. Gunakan nama yang ditentukan dalam daftar kolom opsional untuk tabel atau daftar pilih kueri. Atau, gunakan nomor posisi, di mana kolom pertama yang dipilih adalah 1, yang kedua adalah 2, dan seterusnya. Hanya satu kolom dalam tabel yang dapat menjadi kunci distribusi:

- Jika Anda mendeklarasikan kolom sebagai kolom DISTKEY, DISTSTYLE harus disetel ke KEY atau tidak disetel sama sekali.
- Jika Anda tidak mendeklarasikan kolom DISTKEY, Anda dapat mengatur DISTSTYLE ke EVEN.
- Jika Anda tidak menentukan DISTKEY atau DISTSTYLE, CTAS menentukan gaya distribusi untuk tabel baru berdasarkan rencana kueri untuk klausa SELECT. Untuk informasi selengkapnya, lihat Warisan atribut kolom dan tabel.

Anda dapat mendefinisikan kolom yang sama dengan kunci distribusi dan kunci pengurutan; pendekatan ini cenderung mempercepat penggabungan ketika kolom yang dimaksud adalah kolom penggabungan dalam kueri.

[SENYAWA | DISELINGI] SORTKEY (nama_kolom [,...])

Menentukan satu atau lebih kunci pengurutan untuk tabel. Ketika data dimuat ke dalam tabel, data diurutkan berdasarkan kolom yang ditetapkan sebagai kunci pengurutan.

Anda dapat secara opsional menentukan gaya pengurutan COMPOUND atau INTERLEAVED. Standarnya adalah COMPOUND. Untuk informasi selengkapnya, lihat Urutkan kunci.

Anda dapat menentukan maksimum 400 kolom COMPOUND SORTKEY atau 8 kolom SORTKEY INTERLEAVED per tabel.

Jika Anda tidak menentukan SORTKEY, CTAS menentukan kunci pengurutan untuk tabel baru berdasarkan rencana kueri untuk klausa SELECT. Untuk informasi selengkapnya, lihat Warisan atribut kolom dan tabel.

SENYAWA

Menentukan bahwa data diurutkan menggunakan kunci majemuk yang terdiri dari semua kolom yang terdaftar, dalam urutan tercantum. Kunci pengurutan majemuk paling berguna ketika kueri memindai baris sesuai dengan urutan kolom pengurutan. Manfaat kinerja penyortiran dengan kunci majemuk menurun ketika kueri bergantung pada kolom pengurutan sekunder. Anda dapat menentukan maksimum 400 kolom COMPOUND SORTKEY per tabel.

TERJALIN

Menentukan bahwa data diurutkan menggunakan kunci pengurutan yang disisipkan. Maksimal delapan kolom dapat ditentukan untuk kunci pengurutan yang disisipkan.

Urutan yang disisipkan memberikan bobot yang sama untuk setiap kolom, atau subset kolom, dalam kunci pengurutan, sehingga kueri tidak bergantung pada urutan kolom dalam kunci pengurutan. Ketika kueri menggunakan satu atau lebih kolom pengurutan sekunder, pengurutan yang disisipkan secara signifikan meningkatkan kinerja kueri. Penyortiran interleaved membawa biaya overhead yang kecil untuk operasi pemuatan data dan penyedot debu.

Ku eri AS

Setiap kueri (pernyataan SELECT) yang didukung Amazon Redshift.

Catatan penggunaan CTAS

Batas

Amazon Redshift memberlakukan kuota jumlah tabel per cluster berdasarkan jenis node.

Jumlah karakter maksimum untuk nama tabel adalah 127.

Jumlah maksimum kolom yang dapat Anda tentukan dalam satu tabel adalah 1.600.

Warisan atribut kolom dan tabel

Tabel CREATE TABLE AS (CTAS) tidak mewarisi batasan, kolom identitas, nilai kolom default, atau kunci utama dari tabel tempat pembuatannya.

Anda tidak dapat menentukan pengkodean kompresi kolom untuk tabel CTAS. Amazon Redshift secara otomatis menetapkan pengkodean kompresi sebagai berikut:

- Kolom yang didefinisikan sebagai kunci pengurutan diberi kompresi RAW.
- Kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, DOUBLE PRECISION, GEOMETRY, atau GEOGRAPHY diberi kompresi RAW.
- Kolom yang didefinisikan sebagai SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, atau TIMESTAMPTZ diberi kompresi AZ64.
- Kolom yang didefinisikan sebagai CHAR, VARCHAR, atau VARBYTE diberi kompresi LZO.

Untuk informasi selengkapnya, lihat Pengkodean kompresi dan Jenis Data.

Untuk menetapkan pengkodean kolom secara eksplisit, gunakanCREATE TABLE.

CTAS menentukan gaya distribusi dan kunci pengurutan untuk tabel baru berdasarkan rencana kueri untuk klausa SELECT.

Untuk kueri kompleks, seperti kueri yang menyertakan gabungan, agregasi, urutan berdasarkan klausa, atau klausa batas, CTAS melakukan upaya terbaik untuk memilih gaya distribusi optimal dan mengurutkan kunci berdasarkan rencana kueri.

> **Note**
>
> Untuk kinerja terbaik dengan kumpulan data besar atau kueri kompleks, kami sarankan pengujian menggunakan kumpulan data khas.

Anda sering dapat memprediksi kunci distribusi dan kunci pengurutan yang dipilih CTAS dengan memeriksa rencana kueri untuk melihat kolom mana, jika ada, yang dipilih pengoptimal kueri untuk menyortir dan mendistribusikan data. Jika node teratas dari rencana kueri adalah pemindaian berurutan sederhana dari satu tabel (XN Seq Scan), maka CTAS umumnya menggunakan gaya distribusi tabel sumber dan kunci pengurutan. Jika node teratas dari rencana kueri adalah pemindaian berurutan lainnya (seperti Batas XN, XN Sort, XN HashAggregate, dan sebagainya),

CTAS melakukan upaya terbaik untuk memilih gaya distribusi optimal dan kunci pengurutan berdasarkan rencana kueri.

Misalnya, Anda membuat lima tabel menggunakan jenis klausa SELECT berikut:

- Pernyataan pilihan sederhana
- Klausul batas
- Order by klausa menggunakan LISTID
- Order by Clause menggunakan QTYSOLD
- Fungsi agregat SUM dengan grup demi klausa.

Contoh berikut menunjukkan rencana kueri untuk setiap pernyataan CTAS.

```
explain create table sales1_simple as select listid, dateid, qtysold from sales;
                           QUERY PLAN
-----------------------------------------------------------------
 XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(1 row)


explain create table sales2_limit as select listid, dateid, qtysold from sales limit
 100;
                              QUERY PLAN
----------------------------------------------------------------------
 XN Limit  (cost=0.00..1.00 rows=100 width=8)
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(2 rows)


explain create table sales3_orderbylistid as select listid, dateid, qtysold from sales
 order by listid;
                               QUERY PLAN
-------------------------------------------------------------------------
 XN Sort  (cost=1000000016724.67..1000000017155.81 rows=172456 width=8)
   Sort Key: listid
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(3 rows)


explain create table sales4_orderbyqty as select listid, dateid, qtysold from sales
 order by qtysold;
```

```
                                QUERY PLAN
------------------------------------------------------------------------
 XN Sort  (cost=1000000016724.67..1000000017155.81 rows=172456 width=8)
   Sort Key: qtysold
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(3 rows)


explain create table sales5_groupby as select listid, dateid, sum(qtysold) from sales
 group by listid, dateid;
                              QUERY PLAN
---------------------------------------------------------------------
 XN HashAggregate  (cost=3017.98..3226.75 rows=83509 width=8)
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(2 rows)
```

Untuk melihat kunci distribusi dan kunci pengurutan untuk setiap tabel, kueri tabel katalog sistem PG_TABLE_DEF, seperti yang ditunjukkan berikut.

```
select * from pg_table_def where tablename like 'sales%';

      tablename       |   column   | distkey | sortkey
----------------------+------------+---------+---------
 sales                | salesid    | f       |       0
 sales                | listid     | t       |       0
 sales                | sellerid   | f       |       0
 sales                | buyerid    | f       |       0
 sales                | eventid    | f       |       0
 sales                | dateid     | f       |       1
 sales                | qtysold    | f       |       0
 sales                | pricepaid  | f       |       0
 sales                | commission | f       |       0
 sales                | saletime   | f       |       0
 sales1_simple        | listid     | t       |       0
 sales1_simple        | dateid     | f       |       1
 sales1_simple        | qtysold    | f       |       0
 sales2_limit         | listid     | f       |       0
 sales2_limit         | dateid     | f       |       0
 sales2_limit         | qtysold    | f       |       0
 sales3_orderbylistid | listid     | t       |       1
 sales3_orderbylistid | dateid     | f       |       0
 sales3_orderbylistid | qtysold    | f       |       0
 sales4_orderbyqty    | listid     | t       |       0
```

```
sales4_orderbyqty    | dateid     | f       |       0
sales4_orderbyqty    | qtysold    | f       |       1
sales5_groupby       | listid     | f       |       0
sales5_groupby       | dateid     | f       |       0
sales5_groupby       | sum        | f       |       0
```

Tabel berikut merangkum hasilnya. Untuk kesederhanaan, tabel ini menghilangkan detail biaya, baris, dan lebar dari rencana penjelasan.

Tabel	Pernyataan CTAS SELECT	Jelaskan simpul atas rencana	Kunci Dist	Urutkan kunci
S1_SEDERHANA	`select listid, dateid, qtysold from sales`	`XN Seq Scan on sales ...`	LIST	DATEID
S2_LIMIT	`select listid, dateid, qtysold from sales limit 100`	`XN Limit ...`	Tidak ada (BAHKAN)	Tidak ada
S3_ORDER_DENGAN_LISTID	`select listid, dateid, qtysold from sales order by listid`	`XN Sort ...` `Sort Key: listid`	LIST	LIST
S4_ORDER_MENURUT JUMLAH	`select listid, dateid, qtysold from sales order by qtysold`	`XN Sort ...` `Sort Key: qtysold`	LIST	JUMLAH TERJUAL
S5_GROUP_BY	`select listid, dateid, sum(qtysold) from sales group by listid, dateid`	`XN HashAggregate ...`	Tidak ada (BAHKAN)	Tidak ada

Anda dapat secara eksplisit menentukan gaya distribusi dan mengurutkan kunci dalam pernyataan CTAS. Misalnya, pernyataan berikut membuat tabel menggunakan distribusi EVEN dan menentukan SALESID sebagai kunci pengurutan.

```
create table sales_disteven
diststyle even
sortkey (salesid)
as
select eventid, venueid, dateid, eventname
from event;
```

Pengkodean kompresi

ENCODE AUTO digunakan sebagai default untuk tabel. Amazon Redshift secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel.

Distribusi data yang masuk

Ketika skema distribusi hash dari data yang masuk cocok dengan tabel target, tidak ada distribusi fisik data yang benar-benar diperlukan ketika data dimuat. Misalnya, jika kunci distribusi ditetapkan untuk tabel baru dan data dimasukkan dari tabel lain yang didistribusikan pada kolom kunci yang sama, data dimuat di tempatnya, menggunakan node dan irisan yang sama. Namun, jika tabel sumber dan target keduanya diatur ke distribusi EVEN, data didistribusikan kembali ke tabel target.

Operasi ANALISIS otomatis

Amazon Redshift secara otomatis menganalisis tabel yang Anda buat dengan perintah CTAS. Anda tidak perlu menjalankan perintah ANALYZE pada tabel ini ketika mereka pertama kali dibuat. Jika Anda memodifikasinya, Anda harus menganalisisnya dengan cara yang sama seperti tabel lainnya.

Contoh CTAS

Contoh berikut membuat tabel bernama EVENT_BACKUP untuk tabel EVENT:

```
create table event_backup as select * from event;
```

Tabel yang dihasilkan mewarisi kunci distribusi dan pengurutan dari tabel EVENT.

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'event_backup';
```

```
column    | type                        | encoding | distkey | sortkey
----------+-----------------------------+----------+---------+--------
catid     | smallint                    | none     | false   |       0
dateid    | smallint                    | none     | false   |       1
eventid   | integer                     | none     | true    |       0
eventname | character varying(200)      | none     | false   |       0
starttime | timestamp without time zone | none     | false   |       0
venueid   | smallint                    | none     | false   |       0
```

Perintah berikut membuat tabel baru yang disebut EVENTDISTSORT dengan memilih empat kolom dari tabel EVENT. Tabel baru didistribusikan oleh EVENTID dan diurutkan berdasarkan EVENTID dan DATEID:

```
create table eventdistsort
distkey (1)
sortkey (1,3)
as
select eventid, venueid, dateid, eventname
from event;
```

Hasilnya adalah sebagai berikut:

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdistsort';

column   |          type          | encoding | distkey | sortkey
---------+------------------------+----------+---------+-------
eventid   | integer               | none     | t       | 1
venueid   | smallint              | none     | f       | 0
dateid    | smallint              | none     | f       | 2
eventname | character varying(200)| none     | f       | 0
```

Anda dapat membuat tabel yang persis sama dengan menggunakan nama kolom untuk kunci distribusi dan pengurutan. Contoh:

```
create table eventdistsort1
distkey (eventid)
sortkey (eventid, dateid)
as
select eventid, venueid, dateid, eventname
```

```
from event;
```

Pernyataan berikut menerapkan distribusi genap ke tabel tetapi tidak mendefinisikan kunci pengurutan eksplisit.

```
create table eventdisteven
diststyle even
as
select eventid, venueid, dateid, eventname
from event;
```

Tabel tidak mewarisi kunci pengurutan dari tabel EVENT (EVENTID) karena distribusi EVEN ditentukan untuk tabel baru. Tabel baru tidak memiliki kunci pengurutan dan tidak ada kunci distribusi.

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdisteven';

column    |          type          | encoding | distkey | sortkey
----------+------------------------+----------+---------+---------
eventid   | integer                | none     | f       | 0
venueid   | smallint               | none     | f       | 0
dateid    | smallint               | none     | f       | 0
eventname | character varying(200) | none     | f       | 0
```

Pernyataan berikut menerapkan distribusi genap dan mendefinisikan kunci pengurutan:

```
create table eventdistevensort diststyle even sortkey (venueid)
as select eventid, venueid, dateid, eventname from event;
```

Tabel yang dihasilkan memiliki kunci pengurutan tetapi tidak ada kunci distribusi.

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdistevensort';

column    |          type          | encoding | distkey | sortkey
----------+------------------------+----------+---------+-------
eventid   | integer                | none     | f       | 0
venueid   | smallint               | none     | f       | 1
dateid    | smallint               | none     | f       | 0
eventname | character varying(200) | none     | f       | 0
```

Pernyataan berikut mendistribusikan ulang tabel EVENT pada kolom kunci yang berbeda dari data yang masuk, yang diurutkan pada kolom EVENTID, dan tidak mendefinisikan kolom SORTKEY; oleh karena itu tabel tidak diurutkan.

```
create table venuedistevent distkey(venueid)
as select * from event;
```

Hasilnya adalah sebagai berikut:

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'venuedistevent';

 column   |            type             | encoding | distkey | sortkey
----------+-----------------------------+----------+---------+-------
eventid   | integer                     | none     | f       | 0
venueid   | smallint                    | none     | t       | 0
catid     | smallint                    | none     | f       | 0
dateid    | smallint                    | none     | f       | 0
eventname | character varying(200)      | none     | f       | 0
starttime | timestamp without time zone | none     | f       | 0
```

BUAT TEMPLAT

Membuat template yang dapat digunakan kembali untuk perintah Amazon Redshift seperti MENYALIN. Template menyimpan parameter yang umum digunakan yang dapat direferensikan di beberapa eksekusi perintah, meningkatkan konsistensi dan mengurangi spesifikasi parameter manual.

Template menghilangkan kebutuhan untuk berulang kali menentukan parameter pemformatan yang sama di beberapa operasi, sementara jalur sumber, tabel target, dan otorisasi dapat bervariasi antar operasi.

Hak istimewa yang diperlukan

Untuk membuat template, Anda harus memiliki salah satu dari berikut ini:

- Hak istimewa pengguna super
- Izin BUAT pada skema tempat Anda ingin membuat template, atau BUAT izin cakupan pada skema di database tempat Anda ingin membuat template

Sintaksis

```
CREATE [ OR REPLACE ] TEMPLATE [database_name.][schema_name.]template_name
FOR COPY [ AS ]
[ [ FORMAT ] [ AS ] data_format ]
[ parameter [ argument ] [ , ... ] ];
```

Parameter

ATAU GANTI

Jika template dengan nama yang sama sudah ada di database dan skema yang ditentukan, template yang ada diganti. Anda hanya dapat mengganti template dengan template baru yang mendefinisikan jenis operasi yang sama, misalnya, COPY. Anda harus memiliki hak istimewa yang diperlukan untuk mengganti template.

database_name

(Opsional) Nama database tempat template akan dibuat. Jika tidak ditentukan, template dibuat dalam database saat ini.

Jika database atau skema tidak ada, template tidak dibuat, dan pernyataan mengembalikan kesalahan. Anda tidak dapat membuat templat di database sistem`template0`,`template1`,`padb_harvest`, atau`sys:internal`.

nama skema

(Opsional) Nama skema tempat template akan dibuat. Jika tidak ditentukan, template dibuat dalam skema saat ini.

Jika nama skema diberikan, template baru dibuat dalam skema itu (dengan asumsi pembuat memiliki akses ke skema). Nama template harus berupa nama unik untuk skema itu.

nama_templat

Nama template yang akan dibuat. Secara opsional, nama template dapat memenuhi syarat dengan database dan nama skema. Dalam contoh berikut, nama database adalah`demo_database`, nama skema adalah`demo_schema`, dan nama template adalah`test`. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

```
CREATE TEMPLATE demo_database.demo_schema.test FOR COPY AS CSV;
```

MENYALIN

Menentukan jenis perintah Redshift tempat template dibuat. Saat ini, hanya perintah COPY yang didukung.

[[FORMAT] [AS] data_ format]

Ini adalah parameter opsional. Ini menentukan format data untuk operasi COPY.

[parameter [argumen]]

Setiap parameter yang valid untuk perintah pergeseran merah yang ditentukan.

Misalnya, template untuk perintah COPY dapat mencakup:

- Parameter format data
- Parameter kompresi file
- Parameter konversi data
- Operasi beban data

Untuk daftar lengkap parameter yang didukung, lihat MENYALIN perintah.

Catatan penggunaan

- Secara default, semua pengguna memiliki hak istimewa CREATE dan USAGE pada skema PUBLIK. Untuk melarang pengguna membuat objek dalam skema PUBLIK database, gunakan perintah REVOKE untuk menghapus hak istimewa tersebut.
- Ketika parameter ada di template dan perintah, parameter perintah diutamakan.
- Template adalah objek database dan mengikuti aturan izin dan penamaan objek Redshift standar. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.
- Template tidak dapat berisi spesifikasi file manifes untuk MENYALIN perintah.

Batasan

- Setidaknya satu parameter harus ditentukan saat membuat template.
- Parameter yang dikecualikan — Command-specific parameter seperti jalur sumber, tabel target, kredentif otorisasi, dan spesifikasi file manifes tidak dapat dimasukkan dalam template. Parameter ini harus ditentukan dalam perintah yang sebenarnya.

- Template maksimum per cluster — Anda dapat membuat maksimal 1.000 templat per cluster. Batas ini berlaku untuk jumlah total template di semua database dan skema di cluster.
- Cross-database referensi — Template tidak dapat direferensikan di seluruh database.
- Berbagi data — Template tidak dapat dimasukkan dalam berbagi data. Template harus dibuat secara terpisah di setiap cluster di mana mereka dibutuhkan.

Contoh

Contoh berikut membuat template untuk perintah COPY

```
CREATE TEMPLATE test_schema.demo_template
FOR COPY
AS
FORMAT JSON 'auto'
NULL AS ''
MAXERROR 100;
```

Gunakan TAMPILKAN TEMPLATE untuk mendapatkan definisi template:

```
SHOW TEMPLATE test_schema.demo_template;
CREATE OR REPLACE TEMPLATE dev.test_schema.demo_template FOR COPY AS FORMAT AS JSON
 'auto' NULL '' MAXERROR 100;
```

Kueri SYS_REDSHIFT_TEMPLATE tampilan sistem untuk mendapatkan detail lebih lanjut tentang template.

```
SELECT * FROM SYS_REDSHIFT_TEMPLATE;

database_name | schema_name | template_name | template_type |        create_time
  |     last_modified_time     | owner_id | last_modified_by | template_parameters
---------------+-------------+---------------+---------------
+---------------------------+---------------------------+----------
+------------------+--------------------
 dev           | test_schema | demo_template |             1 | 2025-12-17
 20:06:01.944171 | 2025-12-17 20:06:01.944171 |        1 |               1 | {
    "JSON": "auto",
    "MAXERROR": 100,
    "NULL": ""
}
```

BUAT PENGGUNA

Membuat pengguna database baru. Pengguna database dapat mengambil data, menjalankan perintah, dan melakukan tindakan lain dalam database, tergantung pada hak istimewa dan peran mereka. Anda harus menjadi superuser database untuk menjalankan perintah ini.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk CREATE USER:

- Superuser
- Pengguna dengan hak istimewa CREATE USER

Sintaksis

```
CREATE USER name [ WITH ]
PASSWORD { 'password' | 'md5hash' | 'sha256hash' | DISABLE }
[ option [ ... ] ]

where option can be:

CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }
| IN GROUP groupname [, ... ]
| VALID UNTIL 'abstime'
| CONNECTION LIMIT { limit | UNLIMITED }
| SESSION TIMEOUT limit
| EXTERNALID external_id
```

Parameter

name

Nama pengguna yang akan dibuat. Nama pengguna tidak dapat `PUBLIC`. Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi.

DENGAN

Kata kunci opsional. WITH diabaikan oleh Amazon Redshift

KATA SANDI {'kata sandi' | 'md5hash' | 'sha256hash' | NONAKTIFKAN}

Menetapkan kata sandi pengguna.

Secara default, pengguna dapat mengubah kata sandi mereka sendiri, kecuali kata sandi dinonaktifkan. Untuk menonaktifkan kata sandi pengguna, tentukan NONAKTIFKAN. Ketika kata sandi pengguna dinonaktifkan, kata sandi dihapus dari sistem dan pengguna dapat masuk hanya menggunakan kredenSIAL pengguna sementara AWS Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat Menggunakan Otentikasi IAM untuk Menghasilkan KredenSIAL Pengguna Basis Data. Hanya pengguna super yang dapat mengaktifkan atau menonaktifkan kata sandi. Anda tidak dapat menonaktifkan kata sandi pengguna super. Untuk mengaktifkan kata sandi, jalankan ALTER USER dan tentukan kata sandi.

Anda dapat menentukan kata sandi dalam teks yang jelas, sebagai string hash MD5, atau sebagai string hash SHA256.

> **Note**
>
> Saat meluncurkan cluster baru menggunakan Konsol Manajemen AWS, AWS CLI, atau Amazon Redshift API, Anda harus memberikan kata sandi teks yang jelas untuk pengguna basis data awal. Anda dapat mengubah kata sandi nanti dengan menggunakanALTER USER.

Untuk teks yang jelas, kata sandi harus memenuhi batasan berikut:

- Panjangnya harus 8 hingga 64 karakter.
- Itu harus berisi setidaknya satu huruf besar, satu huruf kecil, dan satu angka.
- Itu dapat menggunakan karakter ASCII apa pun dengan kode ASCII 33-126, kecuali '(tanda kutip tunggal), "(tanda kutip ganda),\,/, atau @.

Sebagai alternatif yang lebih aman untuk meneruskan parameter kata sandi CREATE USER sebagai teks yang jelas, Anda dapat menentukan hash MD5 dari string yang menyertakan kata sandi dan nama pengguna.

> **Note**
>
> Ketika Anda menentukan string hash MD5, perintah CREATE USER memeriksa string hash MD5 yang valid, tetapi tidak memvalidasi bagian kata sandi dari string.

Dimungkinkan dalam hal ini untuk membuat kata sandi, seperti string kosong, yang tidak dapat Anda gunakan untuk masuk ke database.

Untuk menentukan kata sandi MD5, ikuti langkah-langkah berikut:

1. Gabungkan kata sandi dan nama pengguna.

   Misalnya, untuk kata sandi `ez` dan penggunauser1, string gabungan adalah. `ezuser1`

2. Ubah string gabungan menjadi string hash MD5 32 karakter. Anda dapat menggunakan utilitas MD5 apa pun untuk membuat string hash. Contoh berikut menggunakan Amazon Redshift Fungsi MD5 dan operator penggabungan (||) untuk mengembalikan string 32 karakter. MD5-hash

   ```
   select md5('ez' || 'user1');

   md5
   ----------------------------------
   153c434b4b77c89e6b94f12c5393af5b
   ```

3. Gabungkan 'md5' di depan string hash MD5 dan berikan string gabungan sebagai argumen md5hash.

   ```
   create user user1 password 'md5153c434b4b77c89e6b94f12c5393af5b';
   ```

4. Masuk ke database menggunakan kredenSIAL masuk.

   Untuk contoh ini, masuk seperti `user1` dengan kata sandi`ez`.

Alternatif aman lainnya adalah menentukan SHA-256 hash dari string kata sandi; atau Anda dapat memberikan intisari valid Anda sendiri SHA-256 dan garam 256-bit yang digunakan untuk membuat intisari.

- Digest — Output dari fungsi hashing.
- Salt — Data yang dihasilkan secara acak yang dikombinasikan dengan kata sandi untuk membantu mengurangi pola dalam output fungsi hashing.

```
'sha256|Mypassword'
```

```
'sha256|digest|256-bit-salt'
```

Dalam contoh berikut, Amazon Redshift menghasilkan dan mengelola garam.

```
CREATE USER admin PASSWORD 'sha256|Mypassword1';
```

Dalam contoh berikut, intisari SHA-256 yang valid dan garam 256-bit yang digunakan untuk membuat cerna disediakan.

Untuk menentukan kata sandi dan hash dengan garam Anda sendiri, ikuti langkah-langkah ini:

1. Buat garam 256-bit. Anda dapat memperoleh garam dengan menggunakan generator string heksadesimal apa pun untuk menghasilkan string sepanjang 64 karakter. Untuk contoh ini, garam adalah`c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6`.
2. Gunakan fungsi FROM_HEX untuk mengubah garam Anda menjadi biner. Ini karena fungsi SHA2 membutuhkan representasi biner dari garam. Lihat pernyataan berikut.

   ```
   SELECT
    FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6');
   ```
3. Gunakan fungsi CONCAT untuk menambahkan garam Anda ke kata sandi Anda. Untuk contoh ini, kata sandinya adalah`Mypassword1`. Lihat pernyataan berikut.

   ```
   SELECT
    CONCAT('Mypassword1',FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f951303
   ```
4. Gunakan fungsi SHA2 untuk membuat intisari dari kombinasi kata sandi dan garam Anda. Lihat pernyataan berikut.

   ```
   SELECT
    SHA2(CONCAT('Mypassword1',FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f95
    0);
   ```
5. Menggunakan pencernaan dan garam dari langkah-langkah sebelumnya, buat pengguna. Lihat pernyataan berikut.

   ```
   CREATE USER admin PASSWORD 'sha256|
   821708135fcc42eb3afda85286dee0ed15c2c461d000291609f77eb113073ec2|
   c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6';
   ```

6. Masuk ke database menggunakan kredenSIAL masuk.

   Untuk contoh ini, masuk seperti `admin` dengan kata sandi`Mypassword1`.

Jika Anda menetapkan kata sandi dalam teks biasa tanpa menentukan fungsi hashing, maka ringkasan MD5 dihasilkan menggunakan nama pengguna sebagai garam.

CREATEDB | NOCREATEDB

Opsi CREATEDB memungkinkan pengguna baru untuk membuat database. Defaultnya adalah NOCREATEDB.

CREATEUSER | NOCREATEUSER

Opsi CREATEUSER membuat superuser dengan semua hak database, termasuk CREATE USER. Defaultnya adalah NOCREATEUSER. Untuk informasi selengkapnya, lihat superuser.

AKSES SYSLOG {DIBATASI | TIDAK DIBATASI}

Klausa yang menentukan tingkat akses yang dimiliki pengguna ke tabel dan tampilan sistem Amazon Redshift.

Pengguna biasa yang memiliki izin SYSLOG ACCESS RESTRICTED hanya dapat melihat baris yang dihasilkan oleh pengguna tersebut dalam tabel dan tampilan sistem yang terlihat pengguna. Defaultnya adalah DIBATASI.

Pengguna biasa yang memiliki izin SYSLOG ACCESS UNLIMITED dapat melihat semua baris dalam tabel dan tampilan sistem yang terlihat pengguna, termasuk baris yang dihasilkan oleh pengguna lain. UNRESTRICTED tidak memberikan akses pengguna biasa ke tabel yang terlihat oleh pengguna super. Hanya pengguna super yang dapat melihat tabel yang terlihat oleh pengguna super.

> **Note**
>
> Memberikan pengguna akses tak terbatas ke tabel sistem memberikan visibilitas pengguna terhadap data yang dihasilkan oleh pengguna lain. Misalnya, STL_QUERY dan STL_QUERYTEXT berisi teks lengkap pernyataan INSERT, UPDATE, dan DELETE, yang mungkin berisi data sensitif buatan pengguna.

Semua baris di SVV_TRANSACTIONS terlihat oleh semua pengguna.

Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Nama grup DALAM GRUP

Menentukan nama grup yang ada milik pengguna. Beberapa nama grup dapat dicantumkan.

BERLAKU HINGGA ab sen

Opsi VALID UNTIL menetapkan waktu absolut setelah kata sandi pengguna tidak lagi valid. Secara default kata sandi tidak memiliki batas waktu.

BATAS KONEKSI {limit | TIDAK TERBATAS}

Jumlah maksimum koneksi database yang diizinkan untuk dibuka oleh pengguna secara bersamaan. Batas tidak diberlakukan untuk pengguna super. Gunakan kata kunci UNLIMITED untuk mengizinkan jumlah maksimum koneksi bersamaan. Batas jumlah koneksi untuk setiap database mungkin juga berlaku. Untuk informasi selengkapnya, lihat BUAT BASIS DATA. Defaultnya adalah UNLIMITED. Untuk melihat koneksi saat ini, kueri STV_SESSION tampilan sistem.

> **Note**
>
> Jika batas koneksi pengguna dan database berlaku, slot koneksi yang tidak digunakan harus tersedia yang berada dalam kedua batas saat pengguna mencoba menyambung.

Batas BATAS WAKTU SESI

Waktu maksimum dalam detik sesi tetap tidak aktif atau idle. Kisarannya adalah 60 detik (satu menit) hingga 1.728.000 detik (20 hari). Jika tidak ada batas waktu sesi yang ditetapkan untuk pengguna, pengaturan cluster berlaku. Untuk informasi selengkapnya, lihat Kuota dan batasan di Amazon Redshift di Panduan Manajemen Amazon Redshift.

Saat Anda mengatur batas waktu sesi, itu hanya diterapkan pada sesi baru.

Untuk melihat informasi tentang sesi pengguna aktif, termasuk waktu mulai, nama pengguna, dan batas waktu sesi, kueri tampilan STV_SESSION sistem. Untuk melihat informasi tentang riwayat sesi pengguna, kueri STL_SESSIONS tampilan. Untuk mengambil informasi tentang pengguna database, termasuk nilai batas waktu sesi, kueri tampilan. SVL_USER_INFO

ID eksternalid_eksternal

Pengidentifikasi untuk pengguna, yang terkait dengan penyedia identitas. Pengguna harus menonaktifkan kata sandi mereka. Untuk informasi selengkapnya, lihat Federasi penyedia identitas asli (IdP) untuk Amazon Red shift.

Catatan penggunaan

Secara default, semua pengguna memiliki hak istimewa CREATE dan USAGE pada skema PUBLIK. Untuk melarang pengguna membuat objek dalam skema PUBLIK database, gunakan perintah REVOKE untuk menghapus hak istimewa tersebut.

Saat menggunakan otentikasi IAM untuk membuat kredenSIAL pengguna database, Anda mungkin ingin membuat superuser yang dapat masuk hanya menggunakan kredenSIAL sementara. Anda tidak dapat menonaktifkan kata sandi superuser, tetapi Anda dapat membuat kata sandi yang tidak dikenal menggunakan string hash MD5 yang dibuat secara acak.

```
create user iam_superuser password 'md5A1234567890123456780123456789012' createuser;
```

Kasus nama pengguna yang terlampir dalam tanda kutip ganda selalu dipertahankan terlepas dari pengaturan opsi `enable_case_sensitive_identifier` konfigurasi. Untuk informasi selengkapnya, lihat enable_case_sensitive_identifier.

Contoh

Perintah berikut membuat pengguna bernama dbuser, dengan kata sandi “ABCD1234", hak pembuatan database, dan batas koneksi 30.

```
create user dbuser with password 'abcD1234' createdb connection limit 30;
```

Kueri tabel katalog PG_USER_INFO untuk melihat detail tentang pengguna database.

```
select * from pg_user_info;

 usename   | usesysid | usecreatedb | usesuper | usecatupd | passwd   | valuntil |
 useconfig | useconnlimit
-----------+----------+-------------+----------+-----------+----------+----------
+-----------+-------------
 rdsdb     |        1 | true        | true     | true      | ******** | infinity |
      |
 adminuser |      100 | true        | true     | false     | ******** |          |
      | UNLIMITED
 dbuser    |      102 | true        | false    | false     | ******** |          |
      | 30
```

Dalam contoh berikut, kata sandi akun berlaku hingga 10 Juni 2017.

```
create user dbuser with password 'abcD1234' valid until '2017-06-10';
```

Contoh berikut membuat pengguna dengan kata sandi peka huruf besar-kecil yang berisi karakter khusus.

```
create user newman with password '@AbC4321!';
```

Untuk menggunakan garis miring terbalik ('\') di kata sandi MD5 Anda, hindari garis miring terbalik dengan garis miring terbalik di string sumber Anda. Contoh berikut membuat pengguna bernama `slashpass` dengan garis miring terbalik tunggal ('\') sebagai kata sandi.

```
select md5('\\'||'slashpass');

md5
--------------------------------
0c983d1a624280812631c5389e60d48c
```

Buat pengguna dengan kata sandi md5.

```
create user slashpass password 'md50c983d1a624280812631c5389e60d48c';
```

Contoh berikut membuat pengguna bernama `dbuser` dengan batas waktu sesi idle-set menjadi 120 detik.

```
CREATE USER dbuser password 'abcD1234' SESSION TIMEOUT 120;
```

Contoh berikut membuat pengguna bernama bob. Namespace adalah`myco_aad`. Ini hanya sampel. Untuk menjalankan perintah dengan sukses, Anda harus memiliki penyedia identitas terdaftar. Untuk informasi selengkapnya, lihat Federasi penyedia identitas asli (IdP) untuk Amazon Red shift.

```
CREATE USER myco_aad:bob EXTERNALID "ABC123" PASSWORD DISABLE;
```

CREATE VIEW

Membuat tampilan dalam database. Tampilan tidak terwujud secara fisik; kueri yang mendefinisikan tampilan dijalankan setiap kali tampilan direferensikan dalam kueri. Untuk membuat tampilan dengan tabel eksternal, sertakan klausa WITH NO SCHEMA BINDING.

Untuk membuat tampilan standar, Anda memerlukan akses ke tabel yang mendasarinya, atau ke tampilan yang mendasarinya. Untuk menanyakan tampilan standar, Anda perlu memilih izin untuk tampilan itu sendiri, tetapi Anda tidak perlu memilih izin untuk tabel yang mendasarinya. Dalam kasus di mana Anda membuat tampilan yang mereferensikan tabel atau tampilan dalam skema lain, atau jika Anda membuat tampilan yang mereferensikan tampilan yang terwujud, Anda memerlukan izin penggunaan. Untuk menanyakan tampilan pengikatan terlambat, Anda perlu memilih izin untuk tampilan pengikatan terlambat itu sendiri. Anda juga harus memastikan pemilik tampilan pengikatan terlambat memiliki izin pilih untuk objek yang direferensikan (tabel, tampilan, atau fungsi yang ditentukan pengguna). Untuk informasi selengkapnya tentang tampilan pengikatan terlambat, lihat. Catatan penggunaan

Izin yang diperlukan

Untuk menggunakan CREATE VIEW, salah satu izin berikut diperlukan.

- Untuk membuat tampilan menggunakan CREATE [OR REPLACE] VIEW:
  - Superuser
  - Pengguna dengan izin CREATE [REPLACE] VIEW
- Untuk mengganti tampilan yang ada menggunakan CREATE OR REPLACE VIEW:
  - Superuser
  - Pengguna dengan izin CREATE [OR REPLACE] VIEW
  - Lihat pemilik

Jika pengguna ingin mengakses tampilan yang menggabungkan fungsi yang ditentukan pengguna, pengguna harus memiliki izin EXECUTE pada fungsi itu.

Sintaksis

```
CREATE [ OR REPLACE ] VIEW name [ ( column_name [, ...] ) ] AS query
[ WITH NO SCHEMA BINDING ]
```

Parameter

ATAU GANTI

Jika tampilan dengan nama yang sama sudah ada, tampilan diganti. Anda hanya dapat mengganti tampilan dengan kueri baru yang menghasilkan kumpulan kolom yang identik,

menggunakan nama kolom dan tipe data yang sama. CREATE OR REPLACE VIEW mengunci tampilan untuk membaca dan menulis hingga operasi selesai.

Saat tampilan diganti, properti lainnya seperti kepemilikan dan hak istimewa yang diberikan dipertahankan.

name

Nama tampilan. Jika nama skema diberikan (sepertimyschema.myview) tampilan dibuat menggunakan skema yang ditentukan. Jika tidak, tampilan dibuat dalam skema saat ini. Nama tampilan harus berbeda dari nama tampilan atau tabel lain dalam skema yang sama.

Jika Anda menentukan nama tampilan yang dimulai dengan '# ', tampilan dibuat sebagai tampilan sementara yang hanya terlihat di sesi saat ini.

Untuk informasi selengkapnya tentang nama yang valid, lihatNama dan pengidentifikasi. Anda tidak dapat membuat tabel atau tampilan di database sistem template0, template1, padb_harvest, atau sys:internal.

nama_kolom

Daftar nama opsional yang akan digunakan untuk kolom dalam tampilan. Jika tidak ada nama kolom yang diberikan, nama kolom diturunkan dari kueri. Jumlah maksimum kolom yang dapat Anda tentukan dalam satu tampilan adalah 1.600.

query

Kueri (dalam bentuk pernyataan SELECT) yang mengevaluasi ke tabel. Tabel ini mendefinisikan kolom dan baris dalam tampilan.

TANPA PENGIKATAN SKEMA

Klausa yang menentukan bahwa tampilan tidak terikat ke objek database yang mendasarinya, seperti tabel dan fungsi yang ditentukan pengguna. Akibatnya, tidak ada ketergantungan antara tampilan dan objek yang dirujukkannya. Anda dapat membuat tampilan meskipun objek yang direferensikan tidak ada. Karena tidak ada ketergantungan, Anda dapat melepaskan atau mengubah objek yang direferensikan tanpa mempengaruhi tampilan. Amazon Redshift tidak memeriksa dependensi sampai tampilan ditanyakan. Ekspresi tabel umum rekursif (RCTe) tidak didukung dengan tampilan pengikatan terlambat. Untuk melihat detail tentang tampilan pengikatan terlambat, jalankan fungsinya. PG_GET_LATE_BINDING_VIEW_COLS

Bila Anda menyertakan klausa WITH NO SCHEMA BINDING, tabel dan tampilan yang dirujuk dalam pernyataan SELECT harus memenuhi syarat dengan nama skema. Skema harus ada saat

tampilan dibuat, bahkan jika tabel yang direferensikan tidak ada. Misalnya, pernyataan berikut mengembalikan kesalahan.

```
create view myevent as select eventname from event
with no schema binding;
```

Pernyataan berikut berjalan dengan sukses.

```
create view myevent as select eventname from public.event
with no schema binding;
```

> **Note**
>
> Anda tidak dapat memperbarui, menyisipkan ke, atau menghapus dari tampilan.

Catatan penggunaan

Late-binding pandangan

Tampilan pengikatan terlambat tidak memeriksa objek database yang mendasarinya, seperti tabel dan tampilan lainnya, sampai tampilan ditanyakan. Akibatnya, Anda dapat mengubah atau melepaskan objek yang mendasarinya tanpa menjatuhkan dan membuat ulang tampilan. Jika Anda melepaskan objek yang mendasarinya, kueri ke tampilan pengikatan terlambat akan gagal. Jika kueri ke tampilan pengikatan terlambat mereferensikan kolom di objek yang mendasarinya yang tidak ada, kueri akan gagal.

Jika Anda melepas dan kemudian membuat ulang tabel atau tampilan dasar tampilan pengikatan terlambat, objek baru dibuat dengan izin akses default. Anda mungkin perlu memberikan izin ke objek yang mendasarinya untuk pengguna yang akan menanyakan tampilan.

Untuk membuat tampilan pengikatan terlambat, sertakan klausa WITH NO SCHEMA BINDING. Contoh berikut membuat tampilan tanpa pengikatan skema.

```
create view event_vw as select * from public.event
with no schema binding;
```

```
select * from event_vw limit 1;
```

```
eventid | venueid | catid | dateid | eventname     | starttime
--------+---------+-------+--------+---------------+--------------------
      2 |     306 |     8 |   2114 | Boris Godunov | 2008-10-15 20:00:00
```

Contoh berikut menunjukkan bahwa Anda dapat mengubah tabel yang mendasarinya tanpa membuat ulang tampilan.

```
alter table event rename column eventname to title;
```

```
select * from event_vw limit 1;

eventid | venueid | catid | dateid | title         | starttime
--------+---------+-------+--------+---------------+--------------------
      2 |     306 |     8 |   2114 | Boris Godunov | 2008-10-15 20:00:00
```

Anda dapat mereferensikan tabel eksternal Amazon Redshift Spectrum hanya dalam tampilan pengikatan terlambat. Salah satu aplikasi tampilan yang mengikat terlambat adalah menanyakan tabel Amazon Redshift dan Redshift Spectrum. Misalnya, Anda dapat menggunakan MEMBONGKAR perintah untuk mengarsipkan data lama ke Amazon S3. Kemudian, buat tabel eksternal Redshift Spectrum yang mereferensikan data di Amazon S3 dan buat tampilan yang menanyakan kedua tabel. Contoh berikut menggunakan klausa UNION ALL untuk bergabung dengan tabel Amazon Redshift dan `SALES` tabel Redshift Spectrum`SPECTRUM.SALES`.

```
create view sales_vw as
select * from public.sales
union all
select * from spectrum.sales
with no schema binding;
```

Untuk informasi selengkapnya tentang membuat tabel eksternal Redshift Spectrum, termasuk `SPECTRUM.SALES` tabel, lihatMemulai dengan Amazon Redshift Spectrum.

> **Important**
>
> Saat Anda membuat tampilan standar dari tampilan pengikatan terlambat, definisi tampilan standar berisi definisi tampilan pengikatan terlambat pada saat tampilan standar dibuat, termasuk pemilik tampilan pengikatan terlambat. Jika Anda membuat perubahan pada tampilan pengikatan terlambat yang mendasarinya, perubahan tersebut tidak akan digunakan

dalam tampilan standar sampai Anda membuat ulang tampilan standar. Oleh karena itu, ketika tampilan standar ditanyakan, itu akan selalu menggunakan definisi tampilan pengikatan terlambat dan pemilik tampilan pengikatan terlambat untuk pemeriksaan izin pada saat pembuatan tampilan standar tersebut.

Untuk memperbarui tampilan standar untuk merujuk ke definisi terbaru dari tampilan pengikatan terlambat, jalankan CREATE OR REPLACE VIEW dengan definisi tampilan awal yang Anda gunakan untuk membuat tampilan standar.

Lihat contoh berikut untuk membuat tampilan standar dari tampilan pengikatan terlambat.

```
create view sales_vw_lbv as
select * from public.sales
with no schema binding;

show view sales_vw_lbv;
                           Show View DDL statement
------------------------------------------------------------------------------
 create view sales_vw_lbv as select * from public.sales with no schema binding;
(1 row)

create view sales_vw as
select * from sales_vw_lbv;

show view sales_vw;
                                                 Show View DDL statement
------------------------------------------------------------------------------------------------------------
 SELECT sales_vw_lbv.price, sales_vw_lbv."region" FROM (SELECT sales.price,
 sales."region" FROM sales) sales_vw_lbv;
(1 row)
```

Perhatikan bahwa tampilan pengikatan terlambat seperti yang ditunjukkan dalam pernyataan DDL untuk tampilan standar ditentukan saat tampilan standar dibuat, dan tidak akan diperbarui dengan perubahan apa pun yang Anda buat pada tampilan pengikatan terlambat sesudahnya.

Contoh

Perintah contoh menggunakan set sampel objek dan data yang disebut database TICKIT. Untuk informasi selengkapnya, lihat Contoh database.

Perintah berikut membuat tampilan yang disebut myevent dari tabel bernama EVENT.

```
create view myevent as select eventname from event
where eventname = 'LeAnn Rimes';
```

Perintah berikut membuat tampilan yang disebut myuser dari tabel yang disebut USERS.

```
create view myuser as select lastname from users;
```

Perintah berikut membuat atau mengganti tampilan yang disebut myuser dari tabel yang disebut USERS.

```
create or replace view myuser as select lastname from users;
```

Contoh berikut membuat tampilan tanpa pengikatan skema.

```
create view myevent as select eventname from public.event
with no schema binding;
```

DEALOKASI

Menyelokasikan pernyataan yang disiapkan.

Sintaksis

```
DEALLOCATE [PREPARE] plan_name
```

Parameter

MEMPERSIAPKAN

Kata kunci ini opsional dan diabaikan.

nama_rencana

Nama pernyataan yang disiapkan untuk dialokasikan.

Catatan Penggunaan

DEALLOCATE digunakan untuk mengalokasikan pernyataan SQL yang disiapkan sebelumnya. Jika Anda tidak secara eksplisit mengalokasikan pernyataan yang disiapkan, pernyataan tersebut

akan dialokasikan ketika sesi saat ini berakhir. Untuk informasi lebih lanjut tentang pernyataan yang disiapkan, lihatMEMPERSIAPKAN.

Lihat Juga

EXECUTE, MEMPERSIAPKAN

MENYATAKAN

Mendefinisikan kursor baru. Gunakan kursor untuk mengambil beberapa baris sekaligus dari kumpulan hasil kueri yang lebih besar.

Ketika baris pertama kursor diambil, seluruh set hasil diwujudkan pada node pemimpin, dalam memori atau pada disk, jika diperlukan. Karena potensi dampak kinerja negatif dari penggunaan kursor dengan set hasil yang besar, kami sarankan menggunakan pendekatan alternatif bila memungkinkan. Untuk informasi selengkapnya, lihat Pertimbangan kinerja saat menggunakan kursor.

Anda harus mendeklarasikan kursor dalam blok transaksi. Hanya satu kursor pada satu waktu yang dapat dibuka per sesi.

Untuk informasi lebih lanjut, lihatAMBIL,TUTUP.

Sintaksis

```
DECLARE cursor_name CURSOR FOR query
```

Parameter

kursor_nama

Nama kursor baru.

query

Pernyataan SELECT yang mengisi kursor.

Catatan penggunaan DECLARE CURSOR

Jika aplikasi klien Anda menggunakan koneksi ODBC dan kueri Anda membuat kumpulan hasil yang terlalu besar untuk muat dalam memori, Anda dapat melakukan streaming set hasil ke aplikasi klien Anda dengan menggunakan kursor. Ketika Anda menggunakan kursor, seluruh rangkaian hasil diwujudkan pada node pemimpin, dan kemudian klien Anda dapat mengambil hasil secara bertahap.

> **Note**
>
> Untuk mengaktifkan kursor di ODBC untuk Microsoft Windows, aktifkan Declare/Fetch opsi Gunakan di DSN ODBC yang Anda gunakan untuk Amazon Redshift. Sebaiknya atur ukuran cache ODBC, menggunakan bidang Ukuran Cache di dialog opsi ODBC DSN, menjadi 4.000 atau lebih pada cluster multi-node untuk meminimalkan perjalanan pulang pergi. Pada cluster simpul tunggal, setel Ukuran Cache menjadi 1.000.

Karena potensi dampak kinerja negatif dari penggunaan kursor, kami sarankan menggunakan pendekatan alternatif bila memungkinkan. Untuk informasi selengkapnya, lihat Pertimbangan kinerja saat menggunakan kursor.

Kursor Amazon Redshift didukung dengan batasan berikut:

- Hanya satu kursor pada satu waktu yang dapat dibuka per sesi.
- Kursor harus digunakan dalam transaksi (BEGIN... END).
- Ukuran set hasil kumulatif maksimum untuk semua kursor dibatasi berdasarkan jenis node cluster. Ini adalah total seluruh cluster yang dibagi di semua kursor yang terbuka pada saat yang sama; kursor tunggal hanya dapat menggunakan sebagian dari itu, karena beberapa kapasitas dicadangkan sehingga beberapa kursor dapat berjalan secara bersamaan. Jika Anda membutuhkan set hasil yang lebih besar, Anda dapat mengubah ukuran ke konfigurasi node yang lebih besar.

  Untuk informasi selengkapnya, lihat Kendala kursor.

Kendala kursor

Ketika baris pertama kursor diambil, seluruh set hasil diwujudkan pada node pemimpin. Jika set hasil tidak muat dalam memori, itu ditulis ke disk sesuai kebutuhan. Untuk melindungi integritas node pemimpin, Amazon Redshift memberlakukan batasan pada ukuran kumpulan hasil kursor, berdasarkan jenis node cluster.

Nilai dalam tabel berikut adalah total seluruh cluster, bukan batas per kursor. Untuk memastikan bahwa beberapa kursor dapat dibuka secara bersamaan, sebagian dari total kapasitas ini dicadangkan untuk penggunaan bersamaan. Akibatnya, kursor tunggal tidak dapat menggunakan seluruh maksimum kumulatif sendiri; kumpulan hasil terbesar yang dapat ditahan oleh satu kursor kurang dari total yang ditunjukkan dalam tabel. Kursor individu dapat gagal ketika kumpulan hasilnya

mencapai batas per kursor ini, bahkan ketika kapasitas kursor keseluruhan cluster tidak sepenuhnya digunakan.

Tabel berikut menunjukkan ukuran set hasil kumulatif maksimum untuk setiap jenis node cluster. Ukuran set hasil maksimum dalam megabyte.

Tipe simpul	Hasil kumulatif maksimum yang ditetapkan per cluster (MB)
rg.12xlarge beberapa node	7,200.000
rg.4xlarge beberapa node	1.600.000
rg.xlarge beberapa node	240,000
rg.large beberapa node	100,000
rg.large simpul tunggal	8.000
ra3.16xlarge beberapa node	14.400.000
ra3.4xlarge beberapa node	3,200.000
ra3.xlplus beberapa node	1.000.000
ra3.xlplus simpul tunggal	64,000
ra3.large beberapa node	240,000
ra3.large simpul tunggal	8.000
Amazon Redshift Tanpa Server	150.000
dc2.8xlarge beberapa node	3,200.000
dc2.large beberapa node	192.000
dc2.large simpul tunggal	8.000

Untuk melihat konfigurasi kursor aktif untuk cluster, kueri tabel STV_KURSOR_KONFIGURASI sistem sebagai superuser. Untuk melihat keadaan kursor aktif, kueri tabel STV_ACTIVE_KURSOR

sistem. Hanya baris untuk kursor pengguna sendiri yang terlihat oleh pengguna, tetapi superuser dapat melihat semua kursor.

Pertimbangan kinerja saat menggunakan kursor

Karena kursor mewujudkan seluruh set hasil pada node pemimpin sebelum mulai mengembalikan hasil ke klien, menggunakan kursor dengan set hasil yang sangat besar dapat berdampak negatif pada kinerja. Kami sangat menyarankan agar tidak menggunakan kursor dengan set hasil yang sangat besar. Dalam beberapa kasus, seperti ketika aplikasi Anda menggunakan koneksi ODBC, kursor mungkin satu-satunya solusi yang layak. Jika memungkinkan, kami sarankan menggunakan alternatif ini:

- Gunakan MEMBONGKAR untuk mengekspor tabel besar. Saat Anda menggunakan UNLOAD, node komputasi bekerja secara paralel untuk mentransfer data langsung ke file data di Amazon Simple Storage Service. Untuk informasi selengkapnya, lihat Membongkar data di Amazon Redshift.
- Tetapkan parameter ukuran fetch JDBC di aplikasi klien Anda. Jika Anda menggunakan koneksi JDBC dan Anda mengalami kesalahan kehabisan memori sisi klien, Anda dapat mengaktifkan klien untuk mengambil kumpulan hasil dalam batch yang lebih kecil dengan menyetel parameter ukuran pengambilan JDBC. Untuk informasi selengkapnya, lihat Mengatur parameter ukuran pengambilan JDBC.

Node RG memiliki ukuran set hasil kursor maksimum yang lebih kecil dibandingkan dengan node RA3 dengan kapasitas komputasi yang serupa. Jika beban kerja kursor mendekati atau melebihi batas untuk jenis node RG Anda, pertimbangkan opsi berikut:

- Optimalkan ukuran set hasil. Terapkan filter kueri yang lebih selektif, kurangi jumlah kolom yang diambil, atau terapkan paginasi tingkat aplikasi untuk memproses hasil dalam batch yang lebih kecil.
- Bermigrasi ke tipe node RG yang lebih besar. Jenis node yang lebih besar memberikan kapasitas kursor yang lebih tinggi. Jika jumlah node memungkinkan dan kinerja beban kerja tidak terpengaruh, Anda dapat mengurangi jumlah node secara proporsional untuk mempertahankan total biaya cluster yang sebanding.

Untuk menentukan ukuran kumpulan hasil kursor Anda saat ini, kueri tampilan SYS_QUERY_HISTORY untuk `returned_bytes` nilai yang terkait dengan pernyataan

FETCH pertama yang dieksekusi terhadap kursor. Untuk informasi selengkapnya, lihat SYS_ QUERY_HISTORY.

Contoh DECLARE CURSOR

Contoh berikut mendeklarasikan kursor bernama LOLLAPALOOZA untuk memilih informasi penjualan untuk acara Lollapalooza, dan kemudian mengambil baris dari kumpulan hasil menggunakan kursor:

```
-- Begin a transaction

begin;

-- Declare a cursor

declare lollapalooza cursor for
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
where sales.eventid = event.eventid
and eventname='Lollapalooza';

-- Fetch the first 5 rows in the cursor lollapalooza:

fetch forward 5 from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-05-01 19:00:00 |   92.00000000 |       3
 Lollapalooza | 2008-11-15 15:00:00 |  222.00000000 |       2
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       3
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       4
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       1
(5 rows)

-- Fetch the next row:

fetch next from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-10-06 14:00:00 |  114.00000000 |       2

-- Close the cursor and end the transaction:
```

```
close lollapalooza;
commit;
```

Contoh berikut mengulang refcursor dengan semua hasil dari tabel:

```
CREATE TABLE tbl_1 (a int, b int);
INSERT INTO tbl_1 values (1, 2),(3, 4);

CREATE OR REPLACE PROCEDURE sp_cursor_loop() AS $$
DECLARE
    target record;
    curs1 cursor for select * from tbl_1;
BEGIN
    OPEN curs1;
    LOOP
        fetch curs1 into target;
        exit when not found;
        RAISE INFO 'a %', target.a;
    END LOOP;
    CLOSE curs1;
END;
$$ LANGUAGE plpgsql;

CALL sp_cursor_loop();

SELECT message
   from svl_stored_proc_messages
   where querytxt like 'CALL sp_cursor_loop()%';

  message
----------
      a 1
      a 3
```

DELETE

Menghapus baris dari tabel.

> **Note**
>
> Ukuran maksimum untuk satu pernyataan SQL adalah 16 MB.

Sintaksis

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
DELETE [ FROM ] { table_name | materialized_view_name }
    [ { USING } table_name, ... ]
    [ WHERE condition ]
```

Parameter

DENGAN klausa

Klausa opsional yang menentukan satu atau lebih ekspresi tab el-umum. Lihat DENGAN klausa.

FROM

Kata kunci FROM adalah opsional, kecuali ketika klausa USING ditentukan. Pernyataan `delete from event;` dan operasi setara `delete event;` yang menghapus semua baris dari tabel EVENT.

> **Note**
>
> Untuk menghapus semua baris dari tabel, MEMOTONG tabel. TRUNCATE jauh lebih efisien daripada DELETE dan tidak memerlukan VACUUM dan ANALYSIS. Namun, ketahuilah bahwa TRUNCATE melakukan transaksi di mana ia dijalankan.

table_name

Tabel sementara atau persisten. Hanya pemilik tabel atau pengguna dengan hak DELETE pada tabel yang dapat menghapus baris dari tabel.

Pertimbangkan untuk menggunakan perintah TRUNCATE untuk operasi penghapusan cepat tanpa syarat pada tabel besar; lihat. MEMOTONG

> **Note**
>
> Setelah menghapus sejumlah besar baris dari tabel:
>
> - Vakum meja untuk merebut kembali ruang penyimpanan dan mengurutkan ulang baris.
> - Analisis tabel untuk memperbarui statistik untuk perencana kueri.

nama_tampilan terwujud

Pandangan yang terwujud. Pernyataan DELETE bekerja pada tampilan terwujud yang digunakan untukStreaming konsumsi ke tampilan yang terwujud. Hanya pemilik tampilan yang terwujud atau pengguna dengan hak DELETE pada tampilan terwujud yang dapat menghapus baris darinya.

Anda tidak dapat menjalankan DELETE pada tampilan terwujud untuk penyerapan streaming dengan kebijakan keamanan tingkat baris (RLS) yang tidak memiliki izin IGNORE RLS yang diberikan kepada pengguna. Ada pengecualian untuk ini: Jika pengguna yang melakukan DELETE telah diberikan IGNORE RLS, itu berjalan dengan sukses. Untuk informasi selengkapnya, lihat kepemilikan dan manajemen kebijakan RLS.

MENGGUNAKAN table_name,...

Kata kunci USING digunakan untuk memperkenalkan daftar tabel ketika tabel tambahan direferensikan dalam kondisi klausa WHERE. Misalnya, pernyataan berikut menghapus semua baris dari tabel EVENT yang memenuhi kondisi join di atas tabel EVENT dan SALES. Tabel SALES harus dinamai secara eksplisit dalam daftar FROM:

```
delete from event using sales where event.eventid=sales.eventid;
```

Jika Anda mengulangi nama tabel target dalam klausa USING, operasi DELETE menjalankan self-join. Anda dapat menggunakan subquery dalam klausa WHERE alih-alih sintaks USING sebagai cara alternatif untuk menulis kueri yang sama.

Kondisi WHERE

Klausa opsional yang membatasi penghapusan baris ke baris yang cocok dengan kondisi. Misalnya, kondisi dapat berupa pembatasan pada kolom, kondisi gabungan, atau kondisi berdasarkan hasil kueri. Kueri dapat mereferensikan tabel selain target perintah DELETE. Contoh:

```
delete from t1
where col1 in(select col2 from t2);
```

Jika tidak ada kondisi yang ditentukan, semua baris dalam tabel dihapus.

Catatan penggunaan

- Operasi DELETE menahan kunci eksklusif saat dijalankan di Amazon Redshift streaming tampilan terwujud yang terhubung ke salah satu dari berikut ini:

- Aliran data Amazon Kinesis
- Topik Streaming Terkelola Amazon untuk Apache Kafka
- Aliran eksternal yang didukung, seperti topik Confluent Cloud Kafka

Untuk informasi selengkapnya, lihat Streaming konsumsi ke tampilan yang terwujud.

Contoh

Hapus semua baris dari tabel KATEGORI:

```
delete from category;
```

Hapus baris dengan nilai CATID antara 0 dan 9 dari tabel CATEGORY:

```
delete from category
where catid between 0 and 9;
```

Hapus baris dari tabel LISTING yang nilai SELLERID tidak ada di tabel SALES:

```
delete from listing
where listing.sellerid not in(select sales.sellerid from sales);
```

Dua kueri berikut menghapus satu baris dari tabel CATEGORY, berdasarkan gabungan ke tabel EVENT dan batasan tambahan pada kolom CATID:

```
delete from category
using event
where event.catid=category.catid and category.catid=9;
```

```
delete from category
where catid in
(select category.catid from category, event
where category.catid=event.catid and category.catid=9);
```

Kueri berikut menghapus semua baris dari tampilan ter `mv_cities` wujud. Nama tampilan terwujud dalam contoh ini adalah contoh:

```
delete from mv_cities;
```

DESKRIPSI DATA

Menampilkan daftar objek database dalam datashare yang ditambahkan ke dalamnya menggunakan ALTER DATASHARE. Amazon Redshift menampilkan nama, database, skema, dan jenis tabel, tampilan, dan fungsi.

Informasi tambahan tentang objek datashare dapat ditemukan dengan menggunakan tampilan sistem. Untuk informasi selengkapnya, lihat SVV_DATASHARE_OBJECTS dan SVV_DATASHARES. https://docs.aws.amazon.com/redshift/latest/dg/r_SVV_DATASHARES.html

Sintaksis

```
DESC DATASHARE datashare_name [ OF [ ACCOUNT account_id ] NAMESPACE namespace_guid ]
```

Parameter

datahare_name

Nama datashare.

NAMESPACE namespace_guid

Nilai yang menentukan namespace yang digunakan oleh datashare. Saat Anda menjalankan DESC DATAHSARE sebagai administrator cluster konsumen, tentukan parameter NAMESPACE untuk melihat data share masuk.

Akun akun_id

Nilai yang menentukan akun milik datashare.

Catatan Penggunaan

Sebagai administrator akun konsumen, saat Anda menjalankan DESC DATASHARE untuk melihat data share masuk dalam AWS akun, tentukan opsi NAMESPACE. Saat Anda menjalankan DESC DATASHARE untuk melihat data share masuk di seluruh AWS akun, tentukan opsi ACCOUNT dan NAMESPACE.

Contoh

Contoh berikut menampilkan informasi untuk berbagi data keluar pada cluster produsen.

```
DESC DATASHARE salesshare;

producer_account |          producer_namespace           | share_type  | share_name   |
 object_type |        object_name           |  include_new
-----------------+--------------------------------------+-------------+--------------
+-------------+------------------------------+--------------
 123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d  | OUTBOUND    |  salesshare  |
 TABLE       | public.tickit_sales_redshift |
 123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d  | OUTBOUND    |  salesshare  |
 SCHEMA      | public                       |   t
```

Contoh berikut menampilkan informasi untuk inbound datashares pada cluster konsumen.

```
DESC DATASHARE salesshare of ACCOUNT '123456789012' NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';

 producer_account |          producer_namespace           | share_type | share_name |
 object_type |         object_name           |  include_new
-----------------+--------------------------------------+------------+------------
+-------------+------------------------------+--------------
 123456789012     | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare |
 table       | public.tickit_sales_redshift |
 123456789012     | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare |
 schema      | public                       |
(2 rows)
```

PENYEDIA IDENTITAS DESC

Menampilkan informasi tentang penyedia identitas. Hanya pengguna super yang dapat menggambarkan penyedia identitas.

Sintaksis

```
DESC IDENTITY PROVIDER identity_provider_name
```

Parameter

nama penyedia identitas

Nama penyedia identitas.

Contoh

Contoh berikut menampilkan informasi tentang penyedia identitas.

```
DESC IDENTITY PROVIDER azure_idp;
```

Sampel keluaran.

```
  uid   |   name    | type  |             instanceid              | namespc |

                                          params

                           | enabled
--------+-----------+-------+-------------------------------------+---------
+-----------------------------------------------------------------------------------------------------------------
+---------
 126692 | azure_idp | azure | e40d4bb2-7670-44ae-bfb8-5db013221d73 | aad     |
 {"issuer":"https://login.microsoftonline.com/e40d4bb2-7670-44ae-bfb8-5db013221d73/
v2.0", "client_id":"871c010f-5e61-4fb1-83ac-98610a7e9110", "client_secret":'',
 "audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift", "https://
analysis.windows.net/powerbi/connector/AWSRDS"]} | t
(1 row)
```

LEPASKAN KEBIJAKAN MASKING

Melepaskan kebijakan penyembunyian data dinamis yang sudah dilampirkan dari kolom. Untuk informasi selengkapnya tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat melepaskan kebijakan masking.

Sintaksis

```
DETACH MASKING POLICY
{
  policy_name ON table_name
  | database_name.policy_name ON database_name.schema_name.table_name
}
( output_column_names )
FROM { user_name | ROLE role_name | PUBLIC };
```

Parameter

nama_kebijakan

Nama kebijakan masking yang akan dilepaskan.

database_name

Nama database tempat kebijakan dan relasi dibuat. Kebijakan dan hubungan harus berada di database yang sama. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

schema_name

Nama skema milik relasi.

table_name

Nama tabel untuk melepaskan kebijakan masking.

nama_kolom keluaran

Nama-nama kolom yang dilampirkan kebijakan masking.

nama_pengguna

Nama pengguna yang dilampirkan kebijakan masking.

Anda hanya dapat mengatur salah satu dari user_name, role_name, dan PUBLIC dalam satu pernyataan DETACH MASKING POLICY.

role_name

Nama peran yang dilampirkan kebijakan masking.

Anda hanya dapat mengatur salah satu dari user_name, role_name, dan PUBLIC dalam satu pernyataan DETACH MASKING POLICY.

PUBLIK

Menunjukkan bahwa kebijakan dilampirkan ke semua pengguna dalam tabel.

Anda hanya dapat mengatur salah satu dari user_name, role_name, dan PUBLIC dalam satu pernyataan DETACH MASKING POLICY.

Untuk penggunaan KEBIJAKAN DETACH MASKING di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

LEPASKAN KEBIJAKAN RLS

Lepaskan kebijakan keamanan tingkat baris pada tabel dari satu atau lebih pengguna atau peran.

Pengguna super dan pengguna atau peran yang memiliki `sys:secadmin` peran dapat melepaskan kebijakan.

Sintaksis

```
DETACH RLS POLICY
{
  policy_name ON [TABLE] table_name [, ...]
  | database_name.policy_name ON [TABLE] database_name.schema_name.table_name [, ...]
}
FROM { user_name | ROLE role_name | PUBLIC } [, ...];
```

Parameter

nama_kebijakan

Nama kebijakan .

database_name

Nama database tempat kebijakan dan relasi dibuat. Kebijakan dan hubungan harus berada di database yang sama. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

schema_name

Nama skema milik relasi.

table_name

Hubungan yang dilampirkan pada kebijakan keamanan tingkat baris.

DARI {user_name | ROLE role_name | PUBLIK} [,...]

Menentukan apakah kebijakan dipisahkan dari satu atau lebih pengguna atau peran tertentu.

Untuk penggunaan KEBIJAKAN DETACH RLS di Katalog Izin Federasi Amazon Redshift, lihat Mengelola kontrol akses dengan izin federasi Amazon Redshift.

Catatan penggunaan

Saat bekerja dengan pernyataan DETACH RLS POLICY, perhatikan hal berikut:

- Anda dapat melepaskan kebijakan dari relasi, pengguna, peran, atau publik.

Contoh

Contoh berikut melepaskan kebijakan pada tabel dari peran.

```
DETACH RLS POLICY policy_concerts ON tickit_category_redshift FROM ROLE analyst, ROLE
 dbadmin;
```

DROP DATABASE

Mengedrop basis data.

Anda tidak dapat menjalankan DROP DATABASE dalam blok transaksi (BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Sintaksis

```
DROP DATABASE database_name [ FORCE ]
```

Parameter

database_name

Nama database yang akan dihapus. Anda tidak dapat menghapus database dev, padb_harvest, template0, template1, atau sys:internal, dan Anda tidak dapat menghapus database saat ini.

Untuk menghapus database eksternal, lepaskan skema eksternal. Untuk informasi selengkapnya, lihat DROP SCHEMA.

KEKUATAN

Ketika Anda menentukan FORCE, DROP DATABASE mencoba untuk menghentikan koneksi aktif sebelum menjatuhkan database. Jika semua koneksi aktif berhasil dihentikan dalam batas waktu, penurunan akan berlanjut. Jika tidak semua koneksi berakhir, perintah akan menimbulkan kesalahan.

Catatan penggunaan DROP DATABASE

Saat menggunakan pernyataan DROP DATABASE, pertimbangkan hal berikut:

- Secara umum, kami menyarankan agar Anda tidak menghapus database yang berisi AWS Data Exchange datashare menggunakan pernyataan DROP DATABASE. Jika Anda melakukannya, Akun AWS yang memiliki akses ke datashare kehilangan akses. Melakukan perubahan jenis ini dapat melanggar ketentuan produk data AWS Data Exchange.

  Contoh berikut menunjukkan kesalahan ketika database yang berisi AWS Data Exchange datashare dijatuhkan.

  ```
  DROP DATABASE test_db;
  ERROR:   Drop of database test_db that contains ADX-managed datashare(s)
   requires session variable datashare_break_glass_session_var to be set to value
   'ce8d280c10ad41'
  ```

  Untuk memungkinkan penghapusan database, atur variabel berikut dan jalankan pernyataan DROP DATABASE lagi.

  ```
  SET datashare_break_glass_session_var to 'ce8d280c10ad41';
  ```

  ```
  DROP DATABASE test_db;
  ```

  Dalam hal ini, Amazon Redshift menghasilkan nilai satu kali acak untuk menyetel variabel sesi agar mengizinkan DROP DATABASE untuk database yang berisi AWS Data Exchange datashare.

Contoh

Contoh berikut menjatuhkan database bernama TICKIT_TEST:

```
drop database tickit_test;
```

DROP DATA

Menjatuhkan datashare. Perintah ini tidak dapat dibalik.

Hanya pengguna super atau pemilik datashare yang dapat menghapus datashare.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROP DATASHARE:

- Superuser
- Pengguna dengan hak istimewa DROP DATASHARE
- Pemilik Datashare

Sintaksis

```
DROP DATASHARE datashare_name;
```

Parameter

datahare_name

Nama datashare yang akan dihapus.

Catatan penggunaan DROP DATASHARE

Saat menggunakan pernyataan DROP DATASHARE, pertimbangkan hal berikut:

- Secara umum, kami menyarankan agar Anda tidak menghapus AWS Data Exchange datashare menggunakan pernyataan DROP DATASHARE. Jika Anda melakukannya, Akun AWS yang memiliki akses ke datashare kehilangan akses. Melakukan perubahan jenis ini dapat melanggar ketentuan produk data AWS Data Exchange.

  Contoh berikut menunjukkan kesalahan saat AWS Data Exchange datashare dijatuhkan.

  ```
  DROP DATASHARE salesshare;
  ERROR:  Drop of ADX-managed datashare salesshare requires session variable
   datashare_break_glass_session_var to be set to value '620c871f890c49'
  ```

  Untuk mengizinkan penghap AWS Data Exchange usan datashare, atur variabel berikut dan jalankan pernyataan DROP DATASHARE lagi.

  ```
  SET datashare_break_glass_session_var to '620c871f890c49';
  ```

```
DROP DATASHARE salesshare;
```

Dalam hal ini, Amazon Redshift menghasilkan nilai satu kali acak untuk menyetel variabel sesi agar mengizinkan DROP DATASHARE untuk datashare. AWS Data Exchange

Contoh

Contoh berikut menjatuhkan nama datashare. `salesshare`

```
DROP DATASHARE salesshare;
```

JATUHKAN TAMPILAN EKSTERNAL

Menjatuhkan tampilan eksternal dari database. Menjatuhkan tampilan eksternal akan menghapusnya dari semua mesin SQL yang terkait dengan tampilan, seperti Amazon Athena dan Amazon EMR Spark. Perintah ini tidak dapat dibalik. Untuk informasi selengkapnya tentang tampilan Katalog Data, lihat AWS Glue Data Catalog tampilan.

Sintaksis

```
DROP EXTERNAL VIEW schema_name.view_name [ IF EXISTS ]
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
 external_schema_name.view_name}
```

Parameter

nama_skema.nama_tampilan

Skema yang dilampirkan ke AWS Glue database Anda, diikuti dengan nama tampilan.

JIKA ADA

Menjatuhkan tampilan hanya jika ada.

nama_katalog.schema_nama.view_name | awsdatacatalog.dbname.view_name | nama_schema_eksternal_nama.view_name

Notasi skema yang akan digunakan saat menjatuhkan tampilan. Anda dapat menentukan untuk menggunakan AWS Glue Data Catalog, database Glue yang Anda buat, atau skema eksternal

yang Anda buat. Lihat MEMBUAT DATABASE dan MEMBUAT SKEMA EKSTERNAL untuk informasi selengkapnya.

definisi kueri

Definisi kueri SQL yang dijalankan Amazon Redshift untuk mengubah tampilan.

Contoh

Contoh berikut melepaskan tampilan Katalog Data bernama sample_schema.glue_data_catalog_view.

```
DROP EXTERNAL VIEW sample_schema.glue_data_catalog_view IF EXISTS
```

FUNGSI DROP

Menghapus fungsi yang ditentukan pengguna (UDF) dari database. Tanda tangan fungsi, atau daftar tipe data argumen, harus ditentukan karena beberapa fungsi dapat ada dengan nama yang sama tetapi tanda tangan yang berbeda. Anda tidak dapat menghapus fungsi bawaan Amazon Redshift.

Perintah ini tidak dapat dibalik.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROP FUNCTION:

- Superuser
- Pengguna dengan hak istimewa DROP FUNCTION
- Pemilik fungsi

Sintaksis

```
DROP FUNCTION name
( [arg_name] arg_type   [, ...] )
[ CASCADE | RESTRICT ]
```

Parameter

name

Nama fungsi yang akan dihapus.

arg_name

Nama argumen input. DROP FUNCTION mengabaikan nama argumen, karena hanya tipe data argumen yang diperlukan untuk menentukan identitas fungsi.

arg_type

Jenis data dari argumen input. Anda dapat menyediakan daftar yang dipisahkan koma dengan maksimal 32 tipe data.

KASKADE

Kata kunci menentukan untuk secara otomatis menghapus objek yang bergantung pada fungsi, seperti tampilan.

Untuk membuat tampilan yang tidak bergantung pada fungsi, sertakan klausa WITH NO SCHEMA BINDING dalam definisi tampilan. Untuk informasi selengkapnya, lihat CREATE VIEW.

MEMBATASI

Kata kunci yang menentukan bahwa jika ada objek yang bergantung pada fungsi, jangan lepaskan fungsi dan mengembalikan pesan. Tindakan ini adalah default.

Contoh

Contoh berikut menjatuhkan fungsi bernama`f_sqrt`:

```
drop function f_sqrt(int);
```

Untuk menghapus fungsi yang memiliki dependensi, gunakan opsi CASCADE, seperti yang ditunjukkan pada contoh berikut:

```
drop function f_sqrt(int)cascade;
```

KELOMPOK DROP

Menghapus grup pengguna. Perintah ini tidak dapat dibalik. Perintah ini tidak menghapus pengguna individu dalam grup.

Lihat DROP USER untuk menghapus pengguna individual.

Sintaksis

```
DROP GROUP name
```

Parameter

name

Nama grup pengguna yang akan dihapus.

Contoh

Contoh berikut menghapus grup `guests` pengguna:

```
DROP GROUP guests;
```

Anda tidak dapat menghapus grup jika grup memiliki hak istimewa pada objek. Jika Anda mencoba untuk menghapus grup seperti itu, Anda akan menerima kesalahan berikut.

```
ERROR: group "guests" can't be dropped because the group has a privilege on some object
```

Jika grup memiliki hak istimewa untuk objek, Anda harus mencabut hak istimewa sebelum menghentikan grup. Untuk menemukan objek yang memiliki hak istimewa `guests` grup, gunakan contoh berikut. Untuk informasi selengkapnya tentang tampilan metadata yang digunakan dalam contoh, lihat SVV _RELATION_PRIVILEGES.

```
SELECT DISTINCT namespace_name, relation_name, identity_name, identity_type
FROM svv_relation_privileges
WHERE identity_type='group' AND identity_name='guests';

+----------------+---------------+---------------+---------------+
| namespace_name | relation_name | identity_name | identity_type |
```

```
+----------------+---------------+---------------+---------------+
| public         | table1        | guests        | group         |
+----------------+---------------+---------------+---------------+
| public         | table2        | guests        | group         |
+----------------+---------------+---------------+---------------+
```

Contoh berikut mencabut semua hak istimewa pada semua tabel dalam `public` skema dari grup `guests` pengguna, dan kemudian menghapus grup.

```
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM GROUP guests;
DROP GROUP guests;
```

JATUHKAN PENYEDIA IDENTITAS

Menghapus penyedia identitas. Perintah ini tidak dapat dibalik. Hanya pengguna super yang dapat menghapus penyedia identitas.

Sintaksis

```
DROP IDENTITY PROVIDER identity_provider_name [ CASCADE ]
```

Parameter

nama penyedia identitas

Nama penyedia identitas yang akan dihapus.

KASKADE

Menghapus pengguna dan peran yang dilampirkan ke penyedia identitas, saat dihapus.

Contoh

Contoh berikut menghapus penyedia identitas oauth_provider.

```
DROP IDENTITY PROVIDER oauth_provider;
```

Jika Anda menghentikan penyedia identitas, beberapa pengguna mungkin tidak dapat masuk atau menggunakan alat klien yang dikonfigurasi untuk menggunakan penyedia identitas.

JATUHKAN PERPUSTAKAAN

Menghapus pustaka Python kustom dari database. Hanya pemilik perpustakaan atau superuser yang dapat menghapus perpustakaan.

DROP LIBRARY tidak dapat dijalankan di dalam blok transaksi (BEGIN... END). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Perintah ini tidak dapat dibalik. Perintah DROP LIBRARY segera dikomit. Jika UDF yang bergantung pada pustaka berjalan secara bersamaan, UDF mungkin gagal, bahkan jika UDF berjalan dalam transaksi.

Untuk informasi selengkapnya, lihat BUAT PERPUSTAKAAN.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROPLIBRARY:

- Superuser
- Pengguna dengan hak istimewa DROP LIBRARY
- Pemilik perpustakaan

Sintaksis

```
DROP LIBRARY library_name
```

Parameter

nama_perpustakaan

Nama perpustakaan.

KEBIJAKAN DROP MASKING

Menjatuhkan kebijakan penyembunyian data dinamis dari semua database. Anda tidak dapat menghapus kebijakan masking yang masih melekat pada satu atau beberapa tabel. Untuk informasi selengkapnya tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat menghapus kebijakan masking.

Sintaksis

```
DROP MASKING POLICY { policy_name | database_name.policy_name };
```

Parameter

nama_kebijakan

Nama kebijakan penyembunyian yang akan dihapus.

database_name

Nama database tempat kebijakan yang akan dihapus. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

Untuk penggunaan KEBIJAKAN DROP MASKING di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

MODEL JATUHKAN

Menghapus model dari database. Hanya pemilik model atau superuser yang dapat menjatuhkan model.

DROP MODEL juga menghapus semua fungsi prediksi terkait yang diturunkan dari model ini, semua artefak Amazon Redshift yang terkait dengan model, dan semua data Amazon S3 yang terkait dengan model. Sementara model masih dilatih di Amazon SageMaker AI, DROP MODEL akan membatalkan operasi tersebut.

Perintah ini tidak dapat dibalik. Perintah DROP MODEL segera dikomit.

Izin yang diperlukan

Berikut ini adalah izin yang diperlukan untuk DROP MODEL:

- Superuser
- Pengguna dengan izin DROP MODEL
- Pemilik model

- Pemilik skema

Sintaksis

```
DROP MODEL [ IF EXISTS ] model_name
```

Parameter

JIKA ADA

Klausa yang menunjukkan bahwa jika skema yang ditentukan sudah ada, perintah seharusnya tidak membuat perubahan dan mengembalikan pesan bahwa skema itu ada.

nama_model

Nama modul. Nama model dalam skema harus unik.

Contoh

Contoh berikut menjatuhkan model demo_ml.customer_churn.

```
DROP MODEL demo_ml.customer_churn
```

JATUHKAN TAMPILAN TERWUJUD

Menghapus tampilan yang terwujud.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

Sintaksis

```
DROP MATERIALIZED VIEW [ IF EXISTS ] mv_name [, ... ] [ CASCADE | RESTRICT ]
```

Parameter

JIKA ADA

Klausa yang menentukan untuk memeriksa apakah tampilan terwujud bernama ada. Jika tampilan terwujud tidak ada, maka `DROP MATERIALIZED VIEW` perintah mengembalikan pesan

kesalahan. Klausa ini berguna saat membuat skrip, untuk menjaga skrip agar tidak gagal jika Anda menghapus tampilan terwujud yang tidak ada.

mv_nama_

Nama tampilan yang terwujud yang akan dihapus.

KASKADE

Klausa yang menunjukkan untuk secara otomatis menghapus objek yang bergantung pada tampilan yang terwujud, seperti tampilan lain.

MEMBATASI

Klausa yang menunjukkan untuk tidak melepaskan tampilan yang terwujud jika ada objek yang bergantung padanya. Ini adalah opsi default.

Catatan Penggunaan

Hanya pemilik tampilan terwujud yang dapat menggunakan `DROP MATERIALIZED VIEW` tampilan itu. Seorang superuser atau pengguna yang secara khusus telah diberikan hak istimewa DROP dapat menjadi pengecualian untuk ini.

Saat Anda menulis pernyataan drop untuk tampilan terwujud dan tampilan dengan nama yang cocok ada, itu menghasilkan kesalahan yang menginstruksikan Anda untuk menggunakan DROP VIEW. Kesalahan terjadi bahkan dalam kasus di mana Anda menggunakan`DROP MATERIALIZED VIEW IF EXISTS`.

Contoh

Contoh berikut menjatuhkan tampilan yang `tickets_mv` terwujud.

```
DROP MATERIALIZED VIEW tickets_mv;
```

PROSEDUR DROP

Menjatuhkan prosedur. Untuk menghapus prosedur, nama prosedur dan tipe data argumen input (tanda tangan), diperlukan. Secara opsional, Anda dapat menyertakan tipe data argumen lengkap, termasuk argumen OUT. Untuk menemukan tanda tangan untuk prosedur, gunakan

TAMPILKAN PROSEDUR perintah. Untuk informasi selengkapnya tentang tanda tangan prosedur, lihatPG_PROC_INFO.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk PROSEDUR DROP:

- Superuser
- Pengguna dengan hak istimewa DROP PROCEDURE
- Pemilik prosedur

Sintaksis

```
DROP PROCEDURE sp_name ( [ [ argname ] [ argmode ] argtype [, ...] ] )
```

Parameter

sp_nama

Nama prosedur yang akan dihapus.

Argname

Nama argumen input. DROP PROCEDURE mengabaikan nama argumen, karena hanya tipe data argumen yang diperlukan untuk menentukan identitas prosedur.

Argmode

Mode argumen, yang bisa IN, OUT, atau INOUT. Argumen OUT bersifat opsional karena tidak digunakan untuk mengidentifikasi prosedur tersimpan.

Argtype

Jenis data dari argumen input. Untuk daftar tipe data yang didukung, lihatJenis Data.

Contoh

Contoh berikut menjatuhkan prosedur tersimpan bernama`quarterly_revenue`.

```
DROP PROCEDURE quarterly_revenue(volume INOUT bigint, at_price IN numeric,result OUT
 int);
```

KEBIJAKAN DROP RLS

Menjatuhkan kebijakan keamanan tingkat baris untuk semua tabel di semua database.

Pengguna super dan pengguna atau peran yang memiliki peran sys:secadmin dapat menghapus kebijakan.

Sintaksis

```
DROP RLS POLICY [ IF EXISTS ]
{ policy_name | database_name.policy_name }
[ CASCADE | RESTRICT ]
```

Parameter

JIKA ADA

Klausa yang menunjukkan apakah kebijakan yang ditentukan sudah ada.

nama_kebijakan

Nama kebijakan .

database_name

Nama database tempat kebijakan yang akan dihapus. Basis data dapat berupa database yang terhubung atau database yang mendukung izin federasi Amazon Redshift.

KASKADE

Klausa yang menunjukkan untuk secara otomatis melepaskan kebijakan dari semua tabel terlampir sebelum menghapus kebijakan.

MEMBATASI

Klausa yang menunjukkan untuk tidak menghapus kebijakan ketika dilampirkan ke beberapa tabel. Ini adalah opsi default.

Untuk penggunaan KEBIJAKAN DROP RLS di Katalog Izin Federasi Amazon Redshift, lihat M engelola kontrol akses dengan izin federasi Amazon Redshift.

Contoh

Contoh berikut menghapus kebijakan keamanan tingkat baris.

```
DROP RLS POLICY policy_concerts;
```

DROP ROLE

Menghapus peran dari database. Hanya pemilik peran yang membuat peran, pengguna dengan opsi WITH ADMIN, atau superuser yang dapat menghapus peran.

Anda tidak dapat menghapus peran yang diberikan kepada pengguna atau peran lain yang bergantung pada peran ini.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROP ROLE:

- Superuser
- Pemilik peran yang merupakan pengguna yang membuat peran atau pengguna yang telah diberikan peran dengan hak istimewa WITH ADMIN OPTION.

Sintaksis

```
DROP ROLE role_name [ FORCE | RESTRICT ]
```

Parameter

role_name

Nama peran.

[MEMAKSA | MEMBATASI]

Pengaturan default adalah RESTRICT. Amazon Redshift menampilkan kesalahan saat Anda mencoba menghapus peran yang telah mewarisi peran lain. Gunakan FORCE untuk menghapus semua penetapan peran, jika ada.

Contoh

Contoh berikut menghilangkan peran`sample_role`.

```
DROP ROLE sample_role FORCE;
```

Contoh berikut mencoba untuk menghapus peran sample_role1 yang telah diberikan kepada pengguna dengan opsi RESTRICT default.

```
CREATE ROLE sample_role1;
GRANT ROLE sample_role1 TO user1;
DROP ROLE sample_role1;
ERROR:  cannot drop this role since it has been granted on a user
```

Untuk berhasil menghapus sample_role1 yang telah diberikan kepada pengguna, gunakan opsi FORCE.

```
DROP ROLE sample_role1 FORCE;
```

Contoh berikut mencoba untuk menghapus peran sample_role2 yang memiliki peran lain yang bergantung padanya dengan opsi RESTRICT default.

```
CREATE ROLE sample_role1;
CREATE ROLE sample_role2;
GRANT ROLE sample_role1 TO sample_role2;
DROP ROLE sample_role2;
ERROR:  cannot drop this role since it depends on another role
```

Untuk berhasil menghapus sample_role2 yang memiliki peran lain yang bergantung padanya, gunakan opsi FORCE.

```
DROP ROLE sample_role2 FORCE;
```

DROP SCHEMA

Menghapus skema. Untuk skema eksternal, Anda juga dapat menghapus database eksternal yang terkait dengan skema. Perintah ini tidak dapat dibalik.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROP SCHEMA:

- Superuser
- Pemilik skema

- Pengguna dengan hak istimewa DROP SCHEMA

Sintaksis

```
DROP SCHEMA [ IF EXISTS ] name [, ...]
[ DROP EXTERNAL DATABASE ]
[ CASCADE | RESTRICT ]
```

Parameter

JIKA ADA

Klausa yang menunjukkan bahwa jika skema yang ditentukan tidak ada, perintah seharusnya tidak membuat perubahan dan mengembalikan pesan bahwa skema tidak ada, daripada diakhiri dengan kesalahan.

Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika DROP SCHEMA berjalan terhadap skema yang tidak ada.

name

Nama skema yang akan dijatuhkan. Anda dapat menentukan beberapa nama skema yang dipisahkan oleh koma.

JATUHKAN DATABASE EKSTERNAL

Klausa yang menunjukkan bahwa jika skema eksternal dijatuhkan, lepaskan database eksternal yang terkait dengan skema eksternal, jika ada. Jika tidak ada database eksternal, perintah mengembalikan pesan yang menyatakan bahwa tidak ada database eksternal. Jika beberapa skema eksternal dihapus, semua database yang terkait dengan skema yang ditentukan dihapus.

Jika database eksternal berisi objek dependen seperti tabel, sertakan opsi CASCADE untuk melepaskan objek dependen juga.

Ketika Anda menghapus database eksternal, database juga dihapus untuk skema eksternal lainnya yang terkait dengan database. Tabel yang didefinisikan dalam skema eksternal lain menggunakan database juga dijatuhkan.

DROP EXTERNAL DATABASE tidak mendukung database eksternal yang disimpan dalam metastor HIVE.

KASKADE

Kata kunci yang menunjukkan untuk secara otomatis menghapus semua objek dalam skema. Jika DROP EXTERNAL DATABASE ditentukan, semua objek dalam database eksternal juga dijatuhkan.

MEMBATASI

Kata kunci yang menunjukkan untuk tidak menghapus skema atau database eksternal jika berisi objek apa pun. Tindakan ini adalah default.

Contoh

Contoh berikut menghapus skema bernama S_SALES. Contoh ini menggunakan RESTRICT sebagai mekanisme keamanan sehingga skema tidak dihapus jika berisi objek apa pun. Dalam hal ini, Anda perlu menghapus objek skema sebelum menghapus skema.

```
drop schema s_sales restrict;
```

Contoh berikut menghapus skema bernama S_SALES dan semua objek yang bergantung pada skema itu.

```
drop schema s_sales cascade;
```

Contoh berikut menghapus skema S_SALES jika ada, atau tidak melakukan apa-apa dan mengembalikan pesan jika tidak.

```
drop schema if exists s_sales;
```

Contoh berikut menghapus skema eksternal bernama S_SPECTRUM dan database eksternal yang terkait dengannya. Contoh ini menggunakan RESTRICT sehingga skema dan database tidak dihapus jika mengandung objek apa pun. Dalam hal ini, Anda perlu menghapus objek dependen sebelum menghapus skema dan database.

```
drop schema s_spectrum drop external database restrict;
```

Contoh berikut menghapus beberapa skema dan database eksternal yang terkait dengannya, bersama dengan objek dependen.

```
drop schema s_sales, s_profit, s_revenue drop external database cascade;
```

MEJA DROP

Menghapus tabel dari database.

Jika Anda mencoba mengosongkan tabel baris, tanpa menghapus tabel, gunakan perintah DELETE atau TRUNCATE.

DROP TABLE menghapus batasan yang ada pada tabel target. Beberapa tabel dapat dihapus dengan perintah DROP TABLE tunggal.

DROP TABLE dengan tabel eksternal tidak dapat dijalankan di dalam transaksi (BEGIN... END). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Untuk menemukan contoh di mana hak istimewa DROP diberikan kepada grup, lihat GRANTContoh.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROP TABLE:

- Superuser
- Pengguna dengan hak istimewa DROP TABLE
- Pemilik tabel dengan hak istimewa USAGE pada skema

Sintaksis

```
DROP TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Parameter

JIKA ADA

Klausa yang menunjukkan bahwa jika tabel yang ditentukan tidak ada, perintah seharusnya tidak membuat perubahan dan mengembalikan pesan bahwa tabel tidak ada, daripada diakhiri dengan kesalahan.

Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika DROP TABLE berjalan terhadap tabel yang tidak ada.

name

Nama tabel yang akan dijatuhkan.

KASKADE

Klausa yang menunjukkan untuk secara otomatis melepaskan objek yang bergantung pada tabel, seperti tampilan.

Untuk membuat tampilan yang tidak bergantung pada objek database lain, seperti tampilan dan tabel, sertakan klausa WITH NO SCHEMA BINDING dalam definisi tampilan. Untuk informasi selengkapnya, lihat CREATE VIEW.

MEMBATASI

Klausa yang menunjukkan untuk tidak menjatuhkan tabel jika ada objek yang bergantung padanya. Tindakan ini adalah default.

Contoh

Menjatuhkan tabel tanpa ketergantungan

Contoh berikut membuat dan menjatuhkan tabel bernama FEEDBACK yang tidak memiliki dependensi:

```
create table feedback(a int);

drop table feedback;
```

Jika tabel berisi kolom yang direferensikan oleh tampilan atau tabel lain, Amazon Redshift menampilkan pesan seperti berikut ini.

```
Invalid operation: cannot drop table feedback because other objects depend on it
```

Menjatuhkan dua tabel secara bersamaan

Kumpulan perintah berikut membuat tabel FEEDBACK dan tabel BUYERS dan kemudian melepaskan kedua tabel dengan satu perintah:

```
create table feedback(a int);
```

```
create table buyers(a int);

drop table feedback, buyers;
```

Menjatuhkan tabel dengan ketergantungan

Langkah-langkah berikut menunjukkan cara melepaskan tabel yang disebut FEEDBACK menggunakan sakelar CASCADE.

Pertama, buat tabel sederhana yang disebut FEEDBACK menggunakan perintah CREATE TABLE:

```
create table feedback(a int);
```

Selanjutnya, gunakan perintah CREATE VIEW untuk membuat tampilan bernama FEEDBACK_VIEW yang bergantung pada tabel FEEDBACK:

```
create view feedback_view as select * from feedback;
```

Contoh berikut menjatuhkan tabel FEEDBACK dan juga menjatuhkan tampilan FEEDBACK_VIEW, karena FEEDBACK_VIEW bergantung pada tabel FEEDBACK:

```
drop table feedback cascade;
```

Melihat dependensi untuk tabel

Untuk mengembalikan dependensi untuk tabel Anda, gunakan contoh berikut. Ganti *my_schema* dan *my_table* dengan skema dan tabel Anda sendiri.

```
SELECT dependent_ns.nspname as dependent_schema
, dependent_view.relname as dependent_view
, source_ns.nspname as source_schema
, source_table.relname as source_table
, pg_attribute.attname as column_name
FROM pg_depend
JOIN pg_rewrite ON pg_depend.objid = pg_rewrite.oid
JOIN pg_class as dependent_view ON pg_rewrite.ev_class = dependent_view.oid
JOIN pg_class as source_table ON pg_depend.refobjid = source_table.oid
JOIN pg_attribute ON pg_depend.refobjid = pg_attribute.attrelid
    AND pg_depend.refobjsubid = pg_attribute.attnum
JOIN pg_namespace dependent_ns ON dependent_ns.oid = dependent_view.relnamespace
JOIN pg_namespace source_ns ON source_ns.oid = source_table.relnamespace
```

```
WHERE
source_ns.nspname = 'my_schema'
AND source_table.relname = 'my_table'
AND pg_attribute.attnum > 0
ORDER BY 1,2
LIMIT 10;
```

Untuk mel *my_table* epas dan dependensinya, gunakan contoh berikut. Contoh ini juga mengembalikan semua dependensi untuk tabel yang telah dihapus.

```
DROP TABLE my_table CASCADE;

SELECT dependent_ns.nspname as dependent_schema
, dependent_view.relname as dependent_view
, source_ns.nspname as source_schema
, source_table.relname as source_table
, pg_attribute.attname as column_name
FROM pg_depend
JOIN pg_rewrite ON pg_depend.objid = pg_rewrite.oid
JOIN pg_class as dependent_view ON pg_rewrite.ev_class = dependent_view.oid
JOIN pg_class as source_table ON pg_depend.refobjid = source_table.oid
JOIN pg_attribute ON pg_depend.refobjid = pg_attribute.attrelid
    AND pg_depend.refobjsubid = pg_attribute.attnum
JOIN pg_namespace dependent_ns ON dependent_ns.oid = dependent_view.relnamespace
JOIN pg_namespace source_ns ON source_ns.oid = source_table.relnamespace
WHERE
source_ns.nspname = 'my_schema'
AND source_table.relname = 'my_table'
AND pg_attribute.attnum > 0
ORDER BY 1,2
LIMIT 10;

+------------------+----------------+---------------+--------------+-------------+
| dependent_schema | dependent_view | source_schema | source_table | column_name |
+------------------+----------------+---------------+--------------+-------------+
```

Menjatuhkan tabel Menggunakan IF EXISTS

Contoh berikut menjatuhkan tabel FEEDBACK jika ada, atau tidak melakukan apa-apa dan mengembalikan pesan jika tidak:

```
drop table if exists feedback;
```

MENJATUHKAN TEMPLAT

Menjatuhkan template dari database.

Hak istimewa yang diperlukan

Untuk menghapus template, Anda harus memiliki salah satu dari berikut ini:

- Hak istimewa pengguna super
- Hak istimewa DROP TEMPLATE dan hak istimewa PENGGUNAAN pada skema yang berisi template

Sintaksis

```
DROP TEMPLATE [database_name.][schema_name.]template_name;
```

Parameter

database_name

(Opsional) Nama database tempat template dibuat. Jika tidak ditentukan, database saat ini digunakan.

nama skema

(Opsional) Nama skema tempat template dibuat. Jika tidak ditentukan, template dicari di jalur pencarian saat ini.

nama_templat

Nama template yang akan dihapus. Dalam contoh berikut, nama database adalah`demo_database`, nama skema adalah`demo_schema`, dan nama template adalah`test`.

```
DROP TEMPLATE demo_database.demo_schema.test;
```

Contoh

Contoh berikut menghapus template test_template dari skema saat ini:

```
DROP TEMPLATE test_template;
```

Contoh berikut menjatuhkan template test_template dari skema test_schema:

```
DROP TEMPLATE test_schema.test_template;
```

JATUHKAN PENGGUNA

Menjatuhkan pengguna dari database. Beberapa pengguna dapat dihapus dengan satu perintah DROP USER. Anda harus menjadi superuser database atau memiliki izin DROP USER untuk menjalankan perintah ini.

Sintaksis

```
DROP USER [ IF EXISTS ] name [, ... ]
```

Parameter

JIKA ADA

Klausa yang menunjukkan bahwa jika pengguna yang ditentukan tidak ada, perintah tidak boleh membuat perubahan dan mengembalikan pesan bahwa pengguna tidak ada, daripada diakhiri dengan kesalahan.

Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika DROP USER berjalan terhadap pengguna yang tidak ada.

name

Nama pengguna yang akan dihapus. Anda dapat menentukan beberapa pengguna, dengan koma memisahkan setiap nama pengguna dari yang berikutnya.

Catatan penggunaan

Anda tidak dapat menghapus pengguna bernama `rdsdb` atau pengguna administrator dari database yang biasanya diberi nama `awsuser` atau`admin`.

Anda tidak dapat menghapus pengguna jika pengguna memiliki objek database apa pun, seperti skema, database, tabel, atau tampilan, atau jika pengguna memiliki hak istimewa pada database,

tabel, kolom, atau grup. Jika Anda mencoba untuk menghapus pengguna tersebut, Anda menerima salah satu kesalahan berikut.

```
ERROR: user "username" can't be dropped because the user owns some object [SQL
 State=55006]

ERROR: user "username" can't be dropped because the user has a privilege on some object
 [SQL State=55006]
```

Untuk petunjuk terperinci tentang cara menemukan objek yang dimiliki oleh pengguna database, lihat Bagaimana cara mengatasi kesalahan “pengguna tidak dapat dijatuhkan” di Amazon Redshift? di Pusat Pengetahuan.

> **Note**
>
> Amazon Redshift hanya memeriksa database saat ini sebelum menghapus pengguna. DROP USER tidak mengembalikan kesalahan jika pengguna memiliki objek database atau memiliki hak istimewa pada objek di database lain. Jika Anda menghapus pengguna yang memiliki objek di database lain, pemilik untuk objek tersebut diubah menjadi 'tidak diketahui'.

Jika pengguna memiliki objek, pertama-tama jatuhkan objek atau ubah kepemilikannya ke pengguna lain sebelum menghapus pengguna asli. Jika pengguna memiliki hak istimewa untuk suatu objek, pertama-tama cabut hak istimewa sebelum menghapus pengguna. Contoh berikut menunjukkan menjatuhkan objek, mengubah kepemilikan, dan mencabut hak istimewa sebelum menghapus pengguna.

```
drop database dwdatabase;
alter schema dw owner to dwadmin;
revoke all on table dwtable from dwuser;
drop user dwuser;
```

Contoh

Contoh berikut menjatuhkan pengguna bernama paul:

```
drop user paulo;
```

Contoh berikut menjatuhkan dua pengguna, Paulo dan Martha:

```
drop user paulo, martha;
```

Contoh berikut menjatuhkan user paul jika ada, atau tidak melakukan apa-apa dan mengembalikan pesan jika tidak:

```
drop user if exists paulo;
```

MENJATUHKAN TAMPILAN

Menghapus tampilan dari database. Beberapa tampilan dapat dihapus dengan satu perintah DROP VIEW. Perintah ini tidak dapat dibalik.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk DROP VIEW:

- Superuser
- Pengguna dengan hak istimewa DROP VIEW
- Lihat pemilik

Sintaksis

```
DROP VIEW [ IF EXISTS ] name [, ... ] [ CASCADE | RESTRICT ]
```

Parameter

JIKA ADA

Klausa yang menunjukkan bahwa jika tampilan yang ditentukan tidak ada, perintah seharusnya tidak membuat perubahan dan mengembalikan pesan bahwa tampilan tidak ada, daripada diakhiri dengan kesalahan.

Klausa ini berguna saat membuat skrip, sehingga skrip tidak gagal jika DROP VIEW berjalan terhadap tampilan yang tidak ada.

name

Nama tampilan yang akan dihapus.

KASKADE

Klausa yang menunjukkan untuk secara otomatis melepaskan objek yang bergantung pada tampilan, seperti tampilan lain.

Untuk membuat tampilan yang tidak bergantung pada objek database lain, seperti tampilan dan tabel, sertakan klausa WITH NO SCHEMA BINDING dalam definisi tampilan. Untuk informasi selengkapnya, lihat CREATE VIEW.

Perhatikan bahwa jika Anda menyertakan CASCADE dan jumlah objek database turun menjadi sepuluh atau lebih, mungkin klien database Anda tidak akan mencantumkan semua objek yang dijatuhkan dalam hasil ringkasan. Ini biasanya karena alat klien SQL memiliki batasan default pada hasil yang dikembalikan.

MEMBATASI

Klausa yang menunjukkan untuk tidak melepaskan tampilan jika ada objek yang bergantung padanya. Tindakan ini adalah default.

Contoh

Contoh berikut menjatuhkan tampilan yang disebut event:

```
drop view event;
```

Untuk menghapus tampilan yang memiliki dependensi, gunakan opsi CASCADE. Misalnya, Anda mulai dengan tabel bernama EVENT. Anda kemudian membuat tampilan eventview dari tabel EVENT dengan menggunakan perintah CREATE VIEW, seperti yang ditunjukkan dalam contoh berikut:

```
create view eventview as
select dateid, eventname, catid
from event where catid = 1;
```

Sekarang, Anda membuat tampilan kedua yang disebut myeventview yang didasarkan pada tampilan pertama eventview:

```
create view myeventview as
select eventname, catid
from eventview where eventname <> ' ';
```

Pada titik ini, dua tampilan telah dibuat: eventview dan my eventview.

T ampilan myevent view adalah tampilan turunan dengan eventview sebagai induknya.

Untuk menghapus tampilan event view, perintah yang jelas untuk digunakan adalah sebagai berikut:

```
drop view eventview;
```

Perhatikan bahwa jika Anda menjalankan perintah ini dalam kasus ini, Anda mendapatkan kesalahan berikut:

```
drop view eventview;
ERROR: can't drop view eventview because other objects depend on it
HINT: Use DROP ... CASCADE to drop the dependent objects too.
```

Untuk memperbaikinya, jalankan perintah berikut (seperti yang disarankan dalam pesan kesalahan):

```
drop view eventview cascade;
```

Baik eventview dan myeventview sekarang telah berhasil dihapus.

Contoh berikut menjatuhkan tampilan event view jika ada, atau tidak melakukan apa-apa dan mengembalikan pesan jika tidak:

```
drop view if exists eventview;
```

AKHIR

Melakukan transaksi saat ini. Melakukan fungsi yang persis sama dengan perintah COMMIT.

Lihat COMMIT untuk dokumentasi yang lebih rinci.

Sintaksis

```
END [ WORK | TRANSACTION ]
```

Parameter

BEKERJA

Kata kunci opsional.

TRANSAKSI

Kata kunci opsional; KERJA dan TRANSAKSI adalah sinonim.

Contoh

Contoh berikut semua mengakhiri blok transaksi dan melakukan transaksi:

```
end;
```

```
end work;
```

```
end transaction;
```

Setelah salah satu perintah ini, Amazon Redshift mengakhiri blok transaksi dan melakukan perubahan.

EXECUTE

Menjalankan pernyataan yang disiapkan sebelumnya.

Sintaksis

```
EXECUTE plan_name [ (parameter [, ...]) ]
```

Parameter

nama_rencana

Nama pernyataan yang disiapkan untuk dijalankan.

Parameter

Nilai sebenarnya dari parameter untuk pernyataan yang disiapkan. Ini harus berupa ekspresi yang menghasilkan nilai tipe yang kompatibel dengan tipe data yang ditentukan untuk posisi parameter ini dalam perintah PREPARE yang membuat pernyataan siap.

Catatan penggunaan

EXECUTE digunakan untuk menjalankan pernyataan yang disiapkan sebelumnya. Karena pernyataan yang disiapkan hanya ada selama sesi, pernyataan yang disiapkan harus dibuat oleh pernyataan PREPARE yang dijalankan sebelumnya di sesi saat ini.

Jika pernyataan PREPARE sebelumnya menentukan beberapa parameter, kumpulan parameter yang kompatibel harus diteruskan ke pernyataan EXECUTE, atau Amazon Redshift mengembalikan kesalahan. Tidak seperti fungsi, pernyataan yang disiapkan tidak kelebihan beban berdasarkan jenis atau jumlah parameter yang ditentukan; nama pernyataan yang disiapkan harus unik dalam sesi database.

Ketika perintah EXECUTE dikeluarkan untuk pernyataan yang disiapkan, Amazon Redshift dapat secara opsional merevisi rencana eksekusi kueri (untuk meningkatkan kinerja berdasarkan nilai parameter yang ditentukan) sebelum menjalankan pernyataan yang disiapkan. Juga, untuk setiap eksekusi baru dari pernyataan yang disiapkan, Amazon Redshift dapat merevisi rencana eksekusi kueri lagi berdasarkan nilai parameter berbeda yang ditentukan dengan pernyataan EXECUTE. Untuk memeriksa rencana eksekusi kueri yang telah dipilih Amazon Redshift untuk setiap pernyataan EXECUTE yang diberikan, gunakan EXPLAIN perintah.

Untuk contoh dan informasi lebih lanjut tentang pembuatan dan penggunaan pernyataan yang disiapkan, lihatMEMPERSIAPKAN.

Lihat juga

DEALOKASI, MEMPERSIAPKAN

EXPLAIN

Menampilkan rencana eksekusi untuk pernyataan kueri tanpa menjalankan kueri. Untuk informasi tentang alur kerja analisis kueri, lihatAlur kerja analisis kueri.

Sintaksis

```
EXPLAIN [ VERBOSE ] query
```

Parameter

BERTELE-TELE

Menampilkan paket kueri lengkap, bukan hanya ringkasan.

query

Pernyataan kueri untuk dijelaskan. Kueri dapat berupa pernyataan SELECT, INSERT, CREATE TABLE AS, UPDATE, atau DELETE.

Catatan penggunaan

Kinerja EXPLAIN terkadang dipengaruhi oleh waktu yang dibutuhkan untuk membuat tabel sementara. Misalnya, kueri yang menggunakan optimasi subexpression umum memerlukan tabel sementara untuk dibuat dan dianalisis untuk mengembalikan output EXPLAIN. Rencana kueri tergantung pada skema dan statistik tabel sementara. Oleh karena itu, perintah EXPLAIN untuk jenis kueri ini mungkin membutuhkan waktu lebih lama untuk dijalankan dari yang diharapkan.

Anda dapat menggunakan EXPLAIN hanya untuk perintah berikut:

- SELECT
- PILIH MENJADI
- CREATE TABLE AS
- INSERT
- UPDATE
- DELETE

Perintah EXPLAIN akan gagal jika Anda menggunakannya untuk perintah SQL lainnya, seperti bahasa definisi data (DDL) atau operasi database.

Biaya unit relatif keluaran EXPLAIN digunakan oleh Amazon Redshift untuk memilih paket kueri. Amazon Redshift membandingkan ukuran berbagai perkiraan sumber daya untuk menentukan rencana.

Langkah-langkah perencanaan dan pelaksanaan kueri

Rencana eksekusi untuk pernyataan kueri Amazon Redshift tertentu memecah eksekusi dan perhitungan kueri menjadi urutan langkah dan operasi tabel terpisah yang akhirnya menghasilkan set hasil akhir untuk kueri. Untuk informasi tentang perencanaan kueri, lihatPemrosesan kueri.

Tabel berikut memberikan ringkasan langkah-langkah yang dapat digunakan Amazon Redshift dalam mengembangkan rencana eksekusi untuk setiap kueri yang dikirimkan pengguna untuk dieksekusi.

JELASKAN operator	Langkah-langkah eksekusi kueri	Deskripsi
MEMINDAI:		
Pemindaian Berurutan	scan	Operator atau langkah pemindaian relasi Amazon Redshift atau pemindaian tabel. Memindai seluruh tabel secara berurutan dari awal hingga akhir; juga mengevaluasi batasan kueri untuk setiap baris (Filter) jika ditentukan dengan klausa WHERE. Juga digunakan untuk menjalankan pernyataan INSERT, UPDATE, dan DELETE.
JOINS: Amazon Redshift menggunakan operator gabungan yang berbeda berdasarkan desain fisik tabel yang digabungkan, lokasi data yang diperlukan untuk bergabung, dan atribut spesifik dari kueri itu sendiri. Pemindaian Subquery - Pemindaian dan penambahan subquery digunakan untuk menjalankan kueri UNION.		
Lingkaran bersarang	nloop	Gabungan paling tidak optimal; terutama digunakan untuk penyambungan silang (produk Cartesian; tanpa kondisi gabungan) dan beberapa gabungan ketidaksetaraan.
Gabung Hash	bergabunglah	Juga digunakan untuk gabungan dalam dan gabungan luar kiri dan kanan dan biasanya lebih cepat daripada gabungan loop bersarang . Hash Join membaca tabel luar, men-hash kolom penggabungan, dan menemukan kecocokan di tabel hash bagian dalam.

JELASKAN operator	Langkah-langkah eksekusi kueri	Deskripsi
		Langkah bisa tumpah ke disk. (Input bagian dalam hjoin adalah langkah hash yang dapat berbasis disk.)
Gabungkan Gabung	bergabunglah	Juga digunakan untuk gabungan dalam dan gabungan luar (untuk tabel gabungan yang didistribusikan dan diurutkan pada kolom penggabungan). Biasanya algoritma bergabung Amazon Redshift tercepat, tidak termasuk pertimbangan biaya lainnya.
AGREGASI: Operator dan langkah-langkah yang digunakan untuk kueri yang melibatkan fungsi agregat dan operasi GROUP BY.		
Agregat	Aggr	Operator/step untuk fungsi agregat skalar.
HashAggregate	Aggr	Operator/step untuk fungsi agregat yang dikelompokkan. Dapat beroperasi dari disk berdasarkan tabel hash tumpah ke disk.
GroupAggregate	Aggr	Operator terkadang dipilih untuk kueri agregat yang dikelompokkan jika pengaturan konfigurasi Amazon Redshift untuk setelan force_hash_dinonaktifkan.
SORT: Operator dan langkah-langkah yang digunakan saat kueri harus mengurutkan atau menggabungkan kumpulan hasil.		
Urutkan	menyortir	Sort melakukan penyortiran yang ditentukan oleh klausa ORDER BY serta operasi lain seperti UniONS dan join. Dapat beroperasi dari disk.

JELASKAN operator	Langkah-langkah eksekusi kueri	Deskripsi
Gabungkan	bermerger	Menghasilkan hasil akhir yang diurutkan dari kueri berdasarkan hasil diurutkan menengah yang berasal dari operasi yang dilakukan secara paralel.
Operasi KECUALI, INTERSECT, dan UNION:		
SetOp Kecuali [Berbeda]	bergabunglah	Digunakan untuk kueri KECUALI. Dapat beroperasi dari disk berdasarkan fakta bahwa hash input dapat berbasis disk.
Hash Berpotongan [Berbeda]	bergabunglah	Digunakan untuk kueri INTERSECT. Dapat beroperasi dari disk berdasarkan fakta bahwa hash input dapat berbasis disk.
Tambahkan [Semua | Distinct]	save	Tambahkan digunakan dengan Subquery Scan untuk mengimplementasikan kueri UNION dan UNION ALL. Dapat beroperasi dari disk berdasarkan keutamaan “simpan”.
Miscellaneous/Other:		
Hash	hash	Digunakan untuk gabungan dalam dan gabungan luar kiri dan kanan (memberikan input ke gabungan hash). Operator Hash membuat tabel hash untuk tabel bagian dalam gabungan. (Tabel bagian dalam adalah tabel yang diperiksa untuk kecocokan dan, dalam gabungan dua tabel, biasanya yang lebih kecil dari keduanya.)
Kuota	batasan	Mengevaluasi klausa LIMIT.
Terwujud	save	Mewujudkan baris untuk input ke gabungan loop bersarang dan beberapa gabungan gabungan. Dapat beroperasi dari disk.

JELASKAN operator	Langkah-langkah eksekusi kueri	Deskripsi
--	mengurai	Digunakan untuk mengurai data input tekstual selama pemuatan.
--	proyek	Digunakan untuk mengatur ulang kolom dan menghitung ekspresi, yaitu data proyek.
Hasil	--	Jalankan fungsi skalar yang tidak melibatkan akses tabel apa pun.
--	kembali	Kembalikan baris ke pemimpin atau klien.
Subrencana	--	Digunakan untuk subquery tertentu.
Unik	khas	Menghilangkan duplikat dari kueri SELECT DISTINCT dan UNION.
Jendela	jendela	Hitung fungsi jendela agregat dan peringkat. Dapat beroperasi dari disk.
Operasi Jaringan:		
Jaringan (Siaran)	bcast	Siaran juga merupakan atribut dari operator dan langkah-langkah Join Explain.
Jaringan (Mendistr ibusikan)	distis	Mendistribusikan baris ke node komputasi untuk pemrosesan paralel oleh cluster gudang data.
Jaringan (Kirim ke Pemimpin)	kembali	Mengirim hasil kembali ke pemimpin untuk diproses lebih lanjut.
Operasi DML (operator yang memodifikasi data):		
Sisipkan (menggunakan Hasil)	sisipkan	Menyisipkan data.
Hapus (Pindai + Filter)	hapus	Menghapus data. Dapat beroperasi dari disk.

JELASKAN operator	Langkah-langkah eksekusi kueri	Deskripsi
Perbarui (Pindai + Filter)	hapus, masukkan	Diimplementasikan sebagai hapus dan sisipkan.

Menggunakan EXPLAIN untuk RLS

Jika kueri berisi tabel yang tunduk pada kebijakan keamanan tingkat baris (RLS), EXPLAIN menampilkan node RLS khusus. SecureScan Amazon Redshift juga mencatat jenis node yang sama ke tabel sistem STL_EXPLAIN. EXPLAIN tidak mengungkapkan predikat RLS yang berlaku untuk dim_tbl. Jenis SecureScan node RLS berfungsi sebagai indikator bahwa rencana eksekusi berisi operasi tambahan yang tidak terlihat oleh pengguna saat ini.

Contoh berikut menggambarkan SecureScan node RLS.

```
EXPLAIN
SELECT D.cint
FROM fact_tbl F INNER JOIN dim_tbl D ON F.k_dim = D.k
WHERE F.k_dim / 10 > 0;
                               QUERY PLAN
------------------------------------------------------------------------
 XN Hash Join DS_DIST_ALL_NONE  (cost=0.08..0.25 rows=1 width=4)
   Hash Cond: ("outer".k_dim = "inner"."k")
   ->  *XN* *RLS SecureScan f  (cost=0.00..0.14 rows=2 width=4)*
         Filter: ((k_dim / 10) > 0)
   ->  XN Hash  (cost=0.07..0.07 rows=2 width=8)
         ->  XN Seq Scan on dim_tbl d  (cost=0.00..0.07 rows=2 width=8)
               Filter: (("k" / 10) > 0)
```

Untuk mengaktifkan penyelidikan penuh terhadap paket kueri yang tunduk pada RLS, Amazon Redshift menawarkan izin sistem EXPLAIN RLS. Pengguna yang telah diberikan izin ini dapat memeriksa rencana kueri lengkap yang juga menyertakan predikat RLS.

Contoh berikut menggambarkan Seq Scan tambahan di bawah SecureScan node RLS juga menyertakan predikat kebijakan RLS (k_dim > 1).

```
EXPLAIN SELECT D.cint
FROM fact_tbl F INNER JOIN dim_tbl D ON F.k_dim = D.k
```

```
WHERE F.k_dim / 10 > 0;
                                    QUERY PLAN
------------------------------------------------------------------------------
 XN Hash Join DS_DIST_ALL_NONE  (cost=0.08..0.25 rows=1 width=4)
   Hash Cond: ("outer".k_dim = "inner"."k")
   *->  XN RLS SecureScan f  (cost=0.00..0.14 rows=2 width=4)
         Filter: ((k_dim / 10) > 0)*
         ->  *XN* *Seq Scan on fact_tbl rls_table  (cost=0.00..0.06 rows=5 width=8)
               Filter: (k_dim > 1)*
   ->  XN Hash  (cost=0.07..0.07 rows=2 width=8)
         ->  XN Seq Scan on dim_tbl d  (cost=0.00..0.07 rows=2 width=8)
               Filter: (("k" / 10) > 0)
```

Sementara izin EXPLAIN RLS diberikan kepada pengguna, Amazon Redshift mencatat paket kueri lengkap termasuk predikat RLS dalam tabel sistem STL_EXPLAIN. Kueri yang dijalankan saat izin ini tidak diberikan akan dicatat tanpa internal RLS. Memberikan atau menghapus izin EXPLAIN RLS tidak akan mengubah apa yang telah dicatat Amazon Redshift ke STL_EXPLAIN untuk kueri sebelumnya.

AWS Lake Formation-RLS melindungi hubungan Redshift

Contoh berikut menggambarkan SecureScan node LF, yang dapat Anda gunakan untuk melihat Formation-RLS relasi Lake.

```
EXPLAIN
SELECT *
FROM lf_db.public.t_share
WHERE a > 1;
QUERY PLAN
--------------------------------------------------------------
XN LF SecureScan t_share  (cost=0.00..0.02 rows=2 width=11)
(2 rows)
```

Contoh

> **Note**
>
> Untuk contoh ini, output sampel dapat bervariasi tergantung pada konfigurasi Amazon Redshift.

Contoh berikut mengembalikan rencana kueri untuk kueri yang memilih EVENTID, EVENTNAME, VENUEID, dan VENUENAME dari tabel EVENT dan VENUE:

```
explain
select eventid, eventname, event.venueid, venuename
from event, venue
where event.venueid = venue.venueid;
```

```
                                QUERY PLAN
------------------------------------------------------------------------------
XN Hash Join DS_DIST_OUTER  (cost=2.52..58653620.93 rows=8712 width=43)
Hash Cond: ("outer".venueid = "inner".venueid)
->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=23)
->  XN Hash  (cost=2.02..2.02 rows=202 width=22)
->  XN Seq Scan on venue  (cost=0.00..2.02 rows=202 width=22)
(5 rows)
```

Contoh berikut mengembalikan rencana kueri untuk kueri yang sama dengan output verbose:

```
explain verbose
select eventid, eventname, event.venueid, venuename
from event, venue
where event.venueid = venue.venueid;
```

```
                                QUERY PLAN
------------------------------------------------------------------------------
{HASHJOIN
:startup_cost 2.52
:total_cost 58653620.93
:plan_rows 8712
:plan_width 43
:best_pathkeys <>
:dist_info DS_DIST_OUTER
:dist_info.dist_keys (
TARGETENTRY
{
VAR
:varno 2
:varattno 1
...
```

```
XN Hash Join DS_DIST_OUTER  (cost=2.52..58653620.93 rows=8712 width=43)
Hash Cond: ("outer".venueid = "inner".venueid)
->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=23)
->  XN Hash  (cost=2.02..2.02 rows=202 width=22)
->  XN Seq Scan on venue  (cost=0.00..2.02 rows=202 width=22)
(519 rows)
```

Contoh berikut mengembalikan rencana kueri untuk pernyataan CREATE TABLE AS (CTAS):

```
explain create table venue_nonulls as
select * from venue
where venueseats is not null;

QUERY PLAN
-------------------------------------------------------------
XN Seq Scan on venue  (cost=0.00..2.02 rows=187 width=45)
Filter: (venueseats IS NOT NULL)
(2 rows)
```

AMBIL

Mengambil baris menggunakan kursor. Untuk informasi tentang mendeklarasikan kursor, lihatMENYATAKAN.

FETCH mengambil baris berdasarkan posisi saat ini di dalam kursor. Ketika kursor dibuat, kursor diposisikan sebelum baris pertama. Setelah FETCH, kursor diposisikan pada baris terakhir yang diambil. Jika FETCH berjalan dari akhir baris yang tersedia, seperti mengikuti FETCH ALL, kursor dibiarkan diposisikan setelah baris terakhir.

FORWARD 0 mengambil baris saat ini tanpa menggerakkan kursor; artinya, ia mengambil baris yang terakhir diambil. Jika kursor diposisikan sebelum baris pertama atau setelah baris terakhir, tidak ada baris yang dikembalikan.

Ketika baris pertama kursor diambil, seluruh set hasil diwujudkan pada node pemimpin, dalam memori atau pada disk, jika diperlukan. Karena potensi dampak kinerja negatif dari penggunaan kursor dengan set hasil yang besar, kami sarankan menggunakan pendekatan alternatif bila memungkinkan. Untuk informasi selengkapnya, lihat Pertimbangan kinerja saat menggunakan kursor.

Untuk informasi lebih lanjut, lihatMENYATAKAN,TUTUP.

Sintaksis

```
FETCH [ NEXT | ALL | {FORWARD [ count | ALL ] } ] FROM cursor
```

Parameter

BERIKUTNYA

Mengambil baris berikutnya. Ini adalah opsi default.

SEMUA

Mengambil semua baris yang tersisa. (Sama seperti FORWARD ALL.) ALL tidak didukung untuk cluster simpul tunggal.

MAJU [hitung | SEMUA]

Mengambil baris hitungan berikutnya, atau semua baris yang tersisa. `FORWARD 0` mengambil baris saat ini. Untuk cluster simpul tunggal, nilai maksimum untuk hitungan adalah `1000`. FORWARD ALL tidak didukung untuk cluster simpul tunggal.

kursor

Nama kursor baru.

Contoh FETCH

Contoh berikut mendeklarasikan kursor bernama LOLLAPALOOZA untuk memilih informasi penjualan untuk acara Lollapalooza, dan kemudian mengambil baris dari kumpulan hasil menggunakan kursor:

```
-- Begin a transaction

begin;

-- Declare a cursor

declare lollapalooza cursor for
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
where sales.eventid = event.eventid
```

```
and eventname='Lollapalooza';

-- Fetch the first 5 rows in the cursor lollapalooza:

fetch forward 5 from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-05-01 19:00:00 |   92.00000000 |       3
 Lollapalooza | 2008-11-15 15:00:00 |  222.00000000 |       2
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       3
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       4
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       1
(5 rows)

-- Fetch the next row:

fetch next from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-10-06 14:00:00 |  114.00000000 |       2

-- Close the cursor and end the transaction:

close lollapalooza;
commit;
```

HIBAH

Mendefinisikan izin akses untuk pengguna atau peran.

Izin termasuk opsi akses seperti dapat membaca data dalam tabel dan tampilan, menulis data, membuat tabel, dan melepas tabel. Gunakan perintah ini untuk memberikan izin khusus untuk tabel, database, skema, fungsi, prosedur, bahasa, atau kolom. Untuk mencabut izin dari objek database, gunakan MENCABUT perintah.

Izin juga mencakup opsi akses produsen datashare berikut:

- Memberikan akses datashare ke ruang nama konsumen dan akun.
- Memberikan izin untuk mengubah datashare dengan menambahkan atau menghapus objek dari datashare.

- Memberikan izin untuk berbagi datashare dengan menambahkan atau menghapus ruang nama konsumen dari datashare.

Opsi akses konsumen Datashare adalah sebagai berikut:

- Memberikan pengguna akses penuh ke database yang dibuat dari datashare atau skema eksternal yang mengarah ke database tersebut.
- Memberikan izin tingkat objek kepada pengguna pada database yang dibuat dari datashare seperti yang Anda bisa untuk objek database lokal. Untuk memberikan tingkat izin ini, Anda harus menggunakan klausa WITH PERMISSIONS saat membuat database dari datashare. Untuk informasi selengkapnya, lihat BUAT BASIS DATA.

Untuk informasi selengkapnya tentang izin datashare, lihat. Izin yang dapat Anda berikan untuk berbagi data

Izin juga mencakup Katalog Izin Federasi Amazon Redshift berikut:

- Memberikan izin tingkat tabel kepada pengguna dan peran.
- Memberikan izin tingkat kolom berbutir halus pada tabel, tampilan, dan tampilan yang terwujud.
- Memberikan izin cakupan kepada pengguna dan peran.
- Memberikan izin tingkat database di Katalog Izin Federasi Amazon Redshift.

Untuk informasi selengkapnya tentang mengelola izin di Katalog Izin Federasi Amazon Redshift, lihat. Mengelola kontrol akses di katalog izin federasi Amazon Redshift Untuk informasi selengkapnya tentang grant/revoke sintaks yang didukung Katalog Izin Federasi Amazon Redshift, lihat. Grant/Revoke

Izin juga menyertakan hak istimewa CONNECT untuk pengguna AWS IAM Identity Center federasi. Hak istimewa ini memungkinkan administrator untuk mengontrol akses pengguna melalui izin terperinci di setiap kelompok kerja atau cluster Amazon Redshift di mana Izin Federasi Amazon Redshift diaktifkan. Administrator Amazon Redshift dapat menentukan pengguna AWS IAM Identity Center atau grup federasi mana yang memiliki akses untuk terhubung langsung ke grup kerja Amazon Redshift, memberikan kontrol halus atas akses AWS IAM Identity Center pengguna di setiap kelompok kerja atau cluster.

Anda juga dapat memberikan peran untuk mengelola izin database dan mengontrol apa yang dapat dilakukan pengguna relatif terhadap data Anda. Dengan menentukan peran dan menetapkan peran

kepada pengguna, Anda dapat membatasi tindakan yang dapat dilakukan pengguna tersebut, seperti membatasi pengguna hanya pada perintah CREATE TABLE dan INSERT. Untuk informasi selengkapnya tentang perintah CREATE ROLE, lihatthe section called “CREATE ROLE”. Amazon Redshift memiliki beberapa peran yang ditentukan sistem yang juga dapat Anda gunakan untuk memberikan izin khusus kepada pengguna Anda. Untuk informasi selengkapnya, lihat the section called “Peran yang ditentukan sistem Amazon Redshift”.

Anda hanya dapat MEMBERIKAN atau MENCABUT izin USAGE pada skema eksternal kepada pengguna database dan grup pengguna yang menggunakan sintaks ON SCHEMA. Saat menggunakan ON EXTERNAL SCHEMA dengan AWS Lake Formation, Anda hanya dapat MEMBERIKAN dan MENCABUT izin ke peran AWS Identity and Access Management (IAM). Untuk daftar izin, lihat sintaks.

Untuk prosedur tersimpan, satu-satunya izin yang dapat Anda berikan adalah EXECUTE.

Anda tidak dapat menjalankan GRANT (pada sumber daya eksternal) dalam blok transaksi (BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Untuk melihat izin mana yang telah diberikan pengguna untuk database, gunakanMEMILIKI HAK ISTIMEWA BASIS DATA. Untuk melihat izin mana yang telah diberikan pengguna untuk skema, gunakanMEMILIKI HAK ISTIMEWA SKEMA. Untuk melihat izin mana yang telah diberikan pengguna untuk tabel, gunakanMEMILIKI HAK ISTIMEWA TABEL_.

Sintaksis

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | ALTER | TRUNCATE }
 [,...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { { CREATE | USAGE | TEMPORARY | TEMP | ALTER } [,...] | ALL [ PRIVILEGES ] }
    ON DATABASE db_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
 FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
 PROCEDURES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT USAGE
    ON LANGUAGE language_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { { ALTER | DROP} [,...] | ALL [ PRIVILEGES ] }
    ON COPY JOB job_name [,...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { { ALTER | DROP | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON TEMPLATE [database_name.][schema_name.]template_name [,...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

Memberikan izin tingkat kolom untuk tabel

Berikut ini adalah sintaks untuk izin tingkat kolom pada tabel dan tampilan Amazon Redshift.

```
GRANT { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
 ( column_name [,...] ) }
     ON { [ TABLE ] table_name [, ...] }

     TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

Pemberian izin ASSUMEROLE

Berikut ini adalah sintaks untuk izin ASSUMEROLE yang diberikan kepada pengguna dan grup dengan peran tertentu. Untuk mulai menggunakan hak istimewa ASSUMEROLE, lihat. Catatan penggunaan untuk memberikan izin ASSUMEROLE

```
GRANT ASSUMEROLE
       ON { 'iam_role' [, ...] | default | ALL }
       TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
       FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]
```

Pemberian izin untuk integrasi Redshift Spectrum dengan Lake Formation

Berikut ini adalah sintaks untuk integrasi Redshift Spectrum dengan Lake Formation.

```
GRANT { SELECT | ALL [ PRIVILEGES ] } ( column_list )
    ON EXTERNAL TABLE schema_name.table_name
    TO { IAM_ROLE iam_role } [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | ALTER | DROP | DELETE | INSERT }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL TABLE schema_name.table_name [, ...]
    TO { { IAM_ROLE iam_role } [, ...] | PUBLIC } [ WITH GRANT OPTION ]

GRANT { { CREATE | ALTER | DROP }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL SCHEMA schema_name [, ...]
    TO { IAM_ROLE iam_role } [, ...] [ WITH GRANT OPTION ]
```

Pemberian izin datashare

Producer-side izin datashare

Berikut ini adalah sintaks untuk menggunakan GRANT untuk memberikan izin ALTER atau SHARE kepada pengguna atau peran. Pengguna dapat mengubah datashare dengan izin ALTER, atau memberikan penggunaan kepada konsumen dengan izin SHARE. ALTER dan SHARE adalah satu-satunya izin yang dapat Anda berikan pada datashare kepada pengguna dan peran.

```
GRANT { ALTER | SHARE } ON DATASHARE datashare_name
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

Berikut ini adalah sintaks untuk menggunakan GRANT untuk izin penggunaan datashare di Amazon Redshift. Anda memberikan akses ke datashare kepada konsumen menggunakan izin PENGGUNAAN. Anda tidak dapat memberikan izin ini kepada pengguna atau grup pengguna. Izin ini juga tidak mendukung OPSI WITH GRANT untuk pernyataan GRANT. Hanya pengguna atau grup pengguna dengan izin SHARE yang sebelumnya diberikan kepada mereka UNTUK datashare yang dapat menjalankan jenis pernyataan GRANT ini.

```
GRANT USAGE
    ON DATASHARE datashare_name
    TO NAMESPACE 'namespaceGUID' | ACCOUNT 'accountnumber' [ VIA DATA CATALOG ]
```

Berikut ini adalah contoh bagaimana memberikan penggunaan datashare ke akun Formasi Danau.

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012' VIA DATA CATALOG;
```

Consumer-side izin datashare

Berikut ini adalah sintaks untuk izin penggunaan berbagi data GRANT pada database atau skema tertentu yang dibuat dari datashare.

Izin lebih lanjut yang diperlukan bagi konsumen untuk mengakses database yang dibuat dari datashare bervariasi tergantung pada apakah perintah CREATE DATABASE yang digunakan untuk membuat database dari datashare menggunakan klausa WITH PERMISSIONS atau tidak. Untuk informasi selengkapnya tentang perintah CREATE DATABASE dan WITH PERMISSIONS klausa, lihatBUAT BASIS DATA.

Database dibuat tanpa menggunakan klausa WITH PERMISSIONS

Bila Anda memberikan USAGE pada database yang dibuat dari datashare tanpa klausa WITH PERMISSIONS, Anda tidak perlu memberikan izin secara terpisah pada objek dalam database bersama. Entitas yang diberikan penggunaan pada database yang dibuat dari datashares tanpa klausa WITH PERMISSIONS secara otomatis memiliki akses ke semua objek dalam database.

Database yang dibuat menggunakan klausa WITH PERMISSIONS

Saat Anda memberikan USAGE pada database tempat database bersama dibuat dari datashare dengan klausa WITH PERMISSIONS, identitas sisi konsumen masih harus diberikan izin yang relevan untuk objek database dalam database bersama untuk mengaksesnya, sama seperti Anda akan memberikan izin untuk objek database lokal. Untuk memberikan izin ke objek dalam database yang dibuat dari datashare, gunakan sintaks tiga bagian. `database_name.schema_name.object_name` Untuk memberikan izin ke objek dalam skema eksternal yang menunjuk ke skema bersama dalam database bersama, gunakan sintaks `schema_name.object_name` dua bagian.

```
GRANT USAGE ON { DATABASE shared_database_name [, ...] | SCHEMA shared_schema}
    TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

Pemberian izin bercakupan

Izin cakupan memungkinkan Anda memberikan izin kepada pengguna atau peran pada semua objek dari jenis dalam database atau skema. Pengguna dan peran dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema.

Anda dapat melihat cakupan izin cakupan tingkat database di. SVV_DATABASE_HAK ISTIMEWA
Anda dapat melihat cakupan izin cakupan tingkat skema di. SVV_SCHEMA_HAK ISTIMEWA

Untuk informasi selengkapnya tentang izin bercakupan, lihatIzin terlingkup.

Berikut ini adalah sintaks untuk memberikan izin cakupan kepada pengguna dan peran.

```
GRANT { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }
FOR SCHEMAS IN
DATABASE db_name
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

GRANT
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [PRIVILEGES] } }
FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name} [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name | } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
FOR PROCEDURES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name | } [, ...]

GRANT USAGE
FOR LANGUAGES IN
{DATABASE db_name}
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

GRANT { { CREATE | ALTER | DROP} [,...] | ALL [ PRIVILEGES ] }
FOR COPY JOBS
IN DATABASE db_name
```

```
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

GRANT { { ALTER | DROP | USAGE } [,...] | ALL [ PRIVILEGES ] }
FOR TEMPLATES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]
```

Perhatikan bahwa izin cakupan tidak membedakan antara izin untuk fungsi dan prosedur. Misalnya, pernyataan berikut memberikan bob EXECUTE izin untuk fungsi dan prosedur dalam skema`Sales_schema`.

```
GRANT EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema TO bob;
```

Memberikan izin pembelajaran mesin

Berikut ini adalah sintaks untuk izin model pembelajaran mesin di Amazon Redshift.

```
GRANT CREATE MODEL
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON MODEL model_name [, ...]

    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

Pemberian izin peran

Berikut ini adalah sintaks untuk memberikan peran di Amazon Redshift.

```
GRANT { ROLE role_name } [, ...] TO { { user_name [ WITH ADMIN OPTION ] } |
 ROLE role_name }[, ...]
```

Berikut ini adalah sintaks untuk memberikan izin sistem ke peran di Amazon Redshift. Perhatikan bahwa Anda hanya dapat memberikan izin untuk peran, bukan pengguna.

```
GRANT
  {
```

```
    { CREATE USER | DROP USER | ALTER USER |
    CREATE SCHEMA | DROP SCHEMA |
    ALTER DEFAULT PRIVILEGES |
    ACCESS CATALOG | ACCESS SYSTEM TABLE
    CREATE TABLE | DROP TABLE | ALTER TABLE |
    CREATE OR REPLACE FUNCTION | CREATE OR REPLACE EXTERNAL FUNCTION |
    DROP FUNCTION |
    CREATE OR REPLACE PROCEDURE | DROP PROCEDURE |
    CREATE OR REPLACE VIEW | DROP VIEW |
    CREATE MODEL | DROP MODEL |
    CREATE DATASHARE | ALTER DATASHARE | DROP DATASHARE |
    CREATE LIBRARY | DROP LIBRARY |
    CREATE ROLE | DROP ROLE |
    TRUNCATE TABLE
    VACUUM | ANALYZE | CANCEL |
    IGNORE RLS | EXPLAIN RLS |
    EXPLAIN MASKING }[, ...]
  }
  | { ALL [ PRIVILEGES ] }
TO ROLE role_name [, ...]
```

Pemberian izin penjelasan untuk kebijakan keamanan

Berikut ini adalah sintaks untuk memberikan izin untuk menjelaskan filter kebijakan keamanan dari kueri dalam paket EXPLAIN. Kebijakan keamanan yang mungkin termasuk kebijakan keamanan tingkat baris dan kebijakan penyembunyian data dinamis.

```
GRANT EXPLAIN { RLS | MASKING } TO ROLE rolename
```

Berikut ini adalah sintaks untuk memberikan izin untuk melewati kebijakan keamanan tingkat baris untuk kueri. Sintaks ini tidak berlaku untuk kebijakan penyembunyian data dinamis.

```
GRANT IGNORE RLS TO ROLE rolename
```

Berikut ini adalah sintaks untuk memberikan izin tabel pencarian ke kebijakan keamanan yang ditentukan. Kebijakan keamanan yang mungkin termasuk kebijakan keamanan tingkat baris dan kebijakan penyembunyian data dinamis.

```
GRANT SELECT ON [ TABLE ] table_name [, ...]
TO { RLS | MASKING } POLICY policy_name [, ...]
```

Memberikan izin koneksi

Berikut ini adalah sintaks untuk memberikan izin bagi pengguna AWS IAM Identity Center federasi (atau grup) untuk terhubung ke: workgroup/cluster

```
GRANT CONNECT [ON WORKGROUP]
TO <prefix>:<username> | ROLE <prefix>:<rolename> | PUBLIC;
```

Parameter

SELECT

Memberikan izin untuk memilih data dari tabel atau tampilan menggunakan pernyataan SELECT. Izin SELECT juga diperlukan untuk mereferensikan nilai kolom yang ada untuk operasi UPDATE atau DELETE.

INSERT

Memberikan izin untuk memuat data ke dalam tabel menggunakan pernyataan INSERT atau pernyataan COPY.

UPDATE

Memberikan izin untuk memperbarui kolom tabel menggunakan pernyataan UPDATE. Operasi UPDATE juga memerlukan izin SELECT, karena harus mereferensikan kolom tabel untuk menentukan baris mana yang akan diperbarui, atau untuk menghitung nilai baru untuk kolom.

DELETE

Memberikan izin untuk menghapus baris data dari tabel. Operasi DELETE juga memerlukan izin SELECT, karena harus mereferensikan kolom tabel untuk menentukan baris mana yang akan dihapus.

DROP

Bergantung pada objek database, memberikan izin berikut kepada pengguna atau peran:

- Untuk tabel, DROP memberikan izin untuk menjatuhkan tabel atau tampilan. Untuk informasi selengkapnya, lihat MEJA DROP.
- Untuk database, DROP memberikan izin untuk menghapus database. Untuk informasi selengkapnya, lihat DROP DATABASE.
- Untuk skema, DROP memberikan izin untuk menghapus skema. Untuk informasi selengkapnya, lihat DROP SCHEMA.

REFERENSI

Memberikan izin untuk membuat batasan kunci asing. Anda perlu memberikan izin ini pada tabel yang direferensikan dan tabel referensi; jika tidak, pengguna tidak dapat membuat batasan.

ALTER

Bergantung pada objek database, memberikan izin berikut kepada pengguna atau grup pengguna:

- Untuk tabel, ALTER memberikan izin untuk mengubah tabel atau tampilan. Untuk informasi selengkapnya, lihat ALTER TABLE.
- Untuk database, ALTER memberikan izin untuk mengubah database. Untuk informasi selengkapnya, lihat ALTER DATABASE.
- Untuk skema, ALTER memberikan izin untuk mengubah skema. Untuk informasi selengkapnya, lihat ALTER SCHEMA.
- Untuk tabel eksternal, ALTER memberikan izin untuk mengubah tabel di tabel AWS Glue Data Catalog yang diaktifkan untuk Formasi Danau. Izin ini hanya berlaku saat menggunakan Formasi Danau.

TRUNCATE

Memberikan izin untuk memotong tabel. Tanpa izin ini, hanya pemilik tabel atau superuser yang dapat memotong tabel. Untuk informasi selengkapnya tentang perintah TRUNCATE, lihatthe section called “MEMOTONG”.

SEMUA [HAK ISTIMEWA]

Memberikan semua izin yang tersedia sekaligus kepada pengguna atau peran yang ditentukan. Kata kunci PRIVILEGES adalah opsional.

GRANT ALL ON SCHEMA tidak memberikan izin CREATE untuk skema eksternal.

Anda dapat memberikan izin ALL ke tabel AWS Glue Data Catalog yang diaktifkan untuk Formasi Danau. Dalam hal ini, izin individu (seperti SELECT, ALTER, dan sebagainya) dicatat dalam Katalog Data.

> **Note**
>
> Amazon Redshift tidak mendukung izin RULE dan TRIGGER. Untuk informasi selengkapnya, kunjungi Fitur PostgreSQL yang tidak didukung.

ASSUMEROLE

Memberikan izin untuk menjalankan perintah COPY, UNLOAD, EXTERNAL FUNCTION, dan CREATE MODEL kepada pengguna, peran, atau grup dengan peran tertentu. Pengguna, peran, atau grup mengasumsikan peran itu saat menjalankan perintah yang ditentukan. Untuk mulai menggunakan izin ASSUMEROLE, lihat. Catatan penggunaan untuk memberikan izin ASSUMEROLE

PADA [TABEL] nama_ tabel

Memberikan izin yang ditentukan pada tabel atau tampilan. Kata kunci TABLE adalah opsional. Anda dapat membuat daftar beberapa tabel dan tampilan dalam satu pernyataan.

PADA SEMUA TABEL DI SCHEMA schema_name

Memberikan izin yang ditentukan pada semua tabel dan tampilan dalam skema yang direferensikan.

(column_name [,...]) PADA TABLE table_name

Memberikan izin yang ditentukan kepada pengguna, grup, atau PUBLIK pada kolom tertentu dari tabel atau tampilan Amazon Redshift.

(column_list) PADA TABEL EKSTERNAL schema_name.table_name

Memberikan izin yang ditentukan ke peran IAM pada kolom tertentu dari tabel Formasi Danau dalam skema yang direferensikan.

PADA TABEL EKSTERNAL schema_name.table_name

Memberikan izin yang ditentukan ke peran IAM pada tabel Formasi Danau yang ditentukan dalam skema yang direferensikan.

PADA NAMA skema SKEMA EKSTERNAL

Memberikan izin yang ditentukan ke peran IAM pada skema yang direferensikan.

DI iam_role

Memberikan izin yang ditentukan ke peran IAM.

Nama pengguna TO

Menunjukkan pengguna yang menerima izin.

UNTUK IAM_ROLE iam_role

Menunjukkan peran IAM yang menerima izin.

DENGAN OPSI HIBAH

Menunjukkan bahwa pengguna yang menerima izin pada gilirannya dapat memberikan izin yang sama kepada orang lain. DENGAN OPSI HIBAH tidak dapat diberikan kepada grup atau PUBLIK.

Nama peran

Memberikan izin untuk peran.

Grup nama_grup

Memberikan izin ke grup pengguna. Dapat berupa daftar yang dipisahkan koma untuk menentukan beberapa grup pengguna.

PUBLIK

Memberikan izin yang ditentukan kepada semua pengguna, termasuk pengguna yang dibuat nanti. PUBLIK mewakili grup yang selalu mencakup semua pengguna. Izin pengguna individu terdiri dari jumlah izin yang diberikan kepada PUBLIK, izin yang diberikan kepada grup mana pun yang dimiliki pengguna, dan izin apa pun yang diberikan kepada pengguna secara individual.

Pemberian PUBLIK ke TABEL EKSTERNAL Formasi Danau menghasilkan pemberian izin kepada kelompok semua orang Formasi Danau.

HUBUNGKAN [DI WORKGROUP] KE {<prefix>: <username>| PERAN<prefix>: <rolename>| PUBLIK}

Memberikan izin untuk terhubung ke grup kerja atau cluster kepada pengguna atau grup AWS IAM Identity Center federasi. Awalan mengidentifikasi penyedia identitas. Ketika diberikan ke PUBLIK, izin berlaku untuk semua pengguna AWS IAM Identity Center federasi, termasuk pengguna yang dibuat nanti. Izin ini hanya berlaku jika Izin Federasi Amazon Redshift diaktifkan pada workgroup atau cluster.

CREATE

Bergantung pada objek database, memberikan izin berikut kepada pengguna atau grup pengguna:

- Untuk database, CREATE memungkinkan pengguna untuk membuat skema dalam database.
- Untuk skema, CREATE memungkinkan pengguna untuk membuat objek dalam skema. Untuk mengganti nama objek, pengguna harus memiliki izin CREATE dan memiliki objek yang akan diganti namanya.
- CREATE ON SCHEMA tidak didukung untuk skema eksternal Amazon Redshift Spectrum. Untuk memberikan penggunaan tabel eksternal dalam skema eksternal, berikan USAGE

ON SCHEMA kepada pengguna yang membutuhkan akses. Hanya pemilik skema eksternal atau superuser yang diizinkan untuk membuat tabel eksternal dalam skema eksternal. Untuk mentransfer kepemilikan skema eksternal, gunakan ALTER SCHEMA untuk mengubah pemilik.

SEMENTARA | TEMP

Memberikan izin untuk membuat tabel sementara dalam database yang ditentukan. Untuk menjalankan kueri Amazon Redshift Spectrum, pengguna database harus memiliki izin untuk membuat tabel sementara di database.

> **Note**
>
> Secara default, pengguna diberikan izin untuk membuat tabel sementara oleh keanggotaan otomatis mereka di grup PUBLIK. Untuk menghapus izin bagi pengguna mana pun untuk membuat tabel sementara, cabut izin TEMP dari grup PUBLIK. Kemudian secara eksplisit memberikan izin untuk membuat tabel sementara kepada pengguna atau grup pengguna tertentu.

PADA DATABASE db_name

Memberikan izin yang ditentukan pada database.

PENGGUNAAN

Memberikan izin PENGGUNAAN pada skema tertentu, yang membuat objek dalam skema tersebut dapat diakses oleh pengguna. Tindakan spesifik pada objek ini harus diberikan secara terpisah (misalnya, izin SELECT atau UPDATE pada tabel) untuk skema Amazon Redshift lokal. Secara default, semua pengguna memiliki izin CREATE dan USAGE pada skema PUBLIK.

Saat Anda memberikan USAGE ke skema eksternal menggunakan sintaks ON SCHEMA, Anda tidak perlu memberikan tindakan secara terpisah pada objek dalam skema eksternal. Izin katalog yang sesuai mengontrol izin granular pada objek skema eksternal.

ON SCHEMA schema_name

Memberikan izin yang ditentukan pada skema.

GRANT CREATE ON SCHEMA dan izin BUAT di GRANT ALL ON SCHEMA tidak didukung untuk skema eksternal Amazon Redshift Spectrum. Untuk memberikan penggunaan tabel eksternal dalam skema eksternal, berikan USAGE ON SCHEMA kepada pengguna yang membutuhkan

akses. Hanya pemilik skema eksternal atau superuser yang diizinkan untuk membuat tabel eksternal dalam skema eksternal. Untuk mentransfer kepemilikan skema eksternal, gunakan ALTER SCHEMA untuk mengubah pemilik.

JALANKAN PADA SEMUA FUNGSI DALAM SCHEMA schema_name

Memberikan izin yang ditentukan pada semua fungsi dalam skema yang direferensikan.

Amazon Redshift tidak mendukung pernyataan GRANT atau REVOKE untuk entri bawaan pg_proc yang ditentukan dalam namespace pg_catalog.

JALANKAN NAMA_prosedur PADA PROSED UR

Memberikan izin EXECUTE pada prosedur tersimpan tertentu. Karena nama prosedur yang disimpan dapat kelebihan beban, Anda harus menyertakan daftar argumen untuk prosedur. Untuk informasi selengkapnya, lihat Penamaan prosedur yang disimpan.

JALANKAN PADA SEMUA PROSEDUR DALAM SCHEMA schema_name

Memberikan izin yang ditentukan pada semua prosedur tersimpan dalam skema yang direferensikan.

PENGGUNAAN PADA LANGUAGE language_name

Memberikan izin PENGGUNAAN pada suatu bahasa.

> **Note**
>
> Mulai 1 November 2025, Amazon Redshift tidak akan lagi mendukung pembuatan UDF Python baru. UDF Python yang ada akan terus berfungsi hingga 30 Juni 2026. Mulai 1 Juli 2026, Amazon Redshift tidak akan lagi mendukung Python UDF. Kami menyarankan Anda memigrasikan UDF Python yang ada ke UDF Lambda sebelum 1 November 2025. Untuk informasi tentang membuat dan menggunakan Lambda UDF, lihat. UDF Lambda Skalar Untuk informasi tentang mengonversi UDF Python yang ada ke UDF Lambda, lihat posting blog. https://aws.amazon.com/blogs/big-data/amazon-redshift-python-user-defined-functions-will-reach-end-of-support-after-june-30-2026/

Izin USAGE ON LANGUAGE diperlukan untuk membuat fungsi yang ditentukan pengguna (UDF) dengan menjalankan perintahCREATE FUNCTION. Untuk informasi selengkapnya, lihat Keamanan dan izin UDF.

Izin USAGE ON LANGUAGE diperlukan untuk membuat prosedur tersimpan dengan menjalankan BUAT PROSEDUR perintah. Untuk informasi selengkapnya, lihat Keamanan dan hak istimewa untuk prosedur tersimpan.

Untuk Python UDF, gunakan`plpythonu`. Untuk SQL UDF, gunakan`sql`. Untuk prosedur yang disimpan, gunakan`plpgsql`.

ON COPY JOB nama_ pekerjaan

Memberikan izin yang ditentukan pada pekerjaan penyalinan.

UNTUK {SEMUA | SALIN | MEMBONGKAR | FUNGSI EKSTERNAL | BUAT MODEL} [,...]

Menentukan perintah SQL yang izin diberikan. Anda dapat menentukan SEMUA untuk memberikan izin pada pernyataan COPY, UNLOAD, EXTERNAL FUNCTION, dan CREATE MODEL. Klausul ini hanya berlaku untuk memberikan izin ASSUMEROLE.

MENGUBAH

Memberikan izin ALTER kepada pengguna untuk menambah atau menghapus objek dari datashare, atau untuk mengatur properti PUBLICACCESSIBLE. Untuk informasi selengkapnya, lihat ALTER DATA.

BERBAGI

Memberikan pemrissi kepada pengguna dan grup pengguna untuk menambahkan konsumen data ke datashare. Izin ini diperlukan untuk mengaktifkan konsumen tertentu (akun atau namespace) untuk mengakses datashare dari cluster mereka. Konsumen dapat berupa AWS akun yang sama atau berbeda, dengan namespace cluster yang sama atau berbeda seperti yang ditentukan oleh pengenal unik global (GUID).

PADA DATASHARE datashare_name

Memberikan izin yang ditentukan pada datashare yang direferensikan. Untuk informasi tentang perincian kontrol akses konsumen, lihatBerbagi data di berbagai tingkat di Amazon Redshift.

PENGGUNAAN

Ketika USAGE diberikan ke akun konsumen atau namespace dalam akun yang sama, akun konsumen tertentu atau namespace dalam akun dapat mengakses datashare dan objek datashare dengan cara read-only.

KE NAMESPACE 'GUID' ruang nama cluster'

Menunjukkan namespace di akun yang sama tempat konsumen dapat menerima izin yang ditentukan ke datashare. Ruang nama menggunakan GUID alfanumerik 128-bit.

KE AKUN 'nomor akun' [VIA KATALOG DATA]

Menunjukkan jumlah akun lain yang konsumennya dapat menerima izin yang ditentukan ke datashare. Menentukan 'VIA DATA CATALOG' menunjukkan bahwa Anda memberikan penggunaan datashare ke akun Lake Formation. Menghilangkan parameter ini berarti Anda memberikan penggunaan ke akun yang memiliki cluster.

PADA DATABASE shared_database_name> [,...]

Memberikan izin penggunaan yang ditentukan pada database tertentu yang dibuat di datashare yang ditentukan.

PADA SKEMA shared_schema

Memberikan izin yang ditentukan pada skema tertentu yang dibuat di datashare yang ditentukan.

UNTUK {SKEMA | TABEL | FUNGSI | PROSEDUR | BAHASA | PEKERJAAN SALIN} DI

Menentukan objek database untuk memberikan izin. Parameter berikut IN menentukan ruang lingkup izin yang diberikan.

BUAT MODEL

Memberikan izin CREATE MODEL kepada pengguna atau grup pengguna tertentu.

NAMA_model ON MODEL

Memberikan izin EXECUTE pada model tertentu.

AKSES KATALOG

Memberikan izin untuk melihat metadata objek yang relevan yang dapat diakses peran tersebut.

{peran} [,...]

Peran yang akan diberikan kepada peran lain, pengguna, atau PUBLIK.

PUBLIK mewakili grup yang selalu mencakup semua pengguna. Izin pengguna individu terdiri dari jumlah izin yang diberikan kepada PUBLIK, izin yang diberikan kepada grup mana pun yang dimiliki pengguna, dan izin apa pun yang diberikan kepada pengguna secara individual.

UNTUK {{user_name [DENGAN OPSI ADMIN]} | peran} [,...]

Memberikan peran yang ditentukan kepada pengguna tertentu dengan OPSI DENGAN ADMIN, peran lain, atau PUBLIK.

Klausa WITH ADMIN OPTION menyediakan opsi administrasi untuk semua peran yang diberikan kepada semua penerima hibah.

JELASKAN {RLS | MASKING} UNTUK ROLE ROLE

Memberikan izin untuk menjelaskan filter kebijakan keamanan dari kueri dalam rencana EXPLAIN ke peran. RLS memberikan izin untuk menjelaskan filter kebijakan keamanan tingkat baris. MASKING memberikan izin untuk menjelaskan filter kebijakan penyembunyian data dinamis.

ABAIKAN RLS UNTUK ROLE rolename

Memberikan izin untuk melewati kebijakan keamanan tingkat baris untuk kueri ke peran.

NAMA KEBIJAKAN KE {RLS | MASKING} KEBIJAKAN

Menunjukkan kebijakan keamanan yang menerima izin. KEBIJAKAN TO RLS menunjukkan kebijakan keamanan tingkat baris. TO MASKING POLICY menunjukkan kebijakan penyembunyian data dinamis.

Catatan penggunaan

Untuk mempelajari lebih lanjut tentang catatan penggunaan untuk GRANT, lihatthe section called “Catatan penggunaan”.

Contoh

Untuk contoh cara menggunakan GRANT, lihatthe section called “Contoh”.

Catatan penggunaan

Untuk memberikan hak istimewa pada objek, Anda harus memenuhi salah satu kriteria berikut:

- Jadilah pemilik objek.
- Jadilah pengguna super.
- Miliki hak istimewa hibah untuk objek dan hak istimewa itu.

Misalnya, perintah berikut memungkinkan HR pengguna untuk melakukan perintah SELECT pada tabel karyawan dan untuk memberikan dan mencabut hak istimewa yang sama untuk pengguna lain.

```
grant select on table employees to HR with grant option;
```

HR tidak dapat memberikan hak istimewa untuk operasi apa pun selain SELECT, atau pada tabel lain selain karyawan.

Sebagai contoh lain, perintah berikut memungkinkan HR pengguna untuk melakukan perintah ALTER pada tabel karyawan dan untuk memberikan dan mencabut hak istimewa yang sama untuk pengguna lain.

```
grant ALTER on table employees to HR with grant option;
```

HR tidak dapat memberikan hak istimewa untuk operasi apa pun selain ALTER, atau di meja lain selain karyawan.

Memiliki hak istimewa yang diberikan pada tampilan tidak berarti memiliki hak istimewa pada tabel yang mendasarinya. Demikian pula, memiliki hak istimewa yang diberikan pada skema tidak menyiratkan memiliki hak istimewa pada tabel dalam skema. Sebagai gantinya, berikan akses ke tabel yang mendasarinya secara eksplisit.

Untuk memberikan hak istimewa ke AWS Lake Formation tabel, peran IAM yang terkait dengan skema eksternal tabel harus memiliki izin untuk memberikan hak istimewa ke tabel eksternal. Contoh berikut membuat skema eksternal dengan peran `myGrantor` IAM terkait. Peran IAM `myGrantor` memiliki izin untuk memberikan izin kepada orang lain. Perintah GRANT menggunakan izin peran IAM `myGrantor` yang terkait dengan skema eksternal untuk memberikan izin ke peran IAM. `myGrantee`

```
create external schema mySchema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myGrantor'
create external database if not exists;
```

```
grant select
on external table mySchema.mytable
to iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

Jika Anda MEMBERIKAN SEMUA hak istimewa ke peran IAM, hak istimewa individu diberikan dalam Katalog Data yang diaktifkan Lake Formation terkait. Misalnya, GRANT ALL berikut menghasilkan hak istimewa individu yang diberikan (SELECT, ALTER, DROP, DELETE, dan INSERT) yang ditampilkan di konsol Lake Formation.

```
grant all
on external table mySchema.mytable
to iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

Superuser dapat mengakses semua objek terlepas dari perintah GRANT dan REVOKE yang menetapkan hak istimewa objek.

Catatan penggunaan untuk kontrol akses tingkat kolom

Catatan penggunaan berikut berlaku untuk hak istimewa tingkat kolom pada tabel dan tampilan Amazon Redshift. Catatan ini menjelaskan tabel; catatan yang sama berlaku untuk tampilan kecuali pengecualian dicatat secara eksplisit.

- Untuk tabel Amazon Redshift, Anda hanya dapat memberikan hak istimewa SELECT dan UPDATE di tingkat kolom. Untuk tampilan Amazon Redshift, Anda hanya dapat memberikan hak istimewa SELECT di tingkat kolom.
- Kata kunci ALL adalah sinonim untuk hak istimewa SELECT dan UPDATE yang digabungkan ketika digunakan dalam konteks GRANT tingkat kolom pada tabel.
- Jika Anda tidak memiliki hak istimewa SELECT pada semua kolom dalam tabel, melakukan operasi SELECT * hanya mengembalikan kolom yang Anda akses. Saat menggunakan tampilan, operasi SELECT * mencoba mengakses semua kolom dalam tampilan. Jika Anda tidak memiliki izin untuk mengakses semua kolom, kueri ini gagal dengan kesalahan izin ditolak.
- SELECT * tidak meluas ke kolom yang dapat diakses hanya dalam kasus berikut:
  - Anda tidak dapat membuat tampilan biasa dengan hanya kolom yang dapat diakses menggunakan SELECT *.
  - Anda tidak dapat membuat tampilan terwujud hanya dengan kolom yang dapat diakses menggunakan SELECT *.
- Jika Anda memiliki hak istimewa SELECT atau UPDATE pada tabel atau tampilan dan menambahkan kolom, Anda masih memiliki hak istimewa yang sama pada tabel atau tampilan dan dengan demikian semua kolomnya.
- Hanya pemilik tabel atau superuser yang dapat memberikan hak istimewa tingkat kolom.
- Klausa WITH GRANT OPTION tidak didukung untuk hak istimewa tingkat kolom.
- Anda tidak dapat memegang hak istimewa yang sama di tingkat tabel dan tingkat kolom. Misalnya, pengguna tidak `data_scientist` dapat memiliki hak istimewa SELECT pada tabel `employee` dan hak pilih pada kolom`employee.department`. Pertimbangkan hasil berikut saat memberikan hak istimewa yang sama ke tabel dan kolom di dalam tabel:

  - Jika pengguna memiliki hak istimewa tingkat tabel pada tabel, maka pemberian hak istimewa yang sama di tingkat kolom tidak berpengaruh.
  - Jika pengguna memiliki hak istimewa tingkat tabel pada tabel, maka mencabut hak istimewa yang sama untuk satu atau lebih kolom tabel mengembalikan kesalahan. Sebagai gantinya, cabut hak istimewa di tingkat tabel.
  - Jika pengguna memiliki hak istimewa tingkat kolom, maka pemberian hak istimewa yang sama di tingkat tabel mengembalikan kesalahan.
  - Jika pengguna memiliki hak istimewa tingkat kolom, maka mencabut hak istimewa yang sama di tingkat tabel mencabut hak istimewa kolom dan tabel untuk semua kolom pada tabel.
- Anda tidak dapat memberikan hak istimewa tingkat kolom pada tampilan yang mengikat terlambat.
- Untuk membuat tampilan yang terwujud, Anda harus memiliki hak istimewa SELECT tingkat tabel pada tabel dasar. Bahkan jika Anda memiliki hak istimewa tingkat kolom pada kolom tertentu, Anda tidak dapat membuat tampilan terwujud hanya pada kolom tersebut. Namun, Anda dapat memberikan hak istimewa SELECT ke kolom tampilan yang terwujud, mirip dengan tampilan biasa.
- Untuk mencari pemberian hak istimewa tingkat kolom, gunakan tampilan PG_ATTRIBUTE_INFO.

Catatan penggunaan untuk memberikan izin ASSUMEROLE

Catatan penggunaan berikut berlaku untuk memberikan izin ASSUMEROLE di Amazon Redshift.

Anda menggunakan izin ASSUMEROLE untuk mengontrol izin akses peran IAM untuk pengguna database, peran, atau grup pada perintah seperti COPY, UNLOAD, EXTERNAL FUNCTION, atau CREATE MODEL. Setelah Anda memberikan izin ASSUMEROLE kepada pengguna, peran, atau grup untuk peran IAM, pengguna, peran, atau grup dapat mengambil peran itu saat menjalankan perintah. Izin ASSUMEROLE memungkinkan Anda untuk memberikan akses ke perintah yang sesuai sesuai kebutuhan.

Hanya superuser database yang dapat memberikan atau mencabut izin ASSUMEROLE untuk pengguna, peran, dan grup. Seorang superuser selalu mempertahankan izin ASSUMEROLE.

Untuk mengaktifkan penggunaan izin ASSUMEROLE untuk pengguna, peran, dan grup, superuser melakukan dua tindakan berikut:

- Jalankan pernyataan berikut sekali di cluster:

```
revoke assumerole on all from public for all;
```

- Berikan izin ASSUMEROLE kepada pengguna, peran, dan grup untuk perintah yang sesuai.

Anda dapat menentukan rantai peran dalam klausa ON saat memberikan izin ASSUMEROLE. Anda menggunakan koma untuk memisahkan peran dalam rantai peran, misalnya,`Role1,Role2,Role3`. Jika rantai peran ditentukan saat memberikan izin ASSUMEROLE, Anda harus menentukan rantai peran saat melakukan operasi yang diberikan oleh izin ASSUMEROLE. Anda tidak dapat menentukan peran individual dalam rantai peran saat melakukan operasi yang diberikan oleh izin ASSUMEROLE. Misalnya, jika pengguna, peran, atau grup diberikan rantai peran`Role1,Role2,Role3`, Anda tidak dapat menentukan hanya `Role1` untuk melakukan operasi.

Jika pengguna mencoba melakukan operasi COPY, UNLOAD, EXTERNAL FUNCTION, atau CREATE MODEL dan belum diberikan izin ASSUMEROLE, pesan yang mirip dengan berikut ini muncul.

```
ERROR:  User awsuser does not have ASSUMEROLE permission on IAM role
 "arn:aws:iam::123456789012:role/RoleA" for COPY
```

Untuk membuat daftar pengguna yang telah diberikan akses ke peran dan perintah IAM melalui izin ASSUMEROLE, lihat. MEMILIKI HAK ASSUMEROLE Untuk mencantumkan peran IAM dan izin perintah yang telah diberikan kepada pengguna yang Anda tentukan, lihatPG_GET_IAM_ROL_OLE_OLE_PENGGUNA. Untuk mencantumkan pengguna, peran, dan grup yang telah diberikan akses ke peran IAM yang Anda tentukan, lihatPG_DAPAT_PENERIMA HIBAH OLEH_IAM_ROLE.

Catatan penggunaan untuk memberikan izin pembelajaran mesin

Anda tidak dapat langsung memberikan atau mencabut izin yang terkait dengan fungsi ML. Fungsi ML milik model ML dan izin dikendalikan melalui model. Sebagai gantinya, Anda dapat memberikan izin yang terkait dengan model ML. Contoh berikut menunjukkan cara memberikan izin kepada semua pengguna untuk menjalankan fungsi ML yang terkait dengan model`customer_churn`.

```
GRANT EXECUTE ON MODEL customer_churn TO PUBLIC;
```

Anda juga dapat memberikan semua izin kepada pengguna untuk model ML`customer_churn`.

```
GRANT ALL on MODEL customer_churn TO ml_user;
```

Pember EXECUTE ian izin yang terkait dengan fungsi ML akan gagal jika ada fungsi ML dalam skema, bahkan jika fungsi ML itu sudah memiliki EXECUTE izin melaluiGRANT EXECUTE ON MODEL. Sebaiknya gunakan skema terpisah saat menggunakan CREATE MODEL perintah untuk menjaga fungsi ML dalam skema terpisah sendiri. Contoh berikut menunjukkan bagaimana melakukannya.

```
CREATE MODEL ml_schema.customer_churn
FROM customer_data
TARGET churn
FUNCTION ml_schema.customer_churn_prediction
IAM_ROLE default
SETTINGS (
 S3_BUCKET 'amzn-s3-demo-bucket'
);
```

Contoh

Contoh berikut memberikan hak istimewa SELECT pada tabel SALES kepada penggunafred.

```
grant select on table sales to fred;
```

Contoh berikut memberikan hak istimewa SELECT pada semua tabel dalam skema QA_TICKIT kepada pengguna. fred

```
grant select on all tables in schema qa_tickit to fred;
```

Contoh berikut memberikan semua hak istimewa skema pada skema QA_TICKIT ke grup pengguna QA_USERS. Hak istimewa skema adalah CREATE dan USAGE. USAGE memberi pengguna akses ke objek dalam skema, tetapi tidak memberikan hak istimewa seperti INSERT atau SELECT pada objek tersebut. Berikan hak istimewa pada setiap objek secara terpisah.

```
create group qa_users;
grant all on schema qa_tickit to group qa_users;
```

Contoh berikut memberikan semua hak istimewa pada tabel SALES dalam skema QA_TICKIT kepada semua pengguna dalam grup QA_USERS.

```
grant all on table qa_tickit.sales to group qa_users;
```

Contoh berikut memberikan semua hak istimewa pada tabel SALES dalam skema QA_TICKIT kepada semua pengguna dalam grup QA_USERS dan RO_USERS.

```
grant all on table qa_tickit.sales to group qa_users, group ro_users;
```

Contoh berikut memberikan hak istimewa DROP pada tabel SALES dalam skema QA_TICKIT kepada semua pengguna dalam grup QA_USERS.

```
grant drop on table qa_tickit.sales to group qa_users;>
```

Urutan perintah berikut menunjukkan bagaimana akses ke skema tidak memberikan hak istimewa pada tabel dalam skema.

```
create user schema_user in group qa_users password 'Abcd1234';
create schema qa_tickit;
create table qa_tickit.test (col1 int);
grant all on schema qa_tickit to schema_user;

set session authorization schema_user;
select current_user;


current_user
--------------
schema_user
(1 row)


select count(*) from qa_tickit.test;


ERROR: permission denied for relation test [SQL State=42501]


set session authorization dw_user;
grant select on table qa_tickit.test to schema_user;
set session authorization schema_user;
select count(*) from qa_tickit.test;


count
-------
```

```
0
(1 row)
```

Urutan perintah berikut menunjukkan bagaimana akses ke tampilan tidak menyiratkan akses ke tabel yang mendasarinya. Pengguna bernama VIEW_USER tidak dapat memilih dari tabel DATE, meskipun pengguna ini telah diberikan semua hak istimewa pada VIEW_DATE.

```
create user view_user password 'Abcd1234';
create view view_date as select * from date;
grant all on view_date to view_user;
set session authorization view_user;
select current_user;


current_user
--------------
view_user
(1 row)


select count(*) from view_date;


count
-------
365
(1 row)


select count(*) from date;


ERROR:  permission denied for relation date
```

Contoh berikut memberikan hak istimewa SELECT pada `cust_phone` kolom `cust_name` dan `cust_profile` tabel kepada penggunauser1.

```
grant select(cust_name, cust_phone) on cust_profile to user1;
```

Contoh berikut memberikan hak istimewa SELECT pada cust_phone kolom cust_name dan dan hak istimewa cust_contact_preference UPDATE pada kolom cust_profile tabel ke sales_group grup.

```
grant select(cust_name, cust_phone), update(cust_contact_preference) on cust_profile to
 group sales_group;
```

Contoh berikut menunjukkan penggunaan kata kunci ALL untuk memberikan hak istimewa SELECT dan UPDATE pada tiga kolom tabel cust_profile ke sales_admin grup.

```
grant ALL(cust_name, cust_phone,cust_contact_preference) on cust_profile to group
 sales_admin;
```

Contoh berikut memberikan hak istimewa SELECT pada cust_name kolom cust_profile_vw tampilan kepada user2 pengguna.

```
grant select(cust_name) on cust_profile_vw to user2;
```

Contoh pemberian akses ke datashares

Contoh berikut menunjukkan izin penggunaan berbagi data GRANT pada database atau skema tertentu yang dibuat dari datashare.

Dalam contoh berikut, admin sisi produsen memberikan izin USAGE pada salesshare datashare ke namespace yang ditentukan.

```
GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Dalam contoh berikut, admin sisi konsumen memberikan izin USAGE pada kesales_db. Bob

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

Dalam contoh berikut, admin sisi konsumen memberikan izin GRANT USAGE pada sales_schema skema ke peran tersebutAnalyst_role. sales_schemaadalah skema eksternal yang menunjuk ke sales_db.

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

Pada titik ini, Bob dan Analyst_role dapat mengakses semua objek database di sales_schema dansales_db.

Contoh berikut menunjukkan pemberian izin tingkat objek tambahan untuk objek dalam database bersama. Izin tambahan ini hanya diperlukan jika perintah CREATE DATABASE yang digunakan untuk membuat database bersama menggunakan klausa WITH PERMISSIONS. Jika perintah CREATE DATABASE tidak menggunakan WITH PERMISSIONS, pemberian USE pada database bersama memberikan akses penuh ke semua objek dalam database tersebut.

```
GRANT SELECT ON sales_db.sales_schema.tickit_sales_redshift to Bob;
```

Contoh pemberian izin bercakupan

Contoh berikut memberikan penggunaan untuk semua skema saat ini dan masa depan dalam Sales_db database ke Sales peran.

```
GRANT USAGE FOR SCHEMAS IN DATABASE Sales_db TO ROLE Sales;
```

Contoh berikut memberikan izin SELECT untuk semua tabel saat ini dan yang akan datang dalam Sales_db database kepada penggunaalice, dan juga memberikan alice izin untuk memberikan izin cakupan pada tabel Sales_db ke pengguna lain.

```
GRANT SELECT FOR TABLES IN DATABASE Sales_db TO alice WITH GRANT OPTION;
```

Contoh berikut memberikan izin EXECUTE untuk fungsi dalam Sales_schema skema kepada penggunabob.

```
GRANT EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema TO bob;
```

Contoh berikut memberikan semua izin untuk semua tabel dalam ShareSchema skema ShareDb database ke Sales peran. Saat menentukan skema, Anda dapat menentukan database skema menggunakan format database.schema dua bagian.

```
GRANT ALL FOR TABLES IN SCHEMA ShareDb.ShareSchema TO ROLE Sales;
```

Contoh berikut adalah sama dengan yang sebelumnya. Anda dapat menentukan database menggunakan DATABASE kata kunci alih-alih menggunakan format dua bagian.

```
GRANT ALL FOR TABLES IN SCHEMA ShareSchema DATABASE ShareDb TO ROLE Sales;
```

Contoh pemberian hak istimewa ASSUMEROLE

Berikut ini adalah contoh pemberian hak istimewa ASSUMEROLE.

Contoh berikut menunjukkan pernyataan REVOKE bahwa superuser berjalan sekali pada cluster untuk mengaktifkan penggunaan hak istimewa ASSUMEROLE untuk pengguna dan grup. Kemudian, superuser memberikan hak istimewa ASSUMEROLE kepada pengguna dan grup untuk perintah yang sesuai. Untuk informasi tentang mengaktifkan penggunaan hak istimewa ASSUMEROLE untuk pengguna dan grup, lihat. Catatan penggunaan untuk memberikan izin ASSUMEROLE

```
revoke assumerole on all from public for all;
```

Contoh berikut memberikan hak istimewa ASSUMEROLE kepada pengguna `reg_user1` untuk peran IAM `Redshift-S3-Read` untuk melakukan operasi COPY.

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-S3-Read'
to reg_user1 for copy;
```

Contoh berikut memberikan hak istimewa ASSUMEROLE kepada pengguna `reg_user1` untuk rantai peran IAM`RoleA`, `RoleB` untuk melakukan operasi UNLOAD.

```
grant assumerole
on 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB'
to reg_user1
for unload;
```

Berikut ini adalah contoh perintah UNLOAD menggunakan rantai `RoleA` peran IAM,. `RoleB`

```
unload ('select * from venue limit 10')
to 's3://companyb/redshift/venue_pipe_'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

Contoh berikut memberikan hak istimewa ASSUMEROLE kepada pengguna `reg_user1` untuk peran IAM `Redshift-Exfunc` untuk membuat fungsi eksternal.

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-Exfunc'
```

```
to reg_user1 for external function;
```

Contoh berikut memberikan hak istimewa ASSUMEROLE kepada pengguna `reg_user1` untuk peran IAM `Redshift-model` untuk membuat model pembelajaran mesin.

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-ML'
to reg_user1 for create model;
```

Contoh pemberian hak istimewa ROLE

Contoh berikut memberikan sample_role1 ke user1.

```
CREATE ROLE sample_role1;
GRANT ROLE sample_role1 TO user1;
```

Contoh berikut memberikan sample_role1 kepada user1 dengan OPSI WITH ADMIN, menetapkan sesi saat ini untuk user1, dan user1 memberikan sample_role1 ke user2.

```
GRANT ROLE sample_role1 TO user1 WITH ADMIN OPTION;
SET SESSION AUTHORIZATION user1;
GRANT ROLE sample_role1 TO user2;
```

Contoh berikut memberikan sample_role1 ke sample_role2.

```
GRANT ROLE sample_role1 TO ROLE sample_role2;
```

Contoh berikut memberikan sample_role2 ke sample_role3 dan sample_role4. Kemudian mencoba memberikan sample_role3 ke sample_role1.

```
GRANT ROLE sample_role2 TO ROLE sample_role3;
GRANT ROLE sample_role3 TO ROLE sample_role2;
ERROR: cannot grant this role, a circular dependency was detected between these roles
```

Contoh berikut memberikan hak istimewa sistem CREATE USER ke sample_role1.

```
GRANT CREATE USER TO ROLE sample_role1;
```

Contoh berikut memberikan peran yang ditentukan sistem kepada user1`sys:dba`.

```
GRANT ROLE sys:dba TO user1;
```

Contoh berikut mencoba memberikan sample_role3 dalam ketergantungan melingkar ke sample_role2.

```
CREATE ROLE sample_role3;
GRANT ROLE sample_role2 TO ROLE sample_role3;
GRANT ROLE sample_role3 TO ROLE sample_role2; -- fail
ERROR:  cannot grant this role, a circular dependency was detected between these roles
```

INSERT

Topik

- Sintaksis
- Parameter
- Catatan penggunaan
- MASUKKAN contoh

Menyisipkan baris baru ke dalam tabel. Anda dapat menyisipkan satu baris dengan sintaks VALUES, beberapa baris dengan sintaks VALUES, atau satu atau lebih baris yang ditentukan oleh hasil kueri (INSERT INTO... SELECT).

> **Note**
>
> Kami sangat menyarankan Anda untuk menggunakan MENYALIN perintah untuk memuat data dalam jumlah besar. Menggunakan pernyataan INSERT individual untuk mengisi tabel mungkin sangat lambat. Atau, jika data Anda sudah ada di tabel database Amazon Redshift lainnya, gunakan INSERT INTO SELECT atau CREATE TABLE AS untuk meningkatkan kinerja. Untuk informasi selengkapnya tentang menggunakan perintah COPY untuk memuat tabel, lihatMemuat data di Amazon Redshift.

> **Note**
>
> Ukuran maksimum untuk satu pernyataan SQL adalah 16 MB.

Sintaksis

```
INSERT INTO table_name [ ( column [, ...] ) ]
{DEFAULT VALUES |
VALUES ( { expression | DEFAULT } [, ...] )
[, ( { expression | DEFAULT } [, ...] )
[, ...] ] |
query }
```

Parameter

table_name

Tabel sementara atau persisten. Hanya pemilik tabel atau pengguna dengan hak istimewa INSERT pada tabel yang dapat menyisipkan baris. Jika Anda menggunakan klausa kueri untuk menyisipkan baris, Anda harus memiliki hak istimewa SELECT pada tabel yang disebutkan dalam kueri.

> **Note**
>
> Gunakan INSERT (tabel eksternal) untuk menyisipkan hasil kueri SELECT ke tabel yang ada di katalog eksternal. Untuk informasi selengkapnya, lihat INSERT (tabel eksternal).

kolom

Anda dapat memasukkan nilai ke dalam satu atau lebih kolom tabel. Anda dapat mencantumkan nama kolom target dalam urutan apa pun. Jika Anda tidak menentukan daftar kolom, nilai yang akan dimasukkan harus sesuai dengan kolom tabel dalam urutan di mana mereka dideklarasikan dalam pernyataan CREATE TABLE. Jika jumlah nilai yang akan dimasukkan kurang dari jumlah kolom dalam tabel, n kolom pertama dimuat.

Baik nilai default yang dinyatakan atau nilai null dimuat ke kolom apa pun yang tidak terdaftar (secara implisit atau eksplisit) dalam pernyataan INSERT.

DEFAULT VALUES

Jika kolom dalam tabel diberi nilai default saat tabel dibuat, gunakan kata kunci ini untuk menyisipkan baris yang seluruhnya terdiri dari nilai default. Jika salah satu kolom tidak memiliki nilai default, nol dimasukkan ke dalam kolom tersebut. Jika salah satu kolom dinyatakan NOT NULL, pernyataan INSERT mengembalikan kesalahan.

NILAI

Gunakan kata kunci ini untuk menyisipkan satu atau lebih baris, setiap baris terdiri dari satu atau lebih nilai. Daftar VALUES untuk setiap baris harus sejajar dengan daftar kolom. Untuk menyisipkan beberapa baris, gunakan pembatas koma di antara setiap daftar ekspresi. Jangan ulangi kata kunci VALUES. Semua daftar VALUES untuk pernyataan INSERT beberapa baris harus berisi jumlah nilai yang sama.

ekspresi

Nilai tunggal atau ekspresi yang mengevaluasi ke nilai tunggal. Setiap nilai harus kompatibel dengan tipe data kolom tempat ia dimasukkan. Jika memungkinkan, nilai yang tipe datanya tidak cocok dengan tipe data yang dideklarasikan kolom secara otomatis dikonversi ke tipe data yang kompatibel. Contoh:

- Nilai desimal `1.1` dimasukkan ke dalam kolom INT sebagai`1`.
- Nilai desimal `100.8976` dimasukkan ke dalam kolom DEC (5,2) sebagai`100.90`.

Anda dapat secara eksplisit mengonversi nilai ke tipe data yang kompatibel dengan menyertakan sintaks tipe cast dalam ekspresi. Misalnya, jika kolom COL1 pada tabel T1 adalah kolom CHAR (3):

```
insert into t1(col1) values('Incomplete'::char(3));
```

Pernyataan ini memasukkan nilai `Inc` ke dalam kolom.

Untuk pernyataan INSERT VALUES baris tunggal, Anda dapat menggunakan subkueri skalar sebagai ekspresi. Hasil subquery dimasukkan ke kolom yang sesuai.

> **Note**
>
> Subquery tidak didukung sebagai ekspresi untuk pernyataan INSERT VALUES beberapa baris.

DEFAULT

Gunakan kata kunci ini untuk menyisipkan nilai default untuk kolom, seperti yang didefinisikan saat tabel dibuat. Jika tidak ada nilai default untuk kolom, null dimasukkan. Anda tidak dapat menyisipkan nilai default ke kolom yang memiliki batasan NOT NULL jika kolom tersebut tidak memiliki nilai default eksplisit yang ditetapkan padanya dalam pernyataan CREATE TABLE.

query

Masukkan satu atau lebih baris ke dalam tabel dengan mendefinisikan kueri apa pun. Semua baris yang dihasilkan query dimasukkan ke dalam tabel. Kueri harus mengembalikan daftar kolom yang kompatibel dengan kolom dalam tabel, tetapi nama kolom tidak harus cocok.

Catatan penggunaan

> **Note**
>
> Kami sangat menyarankan Anda untuk menggunakan MENYALIN perintah untuk memuat data dalam jumlah besar. Menggunakan pernyataan INSERT individual untuk mengisi tabel mungkin sangat lambat. Atau, jika data Anda sudah ada di tabel database Amazon Redshift lainnya, gunakan INSERT INTO SELECT atau CREATE TABLE AS untuk meningkatkan kinerja. Untuk informasi selengkapnya tentang menggunakan perintah COPY untuk memuat tabel, lihatMemuat data di Amazon Redshift.

Format data untuk nilai yang disisipkan harus sesuai dengan format data yang ditentukan oleh definisi CREATE TABLE.

Setelah memasukkan sejumlah besar baris baru ke dalam tabel:

- Vakum meja untuk merebut kembali ruang penyimpanan dan mengurutkan ulang baris.
- Analisis tabel untuk memperbarui statistik untuk perencana kueri.

Ketika nilai dimasukkan ke dalam kolom DECIMAL dan melebihi skala yang ditentukan, nilai yang dimuat dibulatkan sesuai. Misalnya, ketika nilai `20.259` dimasukkan ke dalam kolom DECIMAL (8,2), nilai yang disimpan adalah`20.26`.

Anda dapat menyisipkan ke kolom GENERATED BY DEFAULT AS IDENTITY. Anda dapat memperbarui kolom yang didefinisikan sebagai GENERATED BY DEFAULT AS IDENTITY dengan nilai-nilai yang Anda berikan. Untuk informasi selengkapnya, lihat GENERATED BY DEFAULT AS IDENTITY.

MASUKKAN contoh

Tabel CATEGORY dalam database TICKIT berisi baris berikut:

```
 catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
     1 | Sports   | MLB       | Major League Baseball
     2 | Sports   | NHL       | National Hockey League
     3 | Sports   | NFL       | National Football League
     4 | Sports   | NBA       | National Basketball Association
     5 | Sports   | MLS       | Major League Soccer
     6 | Shows    | Musicals  | Musical theatre
     7 | Shows    | Plays     | All non-musical theatre
     8 | Shows    | Opera     | All opera and light opera
     9 | Concerts | Pop       | All rock and pop music concerts
    10 | Concerts | Jazz      | All jazz singers and bands
    11 | Concerts | Classical | All symphony, concerto, and choir concerts
(11 rows)
```

Buat tabel CATEGORY_STAGE dengan skema yang mirip dengan tabel CATEGORY tetapi tentukan nilai default untuk kolom:

```
create table category_stage
(catid smallint default 0,
catgroup varchar(10) default 'General',
catname varchar(10) default 'General',
catdesc varchar(50) default 'General');
```

Pernyataan INSERT berikut memilih semua baris dari tabel CATEGORY dan menyisipkannya ke dalam tabel CATEGORY_STAGE.

```
insert into category_stage
(select * from category);
```

Tanda kurung di sekitar kueri bersifat opsional.

Perintah ini menyisipkan baris baru ke dalam tabel CATEGORY_STAGE dengan nilai yang ditentukan untuk setiap kolom secara berurutan:

```
insert into category_stage values
(12, 'Concerts', 'Comedy', 'All stand-up comedy performances');
```

Anda juga dapat menyisipkan baris baru yang menggabungkan nilai tertentu dan nilai default:

```
insert into category_stage values
```

```
(13, 'Concerts', 'Other', default);
```

Jalankan kueri berikut untuk mengembalikan baris yang disisipkan:

```
select * from category_stage
where catid in(12,13) order by 1;

 catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
    12 | Concerts | Comedy  | All stand-up comedy performances
    13 | Concerts | Other   | General
(2 rows)
```

Contoh berikut menunjukkan beberapa pernyataan INSERT VALUES beberapa baris. Contoh pertama menyisipkan nilai CATID tertentu untuk dua baris dan nilai default untuk kolom lain di kedua baris.

```
insert into category_stage values
(14, default, default, default),
(15, default, default, default);

select * from category_stage where catid in(14,15) order by 1;
 catid | catgroup | catname | catdesc
-------+----------+---------+---------
    14 | General  | General | General
    15 | General  | General | General
(2 rows)
```

Contoh berikutnya menyisipkan tiga baris dengan berbagai kombinasi nilai spesifik dan default:

```
insert into category_stage values
(default, default, default, default),
(20, default, 'Country', default),
(21, 'Concerts', 'Rock', default);

select * from category_stage where catid in(0,20,21) order by 1;
 catid | catgroup | catname | catdesc
-------+----------+---------+---------
     0 | General  | General | General
    20 | General  | Country | General
    21 | Concerts | Rock    | General
```

```
(3 rows)
```

Kumpulan VALUES pertama dalam contoh ini menghasilkan hasil yang sama seperti menentukan NILAI DEFAULT untuk pernyataan INSERT baris tunggal.

Contoh berikut menunjukkan perilaku INSERT ketika tabel memiliki kolom IDENTITY. Pertama, buat versi baru dari tabel CATEGORY, lalu masukkan baris ke dalamnya dari CATEGORY:

```
create table category_ident
(catid int identity not null,
catgroup varchar(10) default 'General',
catname varchar(10) default 'General',
catdesc varchar(50) default 'General');


insert into category_ident(catgroup,catname,catdesc)
select catgroup,catname,catdesc from category;
```

Perhatikan bahwa Anda tidak dapat memasukkan nilai integer tertentu ke kolom IDENTITY CATID. Nilai kolom IDENTITY dibuat secara otomatis.

Contoh berikut menunjukkan bahwa subquery tidak dapat digunakan sebagai ekspresi dalam pernyataan INSERT VALUES multi-baris:

```
insert into category(catid) values
((select max(catid)+1 from category)),
((select max(catid)+2 from category));

ERROR: can't use subqueries in multi-row VALUES
```

Contoh berikut menunjukkan sisipan ke tabel sementara diisi dengan data dari `venue` tabel menggunakan `WITH SELECT` klausa. Untuk informasi selengkapnya tentang tabel `venue`, lihat Database sampel.

Pertama, buat tabel sementara`#venuetemp`.

```
CREATE TABLE #venuetemp AS SELECT * FROM venue;
```

Buat daftar baris dalam `#venuetemp` tabel.

```
SELECT * FROM #venuetemp ORDER BY venueid;

venueid | venuename                | venuecity  | venuestate| venueseats
--------+--------------------------+------------+-----------+------------
1       Toyota Park                Bridgeview   IL          0
2       Columbus Crew Stadium      Columbus     OH          0
3       RFK Stadium                Washington   DC          0
4       CommunityAmerica Ballpark  Kansas City  KS          0
5       Gillette Stadium           Foxborough   MA          68756
...
```

Masukkan 10 baris duplikat dalam #venuetemp tabel menggunakan WITH SELECT klausa.

```
INSERT INTO #venuetemp (WITH venuecopy AS (SELECT * FROM venue) SELECT * FROM venuecopy
 ORDER BY 1 LIMIT 10);
```

Buat daftar baris dalam #venuetemp tabel.

```
SELECT * FROM #venuetemp ORDER BY venueid;

venueid | venuename                | venuecity  | venuestate| venueseats
--------+--------------------------+------------+-----------+------------
1       Toyota Park                Bridgeview   IL          0
1       Toyota Park                Bridgeview   IL          0
2       Columbus Crew Stadium      Columbus     OH          0
2       Columbus Crew Stadium      Columbus     OH          0
3       RFK Stadium                Washington   DC          0
3       RFK Stadium                Washington   DC          0
4       CommunityAmerica Ballpark  Kansas City  KS          0
4       CommunityAmerica Ballpark  Kansas City  KS          0
5       Gillette Stadium           Foxborough   MA          68756
5       Gillette Stadium           Foxborough   MA          68756
...
```

INSERT (tabel eksternal)

Menyisipkan hasil kueri SELECT ke tabel eksternal yang ada di katalog eksternal seperti for AWS Glue, AWS Lake Formation, atau metastore Apache Hive. Gunakan peran AWS Identity and Access Management (IAM) yang sama yang digunakan untuk perintah CREATE EXTERNAL SCHEMA untuk berinteraksi dengan katalog eksternal dan Amazon S3.

Untuk tabel yang tidak dipartisi, perintah INSERT (tabel eksternal) menulis data ke lokasi Amazon S3 yang ditentukan dalam tabel, berdasarkan properti tabel dan format file yang ditentukan.

Untuk tabel yang dipartisi, INSERT (tabel eksternal) menulis data ke lokasi Amazon S3 sesuai dengan kunci partisi yang ditentukan dalam tabel. Ini juga secara otomatis mendaftarkan partisi baru di katalog eksternal setelah operasi INSERT selesai.

Anda tidak dapat menjalankan INSERT (tabel eksternal) dalam blok transaksi (BEGIN... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.

Sintaksis

```
INSERT INTO external_schema.table_name
{ select_statement }
```

Parameter

nama_tabel skema eksternal_

Nama skema eksternal yang ada dan tabel eksternal target untuk dimasukkan.

select_statement

Pernyataan yang menyisipkan satu atau lebih baris ke tabel eksternal dengan mendefinisikan kueri apa pun. Semua baris yang dihasilkan kueri ditulis ke Amazon S3 baik dalam format teks atau parket berdasarkan definisi tabel. Kueri harus mengembalikan daftar kolom yang kompatibel dengan tipe data kolom di tabel eksternal. Namun, nama kolom tidak harus cocok.

Catatan penggunaan

Jumlah kolom dalam kueri SELECT harus sama dengan jumlah kolom data dan kolom partisi. Lokasi dan tipe data dari setiap kolom data harus sesuai dengan tabel eksternal. Lokasi kolom partisi harus berada di akhir query SELECT, dalam urutan yang sama dengan yang didefinisikan dalam perintah CREATE EXTERNAL TABLE. Nama kolom tidak harus cocok.

Dalam beberapa kasus, Anda mungkin ingin menjalankan perintah INSERT (tabel eksternal) pada Katalog AWS Glue Data atau metastore Hive. Dalam kasus AWS Glue, peran IAM yang digunakan untuk membuat skema eksternal harus memiliki izin baca dan tulis di Amazon S3 dan AWS Glue. Jika Anda menggunakan AWS Lake Formation katalog, peran IAM ini menjadi pemilik tabel Formasi Danau yang baru. Peran IAM ini setidaknya harus memiliki izin berikut:

- PILIH, SISIPKAN, PERBARUI izin pada tabel eksternal
- Izin lokasi data pada jalur Amazon S3 dari tabel eksternal

Untuk memastikan bahwa nama file unik, Amazon Redshift menggunakan format berikut untuk nama setiap file yang diunggah ke Amazon S3 secara default.

*<date>*_*<time>*_*<microseconds>*_*<query_id>*_*<slice-number>*_part_*<part-number>*.*<format>*.

Contohnya adalah `20200303_004509_810669_1007_0001_part_00.parquet`.

Pertimbangkan hal berikut saat menjalankan perintah INSERT (tabel eksternal):

- Tabel eksternal yang memiliki format selain PARQUET atau TEXTFILE tidak didukung.
- Perintah ini mendukung properti tabel yang ada seperti 'write.parallel', 'write.maxfilesize.mb', 'compression_type', dan 'serialization.null.format'. Untuk memperbarui nilai-nilai tersebut, jalankan perintah ALTER TABLE SET TABLE PROPERTIES.
- Properti tabel 'numRows' diperbarui secara otomatis menjelang akhir operasi INSERT. Properti tabel harus ditentukan atau ditambahkan ke tabel jika tidak dibuat oleh operasi CREATE EXTERNAL TABLE AS.
- Klausa LIMIT tidak didukung dalam kueri SELECT luar. Sebagai gantinya, gunakan klausa LIMIT bersarang.
- Anda dapat menggunakan STL_UNLOAD_LOG tabel untuk melacak file yang ditulis ke Amazon S3 oleh setiap operasi INSERT (tabel eksternal).
- Amazon Redshift hanya mendukung enkripsi standar Amazon S3 untuk INSERT (tabel eksternal).

Contoh INSERT (tabel eksternal)

Contoh berikut menyisipkan hasil pernyataan SELECT ke tabel eksternal.

```
INSERT INTO spectrum.lineitem
SELECT * FROM local_lineitem;
```

Contoh berikut menyisipkan hasil pernyataan SELECT ke dalam tabel eksternal yang dipartisi menggunakan partisi statis. Kolom partisi di-hardcode dalam pernyataan SELECT. Kolom partisi harus berada di akhir kueri.

```
INSERT INTO spectrum.customer
SELECT name, age, gender, 'May', 28 FROM local_customer;
```

Contoh berikut menyisipkan hasil pernyataan SELECT ke dalam tabel eksternal yang dipartisi menggunakan partisi dinamis. Kolom partisi tidak di-hardcode. Data secara otomatis ditambahkan ke folder partisi yang ada, atau ke folder baru jika partisi baru ditambahkan.

```
INSERT INTO spectrum.customer
SELECT name, age, gender, month, day FROM local_customer;
```

KUNCI

Membatasi akses ke tabel database. Perintah ini hanya berarti ketika dijalankan di dalam blok transaksi.

Perintah LOCK memperoleh kunci tingkat tabel dalam mode “AKSES EKSKLUSIF”, menunggu jika perlu untuk melepaskan kunci yang saling bertentangan. Mengunci tabel secara eksplisit dengan cara ini menyebabkan pembacaan dan penulisan pada tabel menunggu ketika mereka dicoba dari transaksi atau sesi lain. Kunci tabel eksplisit yang dibuat oleh satu pengguna sementara mencegah pengguna lain memilih data dari tabel itu atau memuat data ke dalamnya. Kunci dilepaskan saat transaksi yang berisi perintah LOCK selesai.

Kunci tabel yang kurang ketat diperoleh secara implisit oleh perintah yang merujuk pada tabel, seperti operasi tulis. Misalnya, jika pengguna mencoba membaca data dari tabel sementara pengguna lain memperbarui tabel, data yang dibaca akan menjadi snapshot dari data yang telah dikomit. (Dalam beberapa kasus, kueri akan berhenti jika melanggar aturan isolasi yang dapat diserialisasi.) Lihat Mengelola operasi penulisan bersamaan.

Beberapa operasi DDL, seperti DROP TABLE dan TRUNCATE, membuat kunci eksklusif. Operasi ini mencegah pembacaan data.

Jika terjadi konflik kunci, Amazon Redshift menampilkan pesan kesalahan untuk mengingatkan pengguna yang memulai transaksi dalam konflik. Transaksi yang menerima konflik kunci dihentikan. Setiap kali konflik kunci terjadi, Amazon Redshift menulis entri ke STL_TR_KONFLIK tabel.

Sintaksis

```
LOCK [ TABLE ] table_name [, ...]
```

Parameter

TABEL

Kata kunci opsional.

table_name

Nama tabel yang akan dikunci. Anda dapat mengunci lebih dari satu tabel dengan menggunakan daftar nama tabel yang dipisahkan koma. Anda tidak dapat mengunci tampilan.

Contoh

```
begin;

lock event, sales;

...
```

MERGE

Secara kondisional menggabungkan baris dari tabel sumber ke tabel target. Secara tradisional, ini hanya dapat dicapai dengan menggunakan beberapa pernyataan menyisipkan, memperbarui atau menghapus secara terpisah. Untuk informasi selengkapnya tentang operasi yang dapat Anda gabungkan dengan MERGE, lihat UP DATE, DEL ETE, dan IN SERT.

Sintaksis

```
MERGE INTO target_table
USING source_table [ [ AS ] alias ]
ON match_condition
[ WHEN MATCHED THEN { UPDATE SET col_name = { expr } [,...] | DELETE }
WHEN NOT MATCHED THEN INSERT [ ( col_name [,...] ) ] VALUES ( { expr } [, ...] ) |
REMOVE DUPLICATES ]
```

Parameter

target_table

Tabel sementara atau permanen yang digabungkan dengan pernyataan MERGE.

tabel_sumber

Tabel sementara atau permanen menyediakan baris untuk digabungkan menjadi target_table. source_table juga bisa menjadi tabel Spectrum.

alias

Nama alternatif sementara untuk source _table.

Parameter ini bersifat opsional. Alias sebelumnya dengan AS juga opsional.

kondisi kecocokan

Menentukan predikat yang sama antara kolom tabel sumber dan kolom tabel target yang digunakan untuk menentukan apakah baris di source_table dapat dicocok kan dengan baris di target_table. Jika kondisi terpenuhi, MERGE menjalankan matched_clauze untuk baris tersebut. Jika tidak, MERGE menjalankan not_matched_clauza untuk baris itu.

SAAT DICOCOKKAN

Menentukan tindakan yang akan dijalankan ketika kondisi kecocokan antara baris sumber dan baris target dievaluasi menjadi True. Anda dapat menentukan tindakan UPDATE atau DELETE.

UPDATE

Memperbarui baris yang cocok di target_table. Hanya nilai dalam col_name yang Anda tentukan yang diperbarui.

DELETE

Menghapus baris yang cocok di target _table.

KETIKA TIDAK COCOK

Menentukan tindakan yang akan dijalankan saat kondisi kecocokan dievaluasi ke False atau Unknown. Anda hanya dapat menentukan tindakan menyisipkan INSERT untuk klausa ini.

INSERT

Menyisip kan ke baris target_table dari source_table yang tidak cocok dengan baris apa pun di target _table, sesuai dengan match_condition. Target col_name dapat dicantumkan dalam urutan apa pun. Jika Anda tidak memberikan nilai col_name apa pun, urutan default adalah semua kolom tabel dalam urutan yang dideklarasikan.

col_name

Satu atau beberapa nama kolom yang ingin Anda ubah. Jangan sertakan nama tabel saat menentukan kolom target.

expres

Ekspresi yang mendefinisikan nilai baru untuk col_name.

MENGHAPUS DUPLIKAT

Menentukan bahwa perintah MERGE berjalan dalam mode yang disederhanakan. Mode yang disederhanakan memiliki persyaratan sebagai berikut:

- target_table dan source_table harus memiliki jumlah kolom yang sama, jenis kolom yang kompatibel, dan urutan kolom yang sama.
- Hilangkan klausa WHEN dan klausa UPDATE dan INSERT dari perintah MERGE Anda.
- Gunakan klausa REMOVE DUPLICATES dalam perintah MERGE Anda.

Dalam mode yang disederhanakan, MERGE melakukan hal berikut:

- Baris di target_table yang memiliki kecocokan di source_table diperbarui agar sesuai dengan nilai di source_table.
- Baris di source_table yang tidak memiliki kecocokan di target_table dimasukkan ke dalam target_table.
- Ketika beberapa baris di target_table cocok dengan baris yang sama di source_table, baris duplikat dihapus. Amazon Redshift menyimpan satu baris dan memperbaruinya. Baris duplikat yang tidak cocok dengan baris di source_table tetap tidak berubah.

Menggunakan REMOVE DUPLICATES memberikan kinerja yang lebih baik daripada menggunakan WHEN MATCHED dan WHEN NOT MATCHED. Sebaiknya gunakan REMOVE DUPLICATES jika target_table dan source_table kompatibel dan Anda tidak perlu menyimpan baris duplikat di target_table.

Catatan penggunaan

- Untuk menjalankan pernyataan MERGE, Anda harus menjadi pemilik source_table dan target_table, atau memiliki izin SELECT untuk tabel tersebut. Selain itu, Anda harus memiliki izin UPDATE, DELETE, dan INSERT untuk target_table tergantung pada operasi yang disertakan dalam pernyataan MERGE Anda.
- target_table tidak bisa berupa tabel sistem, tabel katalog, atau tabel eksternal.
- source_table dan target_table tidak bisa menjadi tabel yang sama.
- Anda tidak dapat menggunakan klausa WITH dalam pernyataan MERGE.

- Baris di target_table tidak dapat mencocokkan beberapa baris di source_table.

  Pertimbangkan contoh berikut:

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (1, 'Bob'), (2, 'John');
INSERT INTO source VALUES (1, 'Tony'), (1, 'Alice'), (3, 'Bill');

MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN UPDATE SET id = source.id, name = source.name
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);
ERROR: Found multiple matches to update the same tuple.

MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN DELETE
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);
ERROR: Found multiple matches to update the same tuple.
```

  Dalam kedua pernyataan MERGE, operasi gagal karena ada beberapa baris dalam `source` tabel dengan nilai ID1.

- match_condition dan expr tidak dapat mereferensikan sebagian kolom tipe SUPER. Misalnya, jika objek tipe SUPER Anda adalah array atau struktur, Anda tidak dapat menggunakan elemen individual dari kolom itu untuk match_condition atau expr, tetapi Anda dapat menggunakan seluruh kolom.

  Pertimbangkan contoh berikut:

```
CREATE TABLE IF NOT EXISTS target (key INT, value SUPER);
CREATE TABLE IF NOT EXISTS source (key INT, value SUPER);

INSERT INTO target VALUES (1, JSON_PARSE('{"key": 88}'));
INSERT INTO source VALUES (1, ARRAY(1, 'John')), (2, ARRAY(2, 'Bill'));

MERGE INTO target USING source ON target.key = source.key
WHEN matched THEN UPDATE SET value = source.value[0]
WHEN NOT matched THEN INSERT VALUES (source.key, source.value[0]);
ERROR: Partial reference of SUPER column is not supported in MERGE statement.
```

  Untuk informasi selengkapnya tentang tipe SUPER, lihat tipe SUPER.

- Jika source_table besar, mendefinisikan kolom gabungan dari target_table dan source_table sebagai kunci distribusi dapat meningkatkan kinerja.
- Untuk menggunakan klausa REMOVE DUPLICATES, Anda memerlukan izin SELECT, INSERT, dan DELETE untuk target _table.
- source_table dapat berupa tampilan atau subquery. Berikut ini adalah contoh pernyataan MERGE di mana source_table adalah subquery yang menghapus baris duplikat.

```
MERGE INTO target
USING (SELECT id, name FROM source GROUP BY 1, 2) as my_source
ON target.id = my_source.id
WHEN MATCHED THEN UPDATE SET id = my_source.id, name = my_source.name
WHEN NOT MATCHED THEN INSERT VALUES (my_source.id, my_source.name);
```

- Target tidak dapat menjadi sumber data dari subkueri apa pun dari pernyataan MERGE yang sama. Misalnya, perintah SQL berikut mengembalikan kesalahan seperti ERROR: Sumber view/ subquery dalam pernyataan Merge tidak dapat mereferensikan tabel target. karena referensi subquery al `target` ih-alih`source`.

```
MERGE INTO target
USING (SELECT id, name FROM target GROUP BY 1, 2) as my_source
ON target.id = my_source.id
WHEN MATCHED THEN UPDATE SET id = my_source.id, name = my_source.name
WHEN NOT MATCHED THEN INSERT VALUES (my_source.id, my_source.name);
```

Contoh

Contoh berikut membuat dua tabel, lalu menjalankan operasi MERGE pada mereka, memperbarui baris yang cocok di tabel target dan memasukkan baris yang tidak cocok. Kemudian ia memasukkan nilai lain ke dalam tabel sumber dan menjalankan operasi MERGE lain, kali ini menghapus baris yang cocok dan memasukkan baris baru dari tabel sumber.

Pertama buat dan isi tabel sumber dan target.

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (101, 'Bob'), (102, 'John'), (103, 'Susan');
INSERT INTO source VALUES (102, 'Tony'), (103, 'Alice'), (104, 'Bill');
```

```
SELECT * FROM target;
 id  |    name
-----+------------
 101 | Bob
 102 | John
 103 | Susan
(3 rows)

SELECT * FROM source;
 id  |    name
-----+------------
 102 | Tony
 103 | Alice
 104 | Bill
(3 rows)
```

Selanjutnya, gabungkan tabel sumber ke tabel target, perbarui tabel target dengan baris yang cocok dan masukkan baris dari tabel sumber yang tidak cocok.

```
MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN UPDATE SET id = source.id, name = source.name
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);

SELECT * FROM target;
 id  |    name
-----+------------
 101 | Bob
 102 | Tony
 103 | Alice
 104 | Bill
(4 rows)
```

Perhatikan bahwa baris dengan nilai id 102 dan 103 diperbarui agar sesuai dengan nilai nama dari tabel target. Juga, baris baru dengan nilai id 104 dan nilai nama Bill dimasukkan ke dalam tabel target.

Selanjutnya, masukkan baris baru ke tabel sumber.

```
INSERT INTO source VALUES (105, 'David');

SELECT * FROM source;
 id  |    name
```

```
-----+------------
 102 | Tony
 103 | Alice
 104 | Bill
 105 | David
(4 rows)
```

Terakhir, jalankan operasi penggabungan menghapus baris yang cocok di tabel target, dan memasukkan baris yang tidak cocok.

```
MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN DELETE
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);

SELECT * FROM target;
 id  |    name
-----+------------
 101 | Bob
 105 | David
(2 rows)
```

Baris dengan nilai id 102, 103, dan 104 dihapus dari tabel target, dan baris baru dengan nilai id 105 dan nilai nama David dimasukkan ke dalam tabel target.

Contoh berikut menunjukkan sintaks sederhana dari perintah MERGE yang menggunakan klausa REMOVE DUPLICATES.

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (30, 'Tony'), (11, 'Alice'), (23, 'Bill');
INSERT INTO source VALUES (23, 'David'), (22, 'Clarence');

MERGE INTO target USING source ON target.id = source.id REMOVE DUPLICATES;

SELECT * FROM target;
id | name
---+------------
30 | Tony
11 | Alice
23 | David
22 | Clarence
```

```
(4 rows)
```

Contoh berikut menunjukkan sintaks sederhana dari perintah MERGE yang menggunakan klausa REMOVE DUPLICATES, menghapus baris duplikat dari target_table jika mereka memiliki baris yang cocok di source_table.

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (30, 'Tony'), (30, 'Daisy'), (11, 'Alice'), (23, 'Bill'),
 (23, 'Nikki');
INSERT INTO source VALUES (23, 'David'), (22, 'Clarence');

MERGE INTO target USING source ON target.id = source.id REMOVE DUPLICATES;

SELECT * FROM target;
id | name
---+------------
30 | Tony
30 | Daisy
11 | Alice
23 | David
22 | Clarence
(5 rows)
```

Setelah MERGE berjalan, hanya ada satu baris dengan nilai ID 23 di target_table. Karena tidak ada baris di source_table dengan nilai ID 30, dua baris duplikat dengan nilai ID 30 tetap berada di target_table.

Lihat juga

INSERT, UPDATE, DELETE

MEMPERSIAPKAN

Siapkan pernyataan untuk eksekusi.

PREPARE membuat pernyataan yang disiapkan. Ketika pernyataan PREPARE dijalankan, pernyataan yang ditentukan (SELECT, INSERT, UPDATE, atau DELETE) diurai, ditulis ulang, dan direncanakan. Ketika perintah EXECUTE kemudian dikeluarkan untuk pernyataan yang disiapkan, Amazon Redshift dapat secara opsional merevisi rencana eksekusi kueri (untuk meningkatkan kinerja berdasarkan nilai parameter yang ditentukan) sebelum menjalankan pernyataan yang disiapkan.

Sintaksis

```
PREPARE plan_name [ (datatype [, ...] ) ] AS statement
```

Parameter

nama_rencana

Nama sewenang-wenang yang diberikan untuk pernyataan khusus yang disiapkan ini. Ini harus unik dalam satu sesi dan selanjutnya digunakan untuk menjalankan atau mengalokasikan pernyataan yang disiapkan sebelumnya.

tipe data

Tipe data parameter untuk pernyataan yang disiapkan. Untuk merujuk pada parameter dalam pernyataan yang disiapkan itu sendiri, gunakan $1, $2, dan seterusnya, hingga maksimum $32767.

pernyataan

Setiap pernyataan SELECT, INSERT, UPDATE, atau DELETE.

Catatan penggunaan

Pernyataan yang disiapkan dapat mengambil parameter: nilai yang diganti ke dalam pernyataan ketika dijalankan. Untuk memasukkan parameter dalam pernyataan yang disiapkan, berikan daftar tipe data dalam pernyataan PREPARE, dan, dalam pernyataan yang akan disiapkan sendiri, lihat parameter berdasarkan posisi menggunakan notasi $1, $2,... Jumlah parameter maksimum adalah 32767. Saat menjalankan pernyataan, tentukan nilai aktual untuk parameter ini dalam pernyataan EXECUTE. Untuk detail selengkapnya, lihat EXECUTE.

Pernyataan yang disiapkan hanya bertahan selama sesi saat ini. Ketika sesi berakhir, pernyataan yang disiapkan dibuang, sehingga harus dibuat ulang sebelum digunakan lagi. Ini juga berarti bahwa satu pernyataan yang disiapkan tidak dapat digunakan oleh beberapa klien database simultan; Namun, setiap klien dapat membuat pernyataan yang disiapkan sendiri untuk digunakan. Pernyataan yang disiapkan dapat dihapus secara manual menggunakan perintah DEALLOCATE.

Pernyataan yang disiapkan memiliki keuntungan kinerja terbesar ketika satu sesi digunakan untuk menjalankan sejumlah besar pernyataan serupa. Seperti disebutkan, untuk setiap eksekusi baru

dari pernyataan yang disiapkan, Amazon Redshift dapat merevisi rencana eksekusi kueri untuk meningkatkan kinerja berdasarkan nilai parameter yang ditentukan. Untuk memeriksa rencana eksekusi kueri yang telah dipilih Amazon Redshift untuk pernyataan EXECUTE tertentu, gunakan EXPLAIN perintah.

Untuk informasi selengkapnya tentang perencanaan kueri dan statistik yang dikumpulkan oleh Amazon Redshift untuk pengoptimalan kueri, lihat MENGANALISA perintah.

Contoh

Buat tabel sementara, siapkan pernyataan INSERT dan kemudian jalankan:

```
DROP TABLE IF EXISTS prep1;
CREATE TABLE prep1 (c1 int, c2 char(20));
PREPARE prep_insert_plan (int, char)
AS insert into prep1 values ($1, $2);
EXECUTE prep_insert_plan (1, 'one');
EXECUTE prep_insert_plan (2, 'two');
EXECUTE prep_insert_plan (3, 'three');
DEALLOCATE prep_insert_plan;
```

Siapkan pernyataan SELECT dan kemudian jalankan:

```
PREPARE prep_select_plan (int)
AS select * from prep1 where c1 = $1;
EXECUTE prep_select_plan (2);
EXECUTE prep_select_plan (3);
DEALLOCATE prep_select_plan;
```

Lihat juga

DEALOKASI, EXECUTE

MENYEGARKAN TAMPILAN YANG TERWUJUD

Menyegarkan tampilan yang terwujud.

Saat Anda membuat tampilan terwujud, isinya mencerminkan keadaan tabel atau tabel database yang mendasarinya pada saat itu. Data dalam tampilan terwujud tetap tidak berubah, bahkan ketika aplikasi membuat perubahan pada data dalam tabel yang mendasarinya.

Untuk memperbarui data dalam tampilan yang terwujud, Anda dapat menggunakan `REFRESH MATERIALIZED VIEW` pernyataan kapan saja. Saat Anda menggunakan pernyataan ini, Amazon Redshift mengidentifikasi perubahan yang telah terjadi di tabel atau tabel dasar, dan kemudian menerapkan perubahan tersebut ke tampilan yang terwujud.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

Sintaksis

```
REFRESH MATERIALIZED VIEW mv_name [ RESTRICT | CASCADE ]
```

Parameter

mv_nama_

Nama tampilan terwujud yang akan disegarkan.

MEMBATASI

Kata kunci opsional. Menyegarkan tampilan terwujud yang ditentukan tetapi bukan tampilan terwujud dependen. Default jika tidak ditentukan RESTRICT atau CASCADE.

KASKADE

Kata kunci opsional. Menyegarkan tampilan terwujud yang ditentukan dan semua tampilan terwujud dependen.

Catatan penggunaan

Hanya pemilik tampilan terwujud yang dapat melakukan `REFRESH MATERIALIZED VIEW` operasi pada tampilan terwujud itu. Selanjutnya, pemilik harus memiliki hak istimewa SELECT pada tabel dasar yang mendasarinya agar berhasil dijalankan`REFRESH MATERIALIZED VIEW`.

Ketika tampilan terwujud memerlukan komputasi ulang penuh, pemilik juga harus memiliki hak istimewa CREATE pada skema yang berisi tampilan terwujud. Komputasi ulang penuh dapat dipicu oleh operasi seperti VACUUM atau TRUNCATE pada tabel dasar.

Per `REFRESH MATERIALIZED VIEW` intah berjalan sebagai transaksi sendiri. Semantik transaksi Amazon Redshift diikuti untuk menentukan data apa dari tabel dasar yang terlihat oleh `REFRESH`

perintah, atau kapan perubahan yang dibuat oleh `REFRESH` perintah dibuat terlihat oleh transaksi lain yang berjalan di Amazon Redshift.

- Untuk tampilan terwujud inkremental, hanya `REFRESH MATERIALIZED VIEW` menggunakan baris tabel dasar yang sudah dikomit. Oleh karena itu, jika operasi refresh berjalan setelah pernyataan bahasa manipulasi data (DML) dalam transaksi yang sama, maka perubahan pernyataan DML tersebut tidak terlihat untuk di-refresh.
- Untuk penyegaran penuh dari tampilan yang terwujud, lihat `REFRESH MATERIALIZED VIEW` semua baris tabel dasar terlihat untuk transaksi penyegaran, sesuai dengan semantik transaksi Amazon Redshift biasa.
- Bergantung pada jenis argumen input, Amazon Redshift masih mendukung penyegaran tambahan untuk tampilan terwujud untuk fungsi berikut dengan jenis argumen input tertentu: DATE (timestamp), DATE_PART (tanggal, waktu, interval, time-tz), DATE_TRUNC (timestamp, interval).
- Penyegaran inkremental didukung pada tampilan terwujud di mana tabel dasar berada di datashare.
- Penyegaran tampilan terwujud bersama dari cluster pembagian data jarak jauh tidak didukung untuk tampilan terwujud yang berisi referensi ke tampilan terwujud lainnya, tabel Spectrum, tabel yang ditentukan dalam cluster Redshift atau UDF yang berbeda. Pandangan terwujud seperti itu dapat disegarkan dari cluster lokal (produsen).

Beberapa operasi di Amazon Redshift berinteraksi dengan tampilan yang terwujud. Beberapa operasi ini mungkin memaksa `REFRESH MATERIALIZED VIEW` operasi untuk menghitung ulang tampilan yang terwujud sepenuhnya meskipun kueri yang mendefinisikan tampilan terwujud hanya menggunakan fitur SQL yang memenuhi syarat untuk penyegaran tambahan. Contoh:

- Operasi vakum latar belakang mungkin diblokir jika tampilan yang terwujud tidak disegarkan. Setelah periode ambang batas yang ditentukan secara internal, operasi vakum diizinkan untuk berjalan. Ketika operasi vakum ini terjadi, setiap tampilan terwujud dependen ditandai untuk perhitungan ulang pada penyegaran berikutnya (bahkan jika mereka inkremental). Untuk informasi tentang VACUUM, lihatVAKUM. Untuk informasi selengkapnya tentang peristiwa dan perubahan status, lihatSTL_MV_STATE.
- Beberapa operasi yang diprakarsai pengguna pada tabel dasar memaksa tampilan yang terwujud untuk dihitung ulang sepenuhnya saat operasi REFRESH dijalankan. Contoh operasi tersebut adalah VACUUM yang dipanggil secara manual, perubahan ukuran klasik, operasi ALTER DISTKEY, operasi ALTER SORTKEY, dan operasi truncate. Operasi otomatis dalam beberapa kasus juga dapat mengakibatkan tampilan terwujud dihitung ulang sepenuhnya saat

operasi REFRESH dijalankan berikutnya. Misalnya, operasi penghapusan vakum otomatis dapat menyebabkan komputasi ulang penuh. Untuk informasi selengkapnya tentang peristiwa dan perubahan status, lihatSTL_MV_STATE.

Penyegaran bertingkat

Opsi CASCADE menyegarkan tampilan terwujud yang ditentukan dan semua tampilan terwujud dependen, dalam urutan ketergantungan: MV dasar disegarkan sebelum MV di atas (urutan topologi). Dengan ini, Anda dapat memperbarui kumpulan tampilan terwujud bersarang dalam satu perintah.

Opsi RESTRICT (default jika RESTRICT atau CASCADE tidak ditentukan) hanya menyegarkan tampilan terwujud yang ditentukan.

Saat menggunakan opsi CASCADE, aturan berikut berlaku:

- Hanya pemilik tampilan yang terwujud atau superuser yang dapat menjalankan perintah`REFRESH MATERIALIZED VIEW ... CASCADE`.
- Jika salah satu tampilan yang terwujud dalam kaskade tidak dapat disegarkan, seluruh operasi kaskade akan berhenti.

Fungsionalitas penyegaran bertingkat hanya didukung untuk MV yang bersarang di atas tampilan lokal dan streaming yang terwujud. Tampilan terwujud dengan jenis sumber lain, seperti Spectrum atau Data Sharing, tidak didukung dalam mode kaskade. CASCADE mengeksekusi refresh dalam satu transaksi untuk semua MV bersarang.

Penyegaran tambahan untuk tampilan terwujud dalam datashare

Amazon Redshift mendukung penyegaran otomatis dan inkremental untuk tampilan yang terwujud di datashare konsumen saat tabel dasar dibagikan. Penyegaran inkremental adalah operasi di mana Amazon Redshift mengidentifikasi perubahan pada tabel dasar atau tabel yang terjadi setelah penyegaran sebelumnya dan hanya memperbarui catatan yang sesuai dalam tampilan yang terwujud. Untuk informasi selengkapnya tentang perilaku ini, lihat MEMBUAT TAMPILAN TERMATERIALISASI.

Batasan untuk penyegaran tambahan

Amazon Redshift saat ini tidak mendukung penyegaran tambahan untuk tampilan terwujud yang ditentukan dengan kueri menggunakan salah satu elemen SQL berikut:

- OUTER JOIN (KANAN, KIRI, atau PENUH).
- Tetapkan operasi: UNION, INTERSECT, EXCEPT, MINUS.
- UNION ALL ketika terjadi dalam subquery dan fungsi agregat, atau klausa GROUP BY hadir dalam kueri, atau tampilan terwujud target berisi sortkey.
- Fungsi agregat: MEDIAN, PERCENTILE_CONT, LISTAGG, STDDEV_SAMP, STDDEV_POP, APPROXIMATE COUNT, APPROXIMATE PERCENTILE, dan fungsi agregat bitwise.

  > **Note**
  >
  > Fungsi agregat COUNT, SUM, MIN, MAX, dan AVG didukung.

- Fungsi agregat DISTINCT, seperti DISTINCT COUNT, DISTINCT SUM, dan sebagainya.
- Fungsi Jendela.
- Kueri yang menggunakan tabel sementara untuk pengoptimalan kueri, seperti mengoptimalkan subexpressi umum.
- Subkueri
- Tabel eksternal merujuk format berikut dalam kueri yang mendefinisikan tampilan terwujud.
  - Danau Delta
  - Hudi

  Penyegaran tambahan didukung untuk tampilan terwujud yang ditentukan menggunakan format selain yang tercantum di atas. Untuk informasi selengkapnya, lihat Tampilan terwujud pada tabel danau data eksternal di Amazon Redshift Spectrum.
- Fungsi yang dapat berubah, seperti fungsi tanggal-waktu, fungsi ACAK dan non-stabil yang ditentukan pengguna.
- Untuk keterbatasan mengenai penyegaran tambahan untuk integrasi Zero-ETL, lihat Per timbangan saat menggunakan integrasi Zero-ETL dengan Amazon Redshift.
- Mengakses tabel dari lebih dari satu database.

Untuk informasi selengkapnya tentang batasan tampilan terwujud, termasuk efek operasi latar belakang seperti VACUUM pada operasi penyegaran tampilan terwujud, lihat. Catatan penggunaan

Contoh

Contoh berikut menyegarkan tampilan yang ter `tickets_mv` wujud.

```
REFRESH MATERIALIZED VIEW tickets_mv;
```

Contoh berikut menyegarkan tampilan ter `products_mv` wujud dan semua tampilan terwujud dependen:

```
REFRESH MATERIALIZED VIEW products_mv CASCADE;
```

MENGATUR ULANG

Mengembalikan nilai parameter konfigurasi ke nilai defaultnya.

Anda dapat mengatur ulang satu parameter tertentu atau semua parameter sekaligus. Untuk mengatur parameter ke nilai tertentu, gunakan SET perintah. Untuk menampilkan nilai parameter saat ini, gunakan MENUNJUKKAN perintah.

Sintaksis

```
RESET { parameter_name | ALL }
```

Pernyataan berikut menetapkan nilai variabel konteks sesi ke NULL.

```
RESET { variable_name | ALL }
```

Parameter

nama_parameter

Nama parameter yang akan diatur ulang. Lihat dokument Memodifikasi konfigurasi server asi lebih lanjut tentang parameter.

SEMUA

Mengatur ulang semua parameter runtime, termasuk semua variabel konteks sesi.

variabel

Nama variabel yang akan diatur ulang. Jika nilai RESET adalah variabel konteks sesi, Amazon Redshift menyetelnya ke NULL.

Contoh

Contoh berikut mengatur ulang `query_group` parameter ke nilai defaultnya:

```
reset query_group;
```

Contoh berikut mengatur ulang semua parameter runtime ke nilai defaultnya.

```
reset all;
```

Contoh berikut mengatur ulang variabel konteks.

```
RESET app_context.user_id;
```

MENCABUT

Menghapus izin akses, seperti izin untuk membuat, menghapus, atau memperbarui tabel, dari pengguna atau peran.

Anda hanya dapat MEMBERIKAN atau MENCABUT izin USAGE pada skema eksternal kepada pengguna dan peran database menggunakan sintaks ON SCHEMA. Saat menggunakan ON EXTERNAL SCHEMA dengan AWS Lake Formation, Anda hanya dapat MEMBERIKAN dan MENCABUT izin ke peran AWS Identity and Access Management (IAM). Untuk daftar izin, lihat sintaks.

Untuk prosedur tersimpan, `plpgsql` izin USAGE ON LANGUAGE diberikan kepada PUBLIK secara default. Izin EXECUTE ON PROCEDURE hanya diberikan kepada pemilik dan superuser secara default.

Tentukan dalam perintah REVOKE izin yang ingin Anda hapus. Untuk memberikan izin, gunakan HIBAH perintah.

Sintaksis

```
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | ALTER | TRUNCATE } [,...] |
 ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
{ { CREATE | TEMPORARY | TEMP | ALTER } [,...] | ALL [ PRIVILEGES ] }
ON DATABASE db_name [, ...]
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }
ON SCHEMA schema_name [, ...]
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
EXECUTE
    ON FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ { EXECUTE } [,...] | ALL [ PRIVILEGES ] }
    ON PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
USAGE
    ON LANGUAGE language_name [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]

REVOKE [GRANT OPTION FOR]
{ { ALTER | DROP} [,...] | ALL [ PRIVILEGES ] }
    ON COPY JOB job_name [,...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]

REVOKE [GRANT OPTION FOR]
{ { ALTER | DROP | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON TEMPLATE template_name [,...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

Mencabut izin tingkat kolom untuk tabel

Berikut ini adalah sintaks untuk izin tingkat kolom pada tabel dan tampilan Amazon Redshift.

```
REVOKE { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
 ( column_name [,...] ) }
     ON { [ TABLE ] table_name [, ...] }
     FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
     [ RESTRICT ]
```

Mencabut izin ASSUMEROLE

Berikut ini adalah sintaks untuk mencabut izin ASSUMEROLE dari pengguna dan grup dengan peran tertentu.

```
REVOKE ASSUMEROLE
    ON { 'iam_role' [, ...]  | default | ALL }
    FROM { user_name | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL }
```

Mencabut izin untuk Redshift Spectrum untuk Formasi Danau

Berikut ini adalah sintaks untuk integrasi Redshift Spectrum dengan Lake Formation.

```
REVOKE [ GRANT OPTION FOR ]
{ SELECT | ALL [ PRIVILEGES ] } ( column_list )
    ON EXTERNAL TABLE schema_name.table_name
    FROM { IAM_ROLE iam_role } [, ...]

REVOKE [ GRANT OPTION FOR ]
{ { SELECT | ALTER | DROP | DELETE | INSERT }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL TABLE schema_name.table_name [, ...]
    FROM { { IAM_ROLE iam_role } [, ...] | PUBLIC }

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | ALTER | DROP }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL SCHEMA schema_name [, ...]
    FROM { IAM_ROLE iam_role } [, ...]
```

Mencabut izin datashare

Producer-side izin datashare

Berikut ini adalah sintaks untuk menggunakan REVOKE untuk menghapus izin ALTER atau SHARE dari pengguna atau peran. Pengguna yang izinnya telah dicabut tidak dapat lagi mengubah data, atau memberikan penggunaan kepada konsumen.

```
REVOKE { ALTER | SHARE } ON DATASHARE datashare_name
 FROM { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

Berikut ini adalah sintaks untuk menggunakan REVOKE untuk menghapus akses konsumen ke datashare.

```
REVOKE USAGE
 ON DATASHARE datashare_name
 FROM NAMESPACE 'namespaceGUID' [, ...] | ACCOUNT 'accountnumber' [ VIA DATA CATALOG ]
 [, ...]
```

Berikut ini adalah contoh pencabutan penggunaan datashare dari akun Formasi Danau.

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012' VIA DATA CATALOG;
```

Consumer-side izin datashare

Berikut ini adalah sintaks REVOKE untuk izin penggunaan berbagi data pada database atau skema tertentu yang dibuat dari datashare. Mencabut izin penggunaan dari database yang dibuat dengan klausa WITH PERMISSIONS tidak mencabut izin tambahan yang Anda berikan kepada pengguna atau peran, termasuk izin tingkat objek yang diberikan untuk objek yang mendasarinya. Jika Anda memberikan kembali izin penggunaan kepada pengguna atau peran tersebut, mereka akan mempertahankan semua izin tambahan yang mereka miliki sebelum Anda mencabut penggunaan.

```
REVOKE USAGE ON { DATABASE shared_database_name [, ...] | SCHEMA shared_schema}
 FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

Mencabut izin yang dicakup

Izin cakupan memungkinkan Anda memberikan izin kepada pengguna atau peran pada semua objek dari jenis dalam database atau skema. Pengguna dan peran dengan izin cakupan memiliki izin yang ditentukan pada semua objek saat ini dan yang akan datang dalam database atau skema.

Anda dapat melihat cakupan izin cakupan tingkat database di. SVV_DATABASE_HAK ISTIMEWA Anda dapat melihat cakupan izin cakupan tingkat skema di. SVV_SCHEMA_HAK ISTIMEWA

Untuk informasi selengkapnya tentang izin bercakupan, lihatIzin terlingkup.

Berikut ini adalah sintaks untuk mencabut izin cakupan dari pengguna dan peran.

```
REVOKE [ GRANT OPTION ]
{ CREATE | USAGE | ALTER | DROP } [,...] | ALL [ PRIVILEGES ] }
FOR SCHEMAS IN
DATABASE db_name
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ]
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [PRIVILEGES] } }
FOR TABLES IN
{ SCHEMA schema_name [ DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ] { EXECUTE | ALL [ PRIVILEGES ] }
FOR FUNCTIONS IN
{ SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ] { EXECUTE | ALL [ PRIVILEGES ] }
FOR PROCEDURES IN
{ SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ] USAGE
FOR LANGUAGES IN
DATABASE db_name
FROM { username | ROLE role_name } [, ...]

REVOKE [GRANT_OPTION]
{ { CREATE | ALTER | DROP} [,...] | ALL [ PRIVILEGES ] }
FOR COPY JOBS
IN DATABASE db_name
FROM { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ]
{ {ALTER | DROP  | USAGE } [,...] | ALL [ PRIVILEGES ] }
FOR TEMPLATES IN
{ SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name } [, ...]
```

Perhatikan bahwa izin cakupan tidak membedakan antara izin untuk fungsi dan prosedur. Misalnya, pernyataan berikut mencabut EXECUTE izin untuk fungsi dan prosedur dari bob dalam skema. Sales_schema

```
REVOKE EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema FROM bob;
```

Mencabut izin pembelajaran mesin

Berikut ini adalah sintaks untuk izin model pembelajaran mesin di Amazon Redshift.

```
REVOKE [ GRANT OPTION FOR ]
    CREATE MODEL FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
    { EXECUTE | ALL [ PRIVILEGES ] }
    ON MODEL model_name [, ...]

    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    [ RESTRICT ]
```

Mencabut izin peran

Berikut ini adalah sintaks untuk mencabut izin peran di Amazon Redshift.

```
REVOKE [ ADMIN OPTION FOR ] { ROLE role_name } [, ...] FROM { user_name } [, ...]
```

```
REVOKE { ROLE role_name } [, ...] FROM { ROLE role_name } [, ...]
```

Berikut ini adalah sintaks untuk mencabut izin sistem untuk peran di Amazon Redshift.

```
REVOKE
  {
    { CREATE USER | DROP USER | ALTER USER |
    CREATE SCHEMA | DROP SCHEMA |
    ALTER DEFAULT PRIVILEGES |
    ACCESS CATALOG |
    CREATE TABLE | DROP TABLE | ALTER TABLE |
    CREATE OR REPLACE FUNCTION | CREATE OR REPLACE EXTERNAL FUNCTION |
    DROP FUNCTION |
    CREATE OR REPLACE PROCEDURE | DROP PROCEDURE |
```

```
    CREATE OR REPLACE VIEW | DROP VIEW |
    CREATE MODEL | DROP MODEL |
    CREATE DATASHARE | ALTER DATASHARE | DROP DATASHARE |
    CREATE LIBRARY | DROP LIBRARY |
    CREATE ROLE | DROP ROLE
    TRUNCATE TABLE
    VACUUM | ANALYZE | CANCEL }[, ...]
  }
  | { ALL [ PRIVILEGES ] }
FROM { ROLE role_name } [, ...]
```

Mencabut izin untuk kebijakan keamanan

Berikut ini adalah sintaks untuk mencabut izin untuk menjelaskan filter kebijakan keamanan dari kueri dalam paket EXPLAIN. Kebijakan keamanan yang mungkin termasuk kebijakan keamanan tingkat baris dan kebijakan penyembunyian data dinamis.

```
REVOKE EXPLAIN { RLS | MASKING } FROM ROLE rolename
```

Berikut ini adalah sintaks untuk mencabut izin untuk melewati kebijakan keamanan tingkat baris untuk kueri.

```
REVOKE IGNORE RLS FROM ROLE rolename
```

Berikut ini adalah sintaks untuk mencabut izin SELECT dari kebijakan keamanan yang ditentukan. Kebijakan keamanan yang mungkin termasuk kebijakan keamanan tingkat baris dan kebijakan penyembunyian data dinamis.

```
REVOKE SELECT ON [ TABLE ] table_name [, ...]
            FROM { RLS | MASKING } POLICY policy_name [, ...]
```

Parameter

OPSI HIBAH UNTUK

Hanya mencabut opsi untuk memberikan izin tertentu kepada pengguna lain dan tidak mencabut izin itu sendiri. Anda tidak dapat mencabut OPSI GRANT dari grup atau dari PUBLIK.

SELECT

Mencabut izin untuk memilih data dari tabel atau tampilan menggunakan pernyataan SELECT.

INSERT

Mencabut izin untuk memuat data ke dalam tabel menggunakan pernyataan INSERT atau pernyataan COPY.

UPDATE

Mencabut izin untuk memperbarui kolom tabel menggunakan pernyataan UPDATE.

DELETE

Mencabut izin untuk menghapus baris data dari tabel.

REFERENSI

Mencabut izin untuk membuat batasan kunci asing. Anda harus mencabut izin ini pada tabel yang direferensikan dan tabel referensi.

TRUNCATE

Mencabut izin untuk memotong tabel. Tanpa izin ini, hanya pemilik tabel atau superuser yang dapat memotong tabel. Untuk informasi selengkapnya tentang perintah TRUNCATE, lihatthe section called “MEMOTONG”.

SEMUA [HAK ISTIMEWA]

Mencabut semua izin yang tersedia sekaligus dari pengguna atau grup yang ditentukan. Kata kunci PRIVILEGES adalah opsional.

> **Note**
>
> Amazon Redshift tidak mendukung izin RULE dan TRIGGER. Untuk informasi selengkapnya, kunjungi Fitur PostgreSQL yang tidak didukung.

ALTER

Bergantung pada objek database, mencabut izin berikut dari pengguna atau grup pengguna:

- Untuk tabel, ALTER mencabut izin untuk mengubah tabel atau tampilan. Untuk informasi selengkapnya, lihat ALTER TABLE.
- Untuk database, ALTER mencabut izin untuk mengubah database. Untuk informasi selengkapnya, lihat ALTER DATABASE.

- Untuk skema, ALTER memberikan pencabutan untuk mengubah skema. Untuk informasi selengkapnya, lihat ALTER SCHEMA.
- Untuk tabel eksternal, ALTER mencabut izin untuk mengubah tabel di tabel AWS Glue Data Catalog yang diaktifkan untuk Formasi Danau. Izin ini hanya berlaku saat menggunakan Formasi Danau.

DROP

Bergantung pada objek database, mencabut izin berikut dari pengguna atau peran:

- Untuk tabel, DROP mencabut izin untuk melepaskan tabel atau tampilan. Untuk informasi selengkapnya, lihat MEJA DROP.
- Untuk database, DROP mencabut izin untuk menghapus database. Untuk informasi selengkapnya, lihat DROP DATABASE.
- Untuk skema, DROP mencabut izin untuk menghapus skema. Untuk informasi selengkapnya, lihat DROP SCHEMA.

ASSUMEROLE

Mencabut izin untuk menjalankan perintah COPY, UNLOAD, EXTERNAL FUNCTION, atau CREATE MODEL dari pengguna, peran, atau grup dengan peran tertentu.

PADA [TABEL] nama_ tabel

Mencabut izin yang ditentukan pada tabel atau tampilan. Kata kunci TABLE adalah opsional.

PADA SEMUA TABEL DI SCHEMA schema_name

Mencabut izin yang ditentukan pada semua tabel dalam skema yang direferensikan.

(column_name [,...]) PADA TABLE table_name

Mencabut izin yang ditentukan dari pengguna, grup, atau PUBLIK pada kolom tertentu dari tabel atau tampilan Amazon Redshift.

(column_list) PADA TABEL EKSTERNAL schema_name.table_name

Mencabut izin yang ditentukan dari peran IAM pada kolom tertentu dari tabel Formasi Danau dalam skema yang direferensikan.

PADA TABEL EKSTERNAL schema_name.table_name

Mencabut izin yang ditentukan dari peran IAM pada tabel Formasi Danau yang ditentukan dalam skema yang direferensikan.

PADA SKEMA EKSTERNAL SCHEMA_NAME

Mencabut izin yang ditentukan dari peran IAM pada skema yang direferensikan.

DARI IAM_ROLE iam_role

Menunjukkan peran IAM kehilangan izin.

Nama peran

Mencabut izin dari peran yang ditentukan.

Grup nama_grup

Mencabut izin dari grup pengguna yang ditentukan.

PUBLIK

Mencabut izin yang ditentukan dari semua pengguna. PUBLIK mewakili grup yang selalu mencakup semua pengguna. Izin pengguna individu terdiri dari jumlah izin yang diberikan kepada PUBLIK, izin yang diberikan kepada grup mana pun yang dimiliki pengguna, dan izin apa pun yang diberikan kepada pengguna secara individual.

Mencabut PUBLIK dari tabel eksternal Formasi Danau mengakibatkan pencabutan izin dari kelompok semua orang Formasi Danau.

CREATE

Bergantung pada objek database, mencabut izin berikut dari pengguna atau grup:

- Untuk database, menggunakan klausa CREATE untuk REVOKE mencegah pengguna membuat skema dalam database.
- Untuk skema, menggunakan klausa CREATE untuk REVOKE mencegah pengguna membuat objek dalam skema. Untuk mengganti nama objek, pengguna harus memiliki izin CREATE dan memiliki objek yang akan diganti namanya.

> **Note**
>
> Secara default, semua pengguna memiliki izin CREATE dan USAGE pada skema PUBLIK.

SEMENTARA | TEMP

Mencabut izin untuk membuat tabel sementara di database yang ditentukan.

> **Note**
>
> Secara default, pengguna diberikan izin untuk membuat tabel sementara oleh keanggotaan otomatis mereka di grup PUBLIK. Untuk menghapus izin bagi pengguna mana pun untuk membuat tabel sementara, cabut izin TEMP dari grup PUBLIK dan kemudian secara eksplisit berikan izin untuk membuat tabel sementara kepada pengguna atau grup pengguna tertentu.

PADA DATABASE db_name

Mencabut izin pada database yang ditentukan.

PENGGUNAAN

Mencabut izin USAGE pada objek dalam skema tertentu, yang membuat objek ini tidak dapat diakses oleh pengguna. Tindakan spesifik pada objek ini harus dicabut secara terpisah (seperti izin EXECUTE pada fungsi).

> **Note**
>
> Secara default, semua pengguna memiliki izin CREATE dan USAGE pada skema PUBLIK.

ON SCHEMA schema_name

Mencabut izin pada skema yang ditentukan. Anda dapat menggunakan izin skema untuk mengontrol pembuatan tabel; izin CREATE untuk database hanya mengontrol pembuatan skema.

MEMBATASI

Hanya mencabut izin yang diberikan pengguna secara langsung. Ini adalah perilaku default .

JALANKAN NAMA_prosedur PADA PROSED UR

Mencabut izin EXECUTE pada prosedur tersimpan tertentu. Karena nama prosedur yang disimpan dapat kelebihan beban, Anda harus menyertakan daftar argumen untuk prosedur. Untuk informasi selengkapnya, lihat Penamaan prosedur yang disimpan.

JALANKAN PADA SEMUA PROSEDUR DI SCHEMA procedur_name

Mencabut izin yang ditentukan pada semua prosedur dalam skema yang direferensikan.

PENGGUNAAN PADA LANGUAGE language_name

Mencabut izin PENGGUNAAN pada suatu bahasa. Untuk fungsi yang ditentukan pengguna Python (UDF), gunakan. `plpythonu` Untuk SQL UDF, gunakan`sql`. Untuk prosedur yang disimpan, gunakan`plpgsql`.

Untuk membuat UDF, Anda harus memiliki izin untuk penggunaan bahasa untuk SQL atau `plpythonu` (Python). Secara default, USAGE ON LANGUAGE SQL diberikan kepada PUBLIK. Namun, Anda harus secara eksplisit memberikan USAGE ON LANGUAGE PLPYTHONU kepada pengguna atau grup tertentu.

Untuk mencabut penggunaan untuk SQL, pertama-tama cabut penggunaan dari PUBLIK. Kemudian berikan penggunaan pada SQL hanya untuk pengguna atau grup tertentu yang diizinkan untuk membuat SQL UDF. Contoh berikut mencabut penggunaan pada SQL dari PUBLIC kemudian memberikan penggunaan ke grup pengguna. `udf_devs`

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

Untuk informasi selengkapnya, lihat Keamanan dan izin UDF.

Untuk mencabut penggunaan prosedur yang disimpan, pertama-tama cabut penggunaan dari PUBLIK. Kemudian berikan penggunaan `plpgsql` hanya kepada pengguna atau grup tertentu yang diizinkan untuk membuat prosedur tersimpan. Untuk informasi selengkapnya, lihat Keamanan dan hak istimewa untuk prosedur tersimpan.

ON COPY JOB nama_ pekerjaan

Mencabut izin yang ditentukan pada pekerjaan penyalinan.

UNTUK {SEMUA | SALIN | MEMBONGKAR | FUNGSI EKSTERNAL | BUAT MODEL} [,...]

Menentukan perintah SQL yang izinnya dicabut. Anda dapat menentukan SEMUA untuk mencabut izin pada pernyataan COPY, UNLOAD, EXTERNAL FUNCTION, dan CREATE MODEL. Klausul ini hanya berlaku untuk mencabut izin ASSUMEROLE.

MENGUBAH

Mencabut izin ALTER untuk pengguna atau grup pengguna yang memungkinkan mereka yang tidak memiliki datashare untuk mengubah datashare. Izin ini diperlukan untuk menambah atau menghapus objek dari datashare, atau untuk mengatur properti PUBLICACCESSIBLE. Untuk informasi selengkapnya, lihat ALTER DATA.

BERBAGI

Mencabut izin bagi pengguna dan grup pengguna untuk menambahkan konsumen ke datashare. Pencabutan izin ini diperlukan untuk menghentikan konsumen tertentu mengakses datashare dari klasternya.

PADA DATASHARE datashare_name

Memberikan izin yang ditentukan pada datashare yang direferensikan.

FROM nama pengguna

Menunjukkan pengguna kehilangan izin.

DARI GRUP group_name

Menunjukkan grup pengguna kehilangan izin.

DENGAN OPSI HIBAH

Menunjukkan bahwa pengguna yang kehilangan izin pada gilirannya dapat mencabut izin yang sama untuk orang lain. Anda tidak dapat mencabut WITH GRANT OPTION untuk grup atau untuk PUBLIK.

PENGGUNAAN

Ketika USAGE dicabut untuk akun konsumen atau namespace dalam akun yang sama, akun konsumen atau namespace yang ditentukan dalam akun tidak dapat mengakses datashare dan objek datashare dengan cara read-only.

Mencabut izin PENGGUNAAN mencabut akses ke datashare dari konsumen.

DARI NAMESPACE 'GUID' clusternamespace '

Menunjukkan namespace di akun yang sama yang membuat konsumen kehilangan izin ke datashare. Ruang nama menggunakan pengenal unik global (GUID) alfanumerik 128-bit.

DARI AKUN 'accountnumber' [VIA KATALOG DATA]

Menunjukkan nomor akun akun lain yang membuat konsumen kehilangan izin ke datashare. Menentukan 'VIA DATA CATALOG' menunjukkan bahwa Anda mencabut penggunaan datashare dari akun Lake Formation. Menghilangkan nomor akun berarti Anda mencabut dari akun yang memiliki cluster.

PADA DATABASE shared_database_name> [,...]

Mencabut izin penggunaan yang ditentukan pada database tertentu yang dibuat di datashare yang ditentukan.

PADA SKEMA shared_schema

Mencabut izin yang ditentukan pada skema tertentu yang dibuat di datashare yang ditentukan.

UNTUK {SKEMA | TABEL | FUNGSI | PROSEDUR | BAHASA | PEKERJAAN SALIN} DI

Menentukan objek database untuk mencabut izin. Parameter berikut IN menentukan ruang lingkup izin yang dicabut.

BUAT MODEL

Mencabut izin CREATE MODEL untuk membuat model pembelajaran mesin dalam database yang ditentukan.

NAMA_model ON MODEL

Mencabut izin EXECUTE untuk model tertentu.

AKSES KATALOG

Mencabut izin untuk melihat metadata objek yang relevan yang dapat diakses peran tersebut.

[OPSI ADMIN UNTUK] {peran} [,...]

Peran yang Anda cabut dari pengguna tertentu yang memiliki OPSI DENGAN ADMIN.

DARI {peran} [,...]

Peran tempat Anda mencabut peran yang ditentukan.

JELASKAN {RLS | MASKING} DARI ROLE rolename

Mencabut izin untuk menjelaskan filter kebijakan keamanan dari kueri dalam rencana EXPLAIN dari peran. RLS mencabut izin untuk menjelaskan filter kebijakan keamanan tingkat baris. MASKING mencabut izin untuk menjelaskan filter kebijakan penyembunyian data dinamis.

ABAIKAN RLS DARI ROLE rolename

Mencabut izin untuk melewati kebijakan keamanan tingkat baris untuk kueri dari peran.

DARI {RLS | MASKING} KEBIJAKAN policy_name

Menunjukkan kebijakan keamanan kehilangan izin. KEBIJAKAN TO RLS menunjukkan kebijakan keamanan tingkat baris. TO MASKING POLICY menunjukkan kebijakan penyembunyian data dinamis.

Catatan penggunaan

Untuk mempelajari lebih lanjut tentang catatan penggunaan untuk REVOKE, lihatthe section called “Catatan penggunaan”.

Contoh

Untuk contoh cara menggunakan REVOKE, lihatthe section called “Contoh”.

Catatan penggunaan

Untuk mencabut hak istimewa dari objek, Anda harus memenuhi salah satu kriteria berikut:

- Jadilah pemilik objek.
- Jadilah pengguna super.
- Miliki hak istimewa hibah untuk objek dan hak istimewa itu.

  Misalnya, perintah berikut memungkinkan HR pengguna untuk melakukan perintah SELECT pada tabel karyawan dan untuk memberikan dan mencabut hak istimewa yang sama untuk pengguna lain.

  ```
  grant select on table employees to HR with grant option;
  ```

  HR tidak dapat mencabut hak istimewa untuk operasi apa pun selain SELECT, atau pada tabel lain selain karyawan.

Superuser dapat mengakses semua objek terlepas dari perintah GRANT dan REVOKE yang menetapkan hak istimewa objek.

PUBLIK mewakili grup yang selalu mencakup semua pengguna. Secara default semua anggota PUBLIK memiliki hak CREATE dan USAGE pada skema PUBLIK. Untuk membatasi izin pengguna apa pun pada skema PUBLIK, Anda harus terlebih dahulu mencabut semua izin dari PUBLIK pada skema PUBLIK, kemudian memberikan hak istimewa kepada pengguna atau grup tertentu. Contoh berikut mengontrol hak pembuatan tabel dalam skema PUBLIK.

```
revoke create on schema public from public;
```

Untuk mencabut hak istimewa dari tabel Lake Formation, peran IAM yang terkait dengan skema eksternal tabel harus memiliki izin untuk mencabut hak istimewa ke tabel eksternal. Contoh berikut

membuat skema eksternal dengan peran `myGrantor` IAM terkait. Peran IAM `myGrantor` memiliki izin untuk mencabut izin dari orang lain. Perintah REVOKE menggunakan izin peran IAM `myGrantor` yang terkait dengan skema eksternal untuk mencabut izin ke peran IAM. `myGrantee`

```
create external schema mySchema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myGrantor'
create external database if not exists;
```

```
revoke select
on external table mySchema.mytable
from iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

> **Note**
>
> Jika peran IAM juga memiliki `ALL` izin dalam AWS Glue Data Catalog yang diaktifkan untuk Formasi Danau, `ALL` izin tidak dicabut. Hanya `SELECT` izin yang dicabut. Anda dapat melihat izin Formasi Danau di konsol Formasi Danau.

Catatan penggunaan untuk mencabut izin ASSUMEROLE

Catatan penggunaan berikut berlaku untuk mencabut hak istimewa ASSUMEROLE di Amazon Redshift.

Hanya superuser database yang dapat mencabut hak istimewa ASSUMEROLE untuk pengguna dan grup. Seorang superuser selalu mempertahankan hak istimewa ASSUMEROLE.

Untuk mengaktifkan penggunaan hak istimewa ASSUMEROLE untuk pengguna dan grup, superuser menjalankan pernyataan berikut sekali di cluster. Sebelum memberikan hak istimewa ASSUMEROLE kepada pengguna dan grup, superuser harus menjalankan pernyataan berikut sekali di cluster.

```
revoke assumerole on all from public for all;
```

Catatan penggunaan untuk mencabut izin pembelajaran mesin

Anda tidak dapat langsung memberikan atau mencabut izin yang terkait dengan fungsi ML. Fungsi ML milik model ML dan izin dikendalikan melalui model. Sebagai gantinya, Anda dapat mencabut

izin yang terkait dengan model ML. Contoh berikut menunjukkan cara mencabut izin run dari semua pengguna yang terkait dengan model. `customer_churn`

```
REVOKE EXECUTE ON MODEL customer_churn FROM PUBLIC;
```

Anda juga dapat mencabut semua izin dari pengguna untuk model `customer_churn` ML.

```
REVOKE ALL on MODEL customer_churn FROM ml_user;
```

Memberikan atau mencab `EXECUTE` ut izin yang terkait dengan fungsi ML akan gagal jika ada fungsi ML dalam skema, bahkan jika fungsi ML itu sudah memiliki `EXECUTE` izin melalui`GRANT EXECUTE ON MODEL`. Sebaiknya gunakan skema terpisah saat menggunakan `CREATE MODEL` perintah untuk menjaga fungsi ML dalam skema terpisah dengan sendirinya. Contoh berikut menunjukkan bagaimana melakukannya.

```
CREATE MODEL ml_schema.customer_churn
FROM customer_data
TARGET churn
FUNCTION ml_schema.customer_churn_prediction
IAM_ROLE default
SETTINGS (
 S3_BUCKET 'amzn-s3-demo-bucket'
);
```

Contoh

Contoh berikut mencabut hak istimewa INSERT pada tabel SALES dari grup pengguna GUESTS. Perintah ini mencegah anggota GUESTS agar tidak dapat memuat data ke tabel SALES dengan menggunakan perintah INSERT.

```
revoke insert on table sales from group guests;
```

Contoh berikut mencabut hak istimewa SELECT pada semua tabel dalam skema QA_TICKIT dari pengguna. `fred`

```
revoke select on all tables in schema qa_tickit from fred;
```

Contoh berikut mencabut hak istimewa untuk memilih dari tampilan untuk pengguna. `bobr`

```
revoke select on table eventview from bobr;
```

Contoh berikut mencabut hak istimewa untuk membuat tabel sementara di database TICKIT dari semua pengguna.

```
revoke temporary on database tickit from public;
```

Contoh berikut mencabut hak istimewa SELECT pada `cust_phone` kolom `cust_name` dan `cust_profile` tabel dari pengguna. `user1`

```
revoke select(cust_name, cust_phone) on cust_profile from user1;
```

Contoh berikut mencabut hak istimewa SELECT pada `cust_phone` kolom `cust_name` dan dan hak istimewa `cust_contact_preference` UPDATE pada kolom `cust_profile` tabel dari grup. `sales_group`

```
revoke select(cust_name, cust_phone), update(cust_contact_preference) on cust_profile
 from group sales_group;
```

Contoh berikut menunjukkan penggunaan kata kunci ALL untuk mencabut hak istimewa SELECT dan UPDATE pada tiga kolom tabel `cust_profile` dari `sales_admin` grup.

```
revoke ALL(cust_name, cust_phone,cust_contact_preference) on cust_profile from group
 sales_admin;
```

Contoh berikut mencabut hak istimewa SELECT pada `cust_name` kolom `cust_profile_vw` tampilan dari pengguna. `user2`

```
revoke select(cust_name) on cust_profile_vw from user2;
```

Contoh pencabutan izin USAGE dari database yang dibuat dari datashares

Contoh berikut mencabut akses ke `salesshare` datashare dari namespace. `13b8833d-17c6-4f16-8fe4-1a018f5ed00d`

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

Contoh berikut mencabut izin PENGGUNAAN pada from. `sales_db` Bob

```
REVOKE USAGE ON DATABASE sales_db FROM Bob;
```

Contoh berikut CABUT izin PENGGUNAAN pada `sales_schema` dari`Analyst_role`.

```
REVOKE USAGE ON SCHEMA sales_schema FROM ROLE Analyst_role;
```

Contoh pencabutan izin cakupan

Contoh berikut mencabut penggunaan untuk semua skema saat ini dan masa depan dalam `Sales_db` database dari peran. `Sales`

```
REVOKE USAGE FOR SCHEMAS IN DATABASE Sales_db FROM ROLE Sales;
```

Contoh berikut mencabut kemampuan untuk memberikan izin SELECT untuk semua tabel saat ini dan masa depan dalam `Sales_db` database dari pengguna. `alice` `alice`mempertahankan akses ke semua tabel di`Sales_db`.

```
REVOKE GRANT OPTION SELECT FOR TABLES IN DATABASE Sales_db FROM alice;
```

Contoh berikut mencabut izin EXECUTE untuk fungsi dalam `Sales_schema` skema dari pengguna. bob

```
REVOKE EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema FROM bob;
```

Contoh berikut mencabut semua izin untuk semua tabel dalam `ShareSchema` skema `ShareDb` database dari peran. `Sales` Saat menentukan skema, Anda juga dapat menentukan database skema menggunakan format `database.schema` dua bagian.

```
REVOKE ALL FOR TABLES IN SCHEMA ShareDb.ShareSchema FROM ROLE Sales;
```

Contoh berikut adalah sama dengan yang sebelumnya. Anda dapat menentukan database skema menggunakan `DATABASE` kata kunci alih-alih menggunakan format dua bagian.

```
REVOKE ALL FOR TABLES IN SCHEMA ShareSchema DATABASE ShareDb FROM ROLE Sales;
```

Contoh pencabutan hak istimewa ASSUMEROLE

Berikut ini adalah contoh pencabutan hak istimewa ASSUMEROLE.

Seorang superuser harus mengaktifkan penggunaan hak istimewa ASSUMEROLE untuk pengguna dan grup dengan menjalankan pernyataan berikut sekali di cluster:

```
revoke assumerole on all from public for all;
```

Pernyataan berikut mencabut hak istimewa ASSUMEROLE dari pengguna reg_user1 pada semua peran untuk semua operasi.

```
revoke assumerole on all from reg_user1 for all;
```

Contoh pencabutan hak istimewa ROLE

Contoh berikut mencabut sample_role1 dari ke sample_role2.

```
CREATE ROLE sample_role2;
GRANT ROLE sample_role1 TO ROLE sample_role2;
REVOKE ROLE sample_role1 FROM ROLE sample_role2;
```

Contoh berikut mencabut hak istimewa sistem dari user1.

```
GRANT ROLE sys:DBA TO user1;
REVOKE ROLE sys:DBA FROM user1;
```

Contoh berikut mencabut sample_role1 dan sample_role2 dari user1.

```
CREATE ROLE sample_role1;
CREATE ROLE sample_role2;
GRANT ROLE sample_role1, ROLE sample_role2 TO user1;
REVOKE ROLE sample_role1, ROLE sample_role2 FROM user1;
```

Contoh berikut mencabut sample_role2 dengan ADMIN OPTION dari user1.

```
GRANT ROLE sample_role2 TO user1 WITH ADMIN OPTION;
REVOKE ADMIN OPTION FOR ROLE sample_role2 FROM user1;
```

```
REVOKE ROLE sample_role2 FROM user1;
```

Contoh berikut mencabut sample_role1 dan sample_role2 dari sample_role5.

```
CREATE ROLE sample_role5;
GRANT ROLE sample_role1, ROLE sample_role2 TO ROLE sample_role5;
REVOKE ROLE sample_role1, ROLE sample_role2 FROM ROLE sample_role5;
```

Contoh berikut mencabut hak istimewa sistem CREATE SCHEMA dan DROP SCHEMA ke sample_role1.

```
GRANT CREATE SCHEMA, DROP SCHEMA TO ROLE sample_role1;
REVOKE CREATE SCHEMA, DROP SCHEMA FROM ROLE sample_role1;
```

ROLLBACK

Menghentikan transaksi saat ini dan membuang semua pembaruan yang dilakukan oleh transaksi tersebut.

Perintah ini melakukan fungsi yang sama dengan MEMBATALKAN perintah.

Sintaksis

```
ROLLBACK [ WORK | TRANSACTION ]
```

Parameter

BEKERJA

Kata kunci opsional. Kata kunci ini tidak didukung dalam prosedur tersimpan.

TRANSAKSI

Kata kunci opsional. KERJA dan TRANSAKSI adalah sinonim. Keduanya tidak didukung dalam prosedur tersimpan.

Untuk informasi tentang menggunakan ROLLBACK dalam prosedur tersimpan, lihatMengelola transaksi.

Contoh

Contoh berikut membuat tabel kemudian memulai transaksi di mana data dimasukkan ke dalam tabel. Perintah ROLLBACK kemudian memutar kembali penyisipan data untuk membiarkan tabel kosong.

Perintah berikut membuat tabel contoh yang disebut MOVIE_GROSS:

```
create table movie_gross( name varchar(30), gross bigint );
```

Kumpulan perintah berikutnya memulai transaksi yang menyisipkan dua baris data ke dalam tabel:

```
begin;

insert into movie_gross values ( 'Raiders of the Lost Ark', 23400000);

insert into movie_gross values ( 'Star Wars', 10000000 );
```

Selanjutnya, perintah berikut memilih data dari tabel untuk menunjukkan bahwa itu berhasil dimasukkan:

```
select * from movie_gross;
```

Output perintah menunjukkan bahwa kedua baris berhasil dimasukkan:

```
name           |  gross
-------------------------+----------
Raiders of the Lost Ark | 23400000
Star Wars               | 10000000
(2 rows)
```

Perintah ini sekarang mengembalikan perubahan data ke tempat transaksi dimulai:

```
rollback;
```

Memilih data dari tabel sekarang menunjukkan tabel kosong:

```
select * from movie_gross;

name | gross
```

```
------+-------
(0 rows)
```

SELECT

Mengembalikan baris dari tabel, tampilan, dan fungsi yang ditentukan pengguna.

> **Note**
>
> Ukuran maksimum untuk satu pernyataan SQL adalah 16 MB.

Sintaksis

```
[ WITH with_subquery [, ...] ]
SELECT
[ TOP number | [ ALL | DISTINCT ]
* | expression [ AS output_name ] [, ...] ]
[ EXCLUDE column_list ]
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ [ START WITH expression ] CONNECT BY expression ]
[ GROUP BY ALL | expression [, ...] ]
[ HAVING condition ]
[ QUALIFY condition ]
[ { UNION | ALL | INTERSECT | EXCEPT | MINUS } query ]
[ ORDER BY expression [ ASC | DESC ] ]
[ LIMIT { number | ALL } ]
[ OFFSET start ]
```

Topik

- DENGAN klausa
- PILIH daftar
- KECUALIKAN column_list
- Klausa FROM
- Klausa WHERE
- Klausa GROUP BY
- Klausa HAVING

- Klausul QUALIFY
- UNION, INTERSECT, dan KECUALI
- Klausa ORDER BY
- CONNECT BY klausa
- Contoh subquery
- Subkueri yang berkorelasi

DENGAN klausa

Klausa WITH adalah klausa opsional yang mendahului daftar SELECT dalam kueri. Klausa WITH mendefinisikan satu atau lebih common_table_expression. Setiap ekspresi tabel umum (CTE) mendefinisikan tabel sementara, yang mirip dengan definisi tampilan. Anda dapat mereferensikan tabel sementara ini dalam klausa FROM. Mereka hanya digunakan saat kueri tempat mereka berjalan. Setiap CTE dalam klausa WITH menentukan nama tabel, daftar opsional nama kolom, dan ekspresi kueri yang mengevaluasi ke tabel (pernyataan SELECT). Ketika Anda mereferensikan nama tabel sementara dalam klausa FROM dari ekspresi kueri yang sama yang mendefinisikannya, CTE bersifat rekursif.

WITH klausa subquery adalah cara yang efisien untuk mendefinisikan tabel yang dapat digunakan sepanjang eksekusi kueri tunggal. Dalam semua kasus, hasil yang sama dapat dicapai dengan menggunakan subquery di badan utama pernyataan SELECT, tetapi dengan subquery klausa mungkin lebih sederhana untuk ditulis dan dibaca. Jika memungkinkan, subkueri klausa WITH yang direferensikan beberapa kali dioptimalkan sebagai subexpresi umum; yaitu, dimungkinkan untuk mengevaluasi subkueri WITH sekali dan menggunakan kembali hasilnya. (Perhatikan bahwa subekpresi umum tidak terbatas pada yang didefinisikan dalam klausa WITH.)

Sintaksis

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
```

dimana common_table_expression dapat berupa non-rekursif atau rekursif. Berikut ini adalah bentuk non-rekursif:

```
CTE_table_name [ ( column_name [, ...] ) ] AS ( query )
```

Berikut ini adalah bentuk rekursif dari common _table_expression:

```
CTE_table_name (column_name [, ...] ) AS ( recursive_query )
```

Parameter

REKURSIF

Kata kunci yang mengidentifikasi query sebagai CTE rekursif. Kata kunci ini diperlukan jika common_table_expression yang didefinisikan dalam klausa WITH bersifat rekursif. Anda hanya dapat menentukan kata kunci RECURSIVE sekali, segera mengikuti kata kunci WITH, bahkan ketika klausa WITH berisi beberapa CTE rekursif. Secara umum, CTE rekursif adalah subquery UNION ALL dengan dua bagian.

Ekspresi tabel_umum

Mendefinisikan tabel sementara yang dapat Anda referensikan Klausa FROM dan hanya digunakan selama pelaksanaan kueri yang menjadi miliknya.

Nama Tabel_CTE

Nama unik untuk tabel sementara yang mendefinisikan hasil subquery klausa WITH. Anda tidak dapat menggunakan nama duplikat dalam satu klausa WITH. Setiap subquery harus diberi nama tabel yang dapat direferensikan diKlausa FROM.

nama_kolom

Daftar nama kolom keluaran untuk subquery klausa WITH, dipisahkan oleh koma. Jumlah nama kolom yang ditentukan harus sama dengan atau kurang dari jumlah kolom yang ditentukan oleh subquery. Untuk CTE yang non-rekursif, klausa column _name bersifat opsional. Untuk CTE rekursif, daftar column_ name diperlukan.

query

Setiap kueri SELECT yang didukung Amazon Redshift. Lihat SELECT.

rekursif_query

Kueri UNION ALL yang terdiri dari dua subquery SELECT:

- Subkueri SELECT pertama tidak memiliki referensi rekursif ke CTE_table_name yang sama. Ini mengembalikan set hasil yang merupakan benih awal dari rekursi. Bagian ini disebut anggota awal atau anggota benih.
- Subquery SELECT kedua mereferensikan CTE_table_name yang sama dalam klausa FROM-nya. Ini disebut anggota rekursif. Recursive_query berisi kondisi WHERE untuk mengakhiri recursive_query.

Catatan penggunaan

Anda dapat menggunakan klausa WITH dalam pernyataan SQL berikut:

- SELECT
- PILIH KE
- CREATE TABLE AS
- CREATE VIEW
- MENYATAKAN
- EXPLAIN
- MASUKKAN KE... PILIH
- MEMPERSIAPKAN
- UPDATE (dalam subkueri klausa WHERE. Anda tidak dapat mendefinisikan CTE rekursif di subquery. CTE rekursif harus mendahului klausa UPDATE.)
- DELETE

Jika klausa FROM dari query yang berisi klausa WITH tidak mereferensikan salah satu tabel yang ditentukan oleh klausa WITH, klausa WITH diabaikan dan query berjalan seperti biasa.

Tabel yang ditentukan oleh subkueri klausa WITH dapat direferensikan hanya dalam lingkup query SELECT yang dimulai klausa WITH. Misalnya, Anda dapat mereferensikan tabel tersebut dalam klausa FROM dari subkueri dalam daftar SELECT, klausa WHERE, atau klausa HAVING. Anda tidak dapat menggunakan klausa WITH dalam subquery dan mereferensikan tabelnya dalam klausa FROM dari query utama atau subkueri lain. Pola kueri ini menghasilkan pesan kesalahan dari formulir `relation table_name doesn't exist` untuk tabel klausa WITH.

Anda tidak dapat menentukan klausa WITH lain di dalam subquery klausa WITH.

Anda tidak dapat membuat referensi ke tabel yang ditentukan oleh subkueri klausa WITH. Misalnya, kueri berikut mengembalikan kesalahan karena referensi maju ke tabel W2 dalam definisi tabel W1:

```
with w1 as (select * from w2), w2 as (select * from w1)
select * from sales;
ERROR:  relation "w2" does not exist
```

Subquery klausa WITH mungkin tidak terdiri dari pernyataan SELECT INTO; namun, Anda dapat menggunakan klausa WITH dalam pernyataan SELECT INTO.

Ekspresi tabel umum rekursif

Sebuah ekspresi tabel umum rekursif (CTE) adalah CTE yang mereferensikan dirinya sendiri. CTE rekursif berguna dalam menanyakan data hierarkis, seperti bagan organisasi yang menunjukkan hubungan pelaporan antara karyawan dan manajer. Lihat Contoh: CTE Rekursif.

Penggunaan umum lainnya adalah daftar bahan bertingkat, ketika suatu produk terdiri dari banyak komponen dan setiap komponen itu sendiri juga terdiri dari komponen atau subrakitan lain.

Pastikan untuk membatasi kedalaman rekursi dengan memasukkan klausa WHERE di subquery SELECT kedua dari kueri rekursif. Sebagai contoh, lihat Contoh: CTE Rekursif. Jika tidak, kesalahan dapat terjadi mirip dengan berikut ini:

- `Recursive CTE out of working buffers.`
- `Exceeded recursive CTE max rows limit, please add correct CTE termination predicates or change the max_recursion_rows parameter.`

> **Note**
>
> `max_recursion_rows`adalah parameter yang mengatur jumlah baris maksimum yang dapat dikembalikan oleh CTE rekursif untuk mencegah loop rekursi tak terbatas. Kami menyarankan agar tidak mengubah ini ke nilai yang lebih besar dari default. Ini mencegah masalah rekursi tak terbatas dalam kueri Anda mengambil ruang yang berlebihan di cluster Anda.

Anda dapat menentukan urutan pengurutan dan membatasi hasil CTE rekursif. Anda dapat menyertakan opsi grup berdasarkan dan berbeda pada hasil akhir dari CTE rekursif.

Anda tidak dapat menentukan klausa WITH RECURSIVE di dalam subquery. Anggota recursive_query tidak dapat menyertakan klausa order by atau limit.

Contoh

Contoh berikut menunjukkan kasus paling sederhana dari kueri yang berisi klausa WITH. Kueri WITH bernama VENUECOPY memilih semua baris dari tabel VENUE. Kueri utama pada gilirannya memilih semua baris dari VENUECOPY. Tabel VENUECOPY hanya ada selama kueri ini.

```
with venuecopy as (select * from venue)
```

```
select * from venuecopy order by 1 limit 10;
```

```
 venueid |         venuename          |    venuecity    | venuestate | venueseats
---------+----------------------------+-----------------+------------+------------
1 | Toyota Park                | Bridgeview      | IL         |          0
2 | Columbus Crew Stadium      | Columbus        | OH         |          0
3 | RFK Stadium                | Washington      | DC         |          0
4 | CommunityAmerica Ballpark  | Kansas City     | KS         |          0
5 | Gillette Stadium           | Foxborough      | MA         |      68756
6 | New York Giants Stadium    | East Rutherford | NJ         |      80242
7 | BMO Field                  | Toronto         | ON         |          0
8 | The Home Depot Center      | Carson          | CA         |          0
9 | Dick's Sporting Goods Park | Commerce City   | CO         |          0
v     10 | Pizza Hut Park             | Frisco          | TX         |          0
(10 rows)
```

Contoh berikut menunjukkan klausa WITH yang menghasilkan dua tabel, bernama VENUE_SALES dan TOP_VENUES. Tabel kueri WITH kedua memilih dari yang pertama. Pada gilirannya, klausa WHERE dari blok kueri utama berisi subquery yang membatasi tabel TOP_VENUES.

```
with venue_sales as
(select venuename, venuecity, sum(pricepaid) as venuename_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
group by venuename, venuecity),

top_venues as
(select venuename
from venue_sales
where venuename_sales > 800000)

select venuename, venuecity, venuestate,
sum(qtysold) as venue_qty,
sum(pricepaid) as venue_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
and venuename in(select venuename from top_venues)
group by venuename, venuecity, venuestate
order by venuename;
```

```
       venuename       |   venuecity   | venuestate | venue_qty | venue_sales
-----------------------+---------------+------------+-----------+-------------
```

```
August Wilson Theatre   | New York City | NY         |      3187 |  1032156.00
Biltmore Theatre        | New York City | NY         |      2629 |   828981.00
Charles Playhouse       | Boston        | MA         |      2502 |   857031.00
Ethel Barrymore Theatre | New York City | NY         |      2828 |   891172.00
Eugene O'Neill Theatre  | New York City | NY         |      2488 |   828950.00
Greek Theatre           | Los Angeles   | CA         |      2445 |   838918.00
Helen Hayes Theatre     | New York City | NY         |      2948 |   978765.00
Hilton Theatre          | New York City | NY         |      2999 |   885686.00
Imperial Theatre        | New York City | NY         |      2702 |   877993.00
Lunt-Fontanne Theatre   | New York City | NY         |      3326 |  1115182.00
Majestic Theatre        | New York City | NY         |      2549 |   894275.00
Nederlander Theatre     | New York City | NY         |      2934 |   936312.00
Pasadena Playhouse      | Pasadena      | CA         |      2739 |   820435.00
Winter Garden Theatre   | New York City | NY         |      2838 |   939257.00
(14 rows)
```

Dua contoh berikut menunjukkan aturan untuk ruang lingkup referensi tabel berdasarkan subquery klausa WITH. Kueri pertama berjalan, tetapi yang kedua gagal dengan kesalahan yang diharapkan. Kueri pertama memiliki subquery klausa WITH di dalam daftar SELECT dari query utama. Tabel yang ditentukan oleh klausa WITH (HOLIDAYS) direferensikan dalam klausa FROM subkueri dalam daftar SELECT:

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t')
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join date on sales.dateid=date.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;

caldate   | daysales | dec25sales
-----------+----------+------------
2008-12-25 | 70402.00 |   70402.00
2008-12-31 | 12678.00 |   70402.00
(2 rows)
```

Kueri kedua gagal karena mencoba untuk mereferensikan tabel HOLIDAYS di kueri utama serta di subkueri daftar SELECT. Referensi kueri utama berada di luar cakupan.

```
select caldate, sum(pricepaid) as daysales,
```

```
(with holidays as (select * from date where holiday ='t')
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join holidays on sales.dateid=holidays.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;

ERROR:  relation "holidays" does not exist
```

Contoh: CTE Rekursif

Berikut ini adalah contoh CTE rekursif yang mengembalikan karyawan yang melaporkan secara langsung atau tidak langsung kepada John. Kueri rekursif berisi klausa WHERE untuk membatasi kedalaman rekursi kurang dari 4 level.

```
--create and populate the sample table
  create table employee (
  id int,
  name varchar (20),
  manager_id int
  );

  insert into employee(id, name, manager_id)  values
(100, 'Carlos', null),
(101, 'John', 100),
(102, 'Jorge', 101),
(103, 'Kwaku', 101),
(110, 'Liu', 101),
(106, 'Mateo', 102),
(110, 'Nikki', 103),
(104, 'Paulo', 103),
(105, 'Richard', 103),
(120, 'Saanvi', 104),
(200, 'Shirley', 104),
(201, 'Sofía', 102),
(205, 'Zhang', 104);

--run the recursive query
  with recursive john_org(id, name, manager_id, level) as
( select id, name, manager_id, 1 as level
  from employee
```

```
  where name = 'John'
  union all
  select e.id, e.name, e.manager_id, level + 1 as next_level
  from employee e, john_org j
  where e.manager_id = j.id and level < 4
  )
select distinct id, name, manager_id from john_org order by manager_id;
```

Berikut ini adalah hasil dari query.

```
  id        name      manager_id
------+-----------+--------------
 101    John           100
 102    Jorge          101
 103    Kwaku          101
 110    Liu            101
 201    Sofía          102
 106    Mateo          102
 110    Nikki          103
 104    Paulo          103
 105    Richard        103
 120    Saanvi         104
 200    Shirley        104
 205    Zhang          104
```

Berikut ini adalah bagan organisasi untuk departemen John.

PILIH daftar

Topik

- Sintaksis
- Parameter
- Catatan penggunaan
- Contoh

Daftar SELECT memberi nama kolom, fungsi, dan ekspresi yang ingin kueri dikembalikan. Daftar ini mewakili output kueri.

Untuk informasi selengkapnya tentang fungsi SQL, lihatReferensi fungsi SQL. Untuk informasi selengkapnya tentang ekspresi, lihatEkspresi bersyarat.

Sintaksis

```
SELECT
[ TOP number ]
[ ALL | DISTINCT ] * | expression [ AS column_alias ] [, ...]
```

Parameter

Nomor TOP

TOP mengambil integer positif sebagai argumennya, yang mendefinisikan jumlah baris yang dikembalikan ke klien. Perilaku dengan klausa TOP sama dengan perilaku dengan klausa LIMIT. Jumlah baris yang dikembalikan tetap, tetapi kumpulan baris tidak. Untuk mengembalikan set baris yang konsisten, gunakan TOP atau LIMIT bersama dengan klausa ORDER BY.

SEMUA

Kata kunci redundan yang mendefinisikan perilaku default jika Anda tidak menentukan DISTINCT. `SELECT ALL *`berarti sama dengan `SELECT *` (pilih semua baris untuk semua kolom dan pertahankan duplikat).

DISTINCT

Opsi yang menghilangkan baris duplikat dari kumpulan hasil, berdasarkan nilai yang cocok dalam satu atau lebih kolom.

> **Note**
>
> Jika aplikasi Anda mengizinkan kunci asing atau kunci utama yang tidak valid, itu dapat menyebabkan kueri mengembalikan hasil yang salah. Misalnya, kueri SELECT DISTINCT mungkin mengembalikan baris duplikat jika kolom kunci utama tidak berisi semua nilai unik. Untuk informasi selengkapnya, lihat Mendefinisikan batasan tabel.

* (tanda bintang)

Mengembalikan seluruh isi tabel (semua kolom dan semua baris).

ekspresi

Ekspresi yang dibentuk dari satu atau lebih kolom yang ada dalam tabel yang direferensikan oleh query. Sebuah ekspresi dapat berisi fungsi SQL. Contoh:

```
avg(datediff(day, listtime, saletime))
```

AS kolumn_alias

Nama sementara untuk kolom yang digunakan dalam kumpulan hasil akhir. Kata kunci AS adalah opsional. Contoh:

```
avg(datediff(day, listtime, saletime)) as avgwait
```

Jika Anda tidak menentukan alias untuk ekspresi yang bukan nama kolom sederhana, kumpulan hasil akan menerapkan nama default ke kolom tersebut.

> **Note**
>
> Alias dikenali tepat setelah didefinisikan dalam daftar target. Anda dapat menggunakan alias dalam ekspresi lain yang ditentukan setelahnya dalam daftar target yang sama. Contoh berikut menggambarkan hal ini.
>
> ```
> select clicks / impressions as probability, round(100 * probability, 1) as
>  percentage from raw_data;
> ```
>
> Manfaat referensi alias lateral adalah Anda tidak perlu mengulangi ekspresi alias saat membangun ekspresi yang lebih kompleks dalam daftar target yang sama. Saat Amazon Redshift mengurai jenis referensi ini, itu hanya memasukkan alias yang ditentukan sebelumnya. Jika ada kolom dengan nama yang sama yang didefinisikan dalam FROM klausa sebagai ekspresi alias sebelumnya, kolom dalam FROM klausa akan diprioritaskan. Misalnya, dalam kueri di atas jika ada kolom bernama 'probabilitas' dalam tabel raw_data, 'probabilitas' dalam ekspresi kedua dalam daftar target mengacu pada kolom itu alih-alih nama alias 'probabilitas'.

Catatan penggunaan

TOP adalah ekstensi SQL; ini memberikan alternatif untuk perilaku LIMIT. Anda tidak dapat menggunakan TOP dan LIMIT dalam kueri yang sama.

Contoh

Contoh berikut mengembalikan 10 baris dari tabel SALES. Meskipun kueri menggunakan klausa TOP, ia masih mengembalikan kumpulan baris yang tidak dapat diprediksi karena tidak ada klausa ORDER BY yang ditentukan,

```
select top 10 *
from sales;
```

Kueri berikut secara fungsional setara, tetapi menggunakan klausa LIMIT alih-alih klausa TOP:

```
select *
from sales
limit 10;
```

Contoh berikut mengembalikan 10 baris pertama dari tabel SALES menggunakan klausa TOP, diurutkan berdasarkan kolom QTYSOLD dalam urutan menurun.

```
select top 10 qtysold, sellerid
from sales
order by qtysold desc, sellerid;

qtysold | sellerid
--------+----------
8 |      518
8 |      520
8 |      574
8 |      718
8 |      868
8 |     2663
8 |     3396
8 |     3726
8 |     5250
8 |     6216
(10 rows)
```

Contoh berikut mengembalikan dua nilai QTYSOLD dan SELLERID pertama dari tabel SALES, diurutkan berdasarkan kolom QTYSOLD:

```
select top 2 qtysold, sellerid
from sales
order by qtysold desc, sellerid;

qtysold | sellerid
--------+----------
8 |      518
8 |      520
(2 rows)
```

Contoh berikut menunjukkan daftar grup kategori yang berbeda dari tabel CATEGORY:

```
select distinct catgroup from category
order by 1;

catgroup
----------
Concerts
Shows
Sports
(3 rows)

--the same query, run without distinct
select catgroup from category
order by 1;

catgroup
----------
Concerts
Concerts
Concerts
Shows
Shows
Shows
Sports
Sports
Sports
Sports
Sports
```

```
(11 rows)
```

Contoh berikut mengembalikan set nomor minggu yang berbeda untuk Desember 2008. Tanpa klausa DISTINCT, pernyataan akan mengembalikan 31 baris, atau satu untuk setiap hari dalam sebulan.

```
select distinct week, month, year
from date
where month='DEC' and year=2008
order by 1, 2, 3;

week | month | year
-----+-------+------
49 | DEC   | 2008
50 | DEC   | 2008
51 | DEC   | 2008
52 | DEC   | 2008
53 | DEC   | 2008
(5 rows)
```

KECUALIKAN column_list

EXCLUDE column_list memberi nama kolom yang dikecualikan dari hasil kueri. Menggunakan opsi EXCLUDE sangat membantu ketika hanya subset kolom yang perlu dikecualikan dari tabel lebar, yang merupakan tabel yang berisi banyak kolom.

Topik

- Sintaksis
- Parameter
- Contoh

Sintaksis

```
EXCLUDE column_list
```

Parameter

column_list

Daftar yang dipisahkan koma dari satu atau lebih nama kolom yang ada dalam tabel yang direferensikan oleh kueri. Column_list secara opsional dapat ditutup dalam tanda kurung. Hanya nama kolom yang didukung dalam daftar pengecualian nama kolom, bukan ekspresi (sepertiupper(col1)) atau tanda bintang (*).

```
column-name, ... | ( column-name, ... )
```

Contoh:

```
SELECT * EXCLUDE col1, col2 FROM tablea;
```

```
SELECT * EXCLUDE (col1, col2) FROM tablea;
```

Contoh

Contoh berikut menggunakan tabel SALES yang berisi kolom: salesid, listid, sellerid, buyerid, eventid, dateid, qtysold, pricepaid, commission, dan saletime. Untuk informasi selengkapnya tentang tabel PENJUALAN, lihatDatabase sampel.

Contoh berikut mengembalikan baris dari tabel SALES, tetapi tidak termasuk kolom SALETIME.

```
SELECT * EXCLUDE saletime FROM sales;

salesid | listid  | sellerid | buyerid | eventid | dateid  | qtysold  | pricepaid  |
 commission
--------+---------+----------+---------+---------+---------+----------+------------
+-----------
150314  | 173969  | 48680    | 816     | 8762    | 1827    | 2        | 688        |
 103.2
8325    | 8942    | 23600    | 1078    | 2557    | 1828    | 5        | 525        |
 78.75
46807   | 52711   | 34388    | 1047    | 2046    | 1828    | 2        | 482        |
 72.3
...
```

Contoh berikut mengembalikan baris dari tabel SALES, tetapi tidak termasuk kolom QTYSOLD dan SALETIME.

```
SELECT * EXCLUDE (qtysold, saletime) FROM sales;

salesid | listid  | sellerid | buyerid | eventid | dateid  | pricepaid  | commission
--------+---------+----------+---------+---------+---------+------------+-----------
150314  | 173969  | 48680    | 816     | 8762    | 1827    | 688        | 103.2
8325    | 8942    | 23600    | 1078    | 2557    | 1828    | 525        |  78.75
46807   | 52711   | 34388    | 1047    | 2046    | 1828    | 482        |  72.3
...
```

Contoh berikut membuat tampilan yang mengembalikan baris dari tabel SALES, tetapi mengecualikan kolom SALETIME.

```
CREATE VIEW sales_view AS SELECT * EXCLUDE saletime FROM sales;
SELECT * FROM sales_view;

salesid | listid  | sellerid | buyerid | eventid | dateid  | qtysold  | pricepaid  |
 commission
--------+---------+----------+---------+---------+---------+----------+------------
+-----------
150314  | 173969  | 48680    | 816     | 8762    | 1827    | 2        | 688        |
 103.2
8325    | 8942    | 23600    | 1078    | 2557    | 1828    | 5        | 525        |
 78.75
46807   | 52711   | 34388    | 1047    | 2046    | 1828    | 2        | 482        |
 72.3
...
```

Contoh berikut hanya memilih kolom yang tidak dikecualikan ke dalam tabel temp.

```
SELECT * EXCLUDE saletime INTO TEMP temp_sales FROM sales;
SELECT * FROM temp_sales;

salesid | listid  | sellerid | buyerid | eventid | dateid  | qtysold  | pricepaid  |
 commission
--------+---------+----------+---------+---------+---------+----------+------------
+-----------
150314  | 173969  | 48680    | 816     | 8762    | 1827    | 2        | 688        |
 103.2
```

```
8325    | 8942    | 23600    | 1078    | 2557    | 1828    | 5        | 525        |
 78.75
46807   | 52711   | 34388    | 1047    | 2046    | 1828    | 2        | 482        |
 72.3
...
```

Klausa FROM

Klausa FROM dalam kueri mencantumkan referensi tabel (tabel, tampilan, dan subkueri) dari mana data dipilih. Jika beberapa referensi tabel terdaftar, tabel harus digabungkan, menggunakan sintaks yang sesuai baik dalam klausa FROM atau klausa WHERE. Jika tidak ada kriteria gabungan yang ditentukan, sistem memproses kueri sebagai penggabungan silang (produk Cartesian).

Topik

- Sintaksis
- Parameter
- Catatan penggunaan
- Contoh PIVOT dan UNPIVOT
- Contoh JOIN
- Contoh UNNEST

Sintaksis

```
FROM table_reference [, ...]
```

dimana table_reference adalah salah satu dari berikut ini:

```
with_subquery_table_name [ table_alias ]
table_name [ * ] [ table_alias ]
( subquery ) [ table_alias ]
table_reference [ NATURAL ] join_type table_reference
   [ ON join_condition | USING ( join_column [, ...] ) ]
table_reference  join_type super_expression
   [ ON join_condition ]
table_reference PIVOT (
   aggregate(expr) [ [ AS ] aggregate_alias ]
   FOR column_name IN ( expression [ AS ] in_alias [, ...] )
) [ table_alias ]
table_reference UNPIVOT [ INCLUDE NULLS | EXCLUDE NULLS ] (
```

```
   value_column_name
   FOR name_column_name IN ( column_reference [ [ AS ]
   in_alias ] [, ...] )
) [ table_alias ]
UNPIVOT expression AS value_alias [ AT attribute_alias ]
( super_expression.attribute_name ) AS value_alias [ AT index_alias ]
UNNEST ( column_reference )
  [AS] table_alias ( unnested_column_name )
UNNEST ( column_reference ) WITH OFFSET
  [AS] table_alias ( unnested_column_name, [offset_column_name] )
```

Table_alias opsional dapat digunakan untuk memberikan nama sementara ke tabel dan referensi tabel kompleks dan, jika diinginkan, kolom mereka juga, seperti berikut ini:

```
[ AS ] alias [ ( column_alias [, ...] ) ]
```

Parameter

dengan_subkueri_nama_tabel

Tabel yang ditentukan oleh subquery diDENGAN klausa.

table_name

Nama tabel atau tampilan.

alias

Nama alternatif sementara untuk tabel atau tampilan. Alias harus disediakan untuk tabel yang berasal dari subquery. Dalam referensi tabel lain, alias bersifat opsional. Kata kunci AS selalu opsional. Alias tabel menyediakan pintasan yang nyaman untuk mengidentifikasi tabel di bagian lain dari kueri, seperti klausa WHERE. Contoh:

```
select * from sales s, listing l
where s.listid=l.listid
```

kolom_alias

Nama alternatif sementara untuk kolom dalam tabel atau tampilan.

subkueri

Ekspresi query yang mengevaluasi ke tabel. Tabel hanya ada selama durasi kueri dan biasanya diberi nama atau alias. Namun, alias tidak diperlukan. Anda juga dapat menentukan nama

kolom untuk tabel yang berasal dari subquery. Penamaan alias kolom penting ketika Anda ingin menggabungkan hasil subquery ke tabel lain dan ketika Anda ingin memilih atau membatasi kolom tersebut di tempat lain dalam kueri.

Subquery mungkin berisi klausa ORDER BY, tetapi klausa ini mungkin tidak berpengaruh jika klausa LIMIT atau OFFSET juga tidak ditentukan.

ALAMI

Mendefinisikan gabungan yang secara otomatis menggunakan semua pasangan kolom dengan nama identik dalam dua tabel sebagai kolom penggabungan. Tidak diperlukan kondisi gabungan eksplisit. Misalnya, jika tabel CATEGORY dan EVENT keduanya memiliki kolom bernama CATID, gabungan alami dari tabel tersebut adalah gabungan di atas kolom CATID mereka.

> **Note**
>
> Jika gabungan NATURAL ditentukan tetapi tidak ada pasangan kolom dengan nama identik yang ada di tabel yang akan digabungkan, kueri default ke cross-join.

tipe gabungan

Tentukan salah satu jenis gabungan berikut:

- [BATIN] BERGABUNG
- KIRI [LUAR] BERGABUNG
- KANAN [LUAR] BERGABUNG
- BERGABUNG PENUH [LUAR]
- CROSS JOIN

Cross-joins adalah gabungan yang tidak memenuhi syarat; mereka mengembalikan produk Cartesian dari dua tabel.

Gabungan dalam dan luar adalah gabungan yang memenuhi syarat. Mereka memenuhi syarat baik secara implisit (dalam gabungan alami); dengan sintaks ON atau USING dalam klausa FROM; atau dengan kondisi klausa WHERE.

Gabungan dalam mengembalikan baris yang cocok saja, berdasarkan kondisi gabungan atau daftar kolom penggabungan. Gabungan luar mengembalikan semua baris yang akan dikembalikan oleh gabungan dalam yang setara ditambah baris yang tidak cocok dari tabel “kiri”,

tabel “kanan”, atau kedua tabel. Tabel kiri adalah tabel yang terdaftar pertama, dan tabel kanan adalah tabel yang terdaftar kedua. Baris yang tidak cocok berisi nilai NULL untuk mengisi celah di kolom keluaran.

ON join_condition

Jenis spesifikasi gabungan di mana kolom penggabungan dinyatakan sebagai kondisi yang mengikuti kata kunci ON. Contoh:

```
sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
```

MENGGUNAKAN (join_column [,...])

Jenis spesifikasi gabungan di mana kolom penggabungan terdaftar dalam tanda kurung. Jika beberapa kolom penggabungan ditentukan, mereka dibatasi oleh koma. Kata kunci USING harus mendahului daftar. Contoh:

```
sales join listing
using (listid,eventid)
```

POROS

Memutar output dari baris ke kolom, untuk tujuan mewakili data tabular dalam format yang mudah dibaca. Output diwakili secara horizontal di beberapa kolom. PIVOT mirip dengan kueri GROUP BY dengan agregasi, menggunakan ekspresi agregat untuk menentukan format output. Namun, berbeda dengan GROUP BY, hasilnya dikembalikan dalam kolom, bukan baris.

Untuk contoh yang menunjukkan cara membuat kueri dengan PIVOT dan UNPIVOT, lihatContoh PIVOT dan UNPIVOT.

TIDAK BERPUTAR

Memutar kolom menjadi baris dengan UNPIVOT - Operator mengubah kolom hasil, dari tabel input atau hasil kueri, menjadi baris, untuk membuat output lebih mudah dibaca. UNPIVOT menggabungkan data kolom input menjadi dua kolom hasil: kolom nama dan kolom nilai. Kolom nama berisi nama kolom dari input, sebagai entri baris. Kolom nilai berisi nilai dari kolom input, seperti hasil agregasi. Misalnya, jumlah item dalam berbagai kategori.

Object unpioting dengan UNPIVOT (SUPER) — Anda dapat melakukan pembongkaran objek, di mana ekspresi adalah ekspresi SUPER yang mengacu pada item klausa FROM lainnya. Untuk

informasi selengkapnya, lihat Objek tidak berputar. Ini juga memiliki contoh yang menunjukkan cara menanyakan data semi-terstruktur, seperti data yang ada JSON-formatted.

super_ekspresi

Ekspresi SUPER yang valid. Amazon Redshift mengembalikan satu baris untuk setiap nilai dalam atribut yang ditentukan. Untuk informasi selengkapnya tentang tipe data SUPER, lihatJenis SUPER. Untuk informasi selengkapnya tentang nilai SUPER yang tidak bersarang, lihat. Kueri tanpa sarang

nama atribut

Nama atribut dalam ekspresi SUPER.

Indeks_alias

Alias untuk indeks yang menandakan posisi nilai dalam ekspresi SUPER.

UNNEST

Memperluas struktur bersarang, biasanya array SUPER, menjadi kolom yang berisi elemen yang tidak bersarang. Untuk informasi selengkapnya tentang menghapus data SUPER, lihatMengkueri data semi-terstruktur. Sebagai contoh, lihat Contoh UNNEST.

nama_kolom tidak bersarang

Nama kolom yang berisi elemen yang tidak bersarang.

UNEST... DENGAN OFFSET

Menambahkan kolom offset ke output yang tidak bersarang, dengan offset mewakili indeks berbasis nol dari setiap elemen dalam array. Varian ini berguna ketika Anda ingin melihat posisi elemen dalam array. Untuk informasi selengkapnya tentang menghapus data SUPER, lihatMengkueri data semi-terstruktur. Sebagai contoh, lihat Contoh UNNEST.

offset_nama_kolom

Nama kustom untuk kolom offset yang Anda gunakan untuk secara eksplisit menentukan bagaimana kolom indeks akan muncul dalam output. Parameter ini bersifat opsional. Secara default, nama kolom offset adalah`offset_col`.

Catatan penggunaan

Kolom penggabungan harus memiliki tipe data yang sebanding.

Gabungan NATURAL atau USING hanya mempertahankan satu dari setiap pasangan kolom penggabungan dalam kumpulan hasil perantara.

Gabungan dengan sintaks ON mempertahankan kedua kolom penggabungan dalam kumpulan hasil perantara.

Lihat juga DENGAN klausa.

Contoh PIVOT dan UNPIVOT

PIVOT dan UNPIVOT adalah parameter dalam klausa FROM yang memutar output kueri dari baris ke kolom dan kolom ke baris, masing-masing. Mereka mewakili hasil kueri tabular dalam format yang mudah dibaca. Contoh berikut menggunakan data uji dan kueri untuk menunjukkan cara menggunakannya.

Untuk informasi lebih lanjut tentang ini dan parameter lainnya, lihat kla usa FROM.

Contoh PIVOT

Siapkan tabel sampel dan data dan gunakan untuk menjalankan kueri contoh berikutnya.

```
CREATE TABLE part (
    partname varchar,
    manufacturer varchar,
    quality int,
    price decimal(12, 2)
);

INSERT INTO part VALUES ('prop', 'local parts co', 2, 10.00);
INSERT INTO part VALUES ('prop', 'big parts co', NULL, 9.00);
INSERT INTO part VALUES ('prop', 'small parts co', 1, 12.00);

INSERT INTO part VALUES ('rudder', 'local parts co', 1, 2.50);
INSERT INTO part VALUES ('rudder', 'big parts co', 2, 3.75);
INSERT INTO part VALUES ('rudder', 'small parts co', NULL, 1.90);

INSERT INTO part VALUES ('wing', 'local parts co', NULL, 7.50);
INSERT INTO part VALUES ('wing', 'big parts co', 1, 15.20);
INSERT INTO part VALUES ('wing', 'small parts co', NULL, 11.80);
```

PIVOT `partname` aktif dengan `AVG` agregasi aktif`price`.

```
SELECT *
```

```
FROM (SELECT partname, price FROM part) PIVOT (
    AVG(price) FOR partname IN ('prop', 'rudder', 'wing')
);
```

Kueri menghasilkan output berikut.

```
  prop   |  rudder  |  wing
---------+----------+---------
 10.33   | 2.71     | 11.50
```

Pada contoh sebelumnya, hasilnya diubah menjadi kolom. Contoh berikut menunjukkan ku GROUP BY eri yang mengembalikan harga rata-rata dalam baris, bukan dalam kolom.

```
SELECT partname, avg(price)
FROM (SELECT partname, price FROM part)
WHERE partname IN ('prop', 'rudder', 'wing')
GROUP BY partname;
```

Kueri menghasilkan output berikut.

```
 partname |  avg
----------+-------
 prop     | 10.33
 rudder   |  2.71
 wing     | 11.50
```

PIVOTContoh dengan `manufacturer` kolom implisit.

```
SELECT *
FROM (SELECT quality, manufacturer FROM part) PIVOT (
    count(*) FOR quality IN (1, 2, NULL)
);
```

Kueri menghasilkan output berikut.

```
 manufacturer      | 1  | 2  | null
-------------------+----+----+------
 local parts co    | 1  | 1  |  1
 big parts co      | 1  | 1  |  1
 small parts co    | 1  | 0  |  2
```

Kolom tabel masukan yang tidak direferensikan dalam PIVOT definisi ditambahkan secara implisit ke tabel hasil. Ini adalah kasus untuk manufacturer kolom dalam contoh sebelumnya. Contoh juga menunjukkan bahwa itu NULL adalah nilai yang valid untuk IN operator.

PIVOTdalam contoh di atas mengembalikan informasi yang sama seperti kueri berikut, yang meliputiGROUP BY. Perbedaannya adalah PIVOT mengembalikan nilai 0 untuk kolom 2 dan pabrikansmall parts co. Ku GROUP BY eri tidak berisi baris yang sesuai. Dalam kebanyakan kasus, PIVOT menyisipkan NULL jika baris tidak memiliki data input untuk kolom tertentu. Namun, agregat hitungan tidak kembali NULL dan 0 merupakan nilai default.

```
SELECT manufacturer, quality, count(*)
FROM (SELECT quality, manufacturer FROM part)
WHERE quality IN (1, 2) OR quality IS NULL
GROUP BY manufacturer, quality
ORDER BY manufacturer;
```

Kueri menghasilkan output berikut.

```
 manufacturer        | quality | count
---------------------+---------+-------
 big parts co        |         |     1
 big parts co        |       2 |     1
 big parts co        |       1 |     1
 local parts co      |       2 |     1
 local parts co      |       1 |     1
 local parts co      |         |     1
 small parts co      |       1 |     1
 small parts co      |         |     2
```

Operator PIVOT menerima alias opsional pada ekspresi agregat dan pada setiap nilai untuk IN operator. Gunakan alias untuk menyesuaikan nama kolom. Jika tidak ada alias agregat, hanya IN daftar alias yang digunakan. Jika tidak, alias agregat ditambahkan ke nama kolom dengan garis bawah untuk memisahkan nama.

```
SELECT *
FROM (SELECT quality, manufacturer FROM part) PIVOT (
    count(*) AS count FOR quality IN (1 AS high, 2 AS low, NULL AS na)
);
```

Kueri menghasilkan output berikut.

```
 manufacturer      | high_count  | low_count | na_count
-------------------+-------------+-----------+----------
 local parts co    |           1 |         1 |        1
 big parts co      |           1 |         1 |        1
 small parts co    |           1 |         0 |        2
```

Siapkan tabel sampel berikut dan data dan gunakan untuk menjalankan kueri contoh berikutnya. Data menunjukkan tanggal pemesanan untuk koleksi hotel.

```
CREATE TABLE bookings (
    booking_id int,
    hotel_code char(8),
    booking_date date,
    price decimal(12, 2)
);

INSERT INTO bookings VALUES (1, 'FOREST_L', '02/01/2023', 75.12);
INSERT INTO bookings VALUES (2, 'FOREST_L', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (3, 'FOREST_L', '02/04/2023', 85.54);

INSERT INTO bookings VALUES (4, 'FOREST_L', '02/08/2023', 75.00);
INSERT INTO bookings VALUES (5, 'FOREST_L', '02/11/2023', 75.00);
INSERT INTO bookings VALUES (6, 'FOREST_L', '02/14/2023', 90.00);

INSERT INTO bookings VALUES (7, 'FOREST_L', '02/21/2023', 60.00);
INSERT INTO bookings VALUES (8, 'FOREST_L', '02/22/2023', 85.00);
INSERT INTO bookings VALUES (9, 'FOREST_L', '02/27/2023', 90.00);

INSERT INTO bookings VALUES (10, 'DESERT_S', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (11, 'DESERT_S', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (12, 'DESERT_S', '02/04/2023', 85.00);

INSERT INTO bookings VALUES (13, 'DESERT_S', '02/05/2023', 75.00);
INSERT INTO bookings VALUES (14, 'DESERT_S', '02/06/2023', 34.00);
INSERT INTO bookings VALUES (15, 'DESERT_S', '02/09/2023', 85.00);

INSERT INTO bookings VALUES (16, 'DESERT_S', '02/12/2023', 23.00);
INSERT INTO bookings VALUES (17, 'DESERT_S', '02/13/2023', 76.00);
INSERT INTO bookings VALUES (18, 'DESERT_S', '02/14/2023', 85.00);

INSERT INTO bookings VALUES (19, 'OCEAN_WV', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (20, 'OCEAN_WV', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (21, 'OCEAN_WV', '02/04/2023', 85.00);
```

```
INSERT INTO bookings VALUES (22, 'OCEAN_WV', '02/06/2023', 75.00);
INSERT INTO bookings VALUES (23, 'OCEAN_WV', '02/09/2023', 34.00);
INSERT INTO bookings VALUES (24, 'OCEAN_WV', '02/12/2023', 85.00);

INSERT INTO bookings VALUES (25, 'OCEAN_WV', '02/13/2023', 23.00);
INSERT INTO bookings VALUES (26, 'OCEAN_WV', '02/14/2023', 76.00);
INSERT INTO bookings VALUES (27, 'OCEAN_WV', '02/16/2023', 85.00);

INSERT INTO bookings VALUES (28, 'CITY_BLD', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (29, 'CITY_BLD', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (30, 'CITY_BLD', '02/04/2023', 85.00);

INSERT INTO bookings VALUES (31, 'CITY_BLD', '02/12/2023', 75.00);
INSERT INTO bookings VALUES (32, 'CITY_BLD', '02/13/2023', 34.00);
INSERT INTO bookings VALUES (33, 'CITY_BLD', '02/17/2023', 85.00);

INSERT INTO bookings VALUES (34, 'CITY_BLD', '02/22/2023', 23.00);
INSERT INTO bookings VALUES (35, 'CITY_BLD', '02/23/2023', 76.00);
INSERT INTO bookings VALUES (36, 'CITY_BLD', '02/24/2023', 85.00);
```

Dalam kueri sampel ini, catatan pemesanan dihitung untuk memberikan total untuk setiap minggu. Tanggal akhir untuk setiap minggu menjadi nama kolom.

```
SELECT * FROM
    (SELECT
       booking_id,
       (date_trunc('week', booking_date::date) + '5 days'::interval)::date as enddate,
       hotel_code AS "hotel code"
FROM bookings
) PIVOT (
    count(booking_id) FOR enddate IN ('2023-02-04','2023-02-11','2023-02-18')
);
```

Kueri menghasilkan output berikut.

```
 hotel code | 2023-02-04  | 2023-02-11 | 2023-02-18
------------+-------------+------------+----------
 FOREST_L   |           3 |          2 |        1
 DESERT_S   |           4 |          3 |        2
 OCEAN_WV   |           3 |          3 |        3
 CITY_BLD   |           3 |          1 |        2
```

Amazon Redshift tidak mendukung CROSSTAB untuk melakukan pivot pada beberapa kolom. Tetapi Anda dapat mengubah data baris ke kolom, dengan cara yang mirip dengan agregasi dengan PIVOT, dengan kueri seperti berikut ini. Ini menggunakan data sampel pemesanan yang sama seperti contoh sebelumnya.

```
SELECT
  booking_date,
  MAX(CASE WHEN hotel_code = 'FOREST_L' THEN 'forest is booked' ELSE '' END) AS
 FOREST_L,
  MAX(CASE WHEN hotel_code = 'DESERT_S' THEN 'desert is booked' ELSE '' END) AS
 DESERT_S,
  MAX(CASE WHEN hotel_code = 'OCEAN_WV' THEN 'ocean is booked' ELSE '' END)  AS
 OCEAN_WV
FROM bookings
GROUP BY booking_date
ORDER BY booking_date asc;
```

Contoh kueri menghasilkan tanggal pemesanan yang tercantum di sebelah frasa pendek yang menunjukkan hotel mana yang dipesan.

```
 booking_date  | forest_l         | desert_s         | ocean_wv
---------------+------------------+------------------+--------------------
 2023-02-01    | forest is booked | desert is booked |  ocean is booked
 2023-02-02    | forest is booked | desert is booked |  ocean is booked
 2023-02-04    | forest is booked | desert is booked |  ocean is booked
 2023-02-05    |                  | desert is booked |
 2023-02-06    |                  | desert is booked |
```

Berikut ini adalah catatan penggunaan untuk`PIVOT`:

- `PIVOT`dapat diterapkan pada tabel, sub-query, dan ekspresi tabel umum (CTE). `PIVOT`tidak dapat diterapkan pada `JOIN` ekspresi, CTE rekursif,`PIVOT`, atau `UNPIVOT` ekspresi apa pun. Juga tidak didukung adalah ekspresi `SUPER` tidak bersarang dan tabel bersarang Redshift Spectrum.
- `PIVOT`mendukung fungsi`COUNT`,`SUM`,`MIN`,`MAX`, dan `AVG` agregat.
- Ek `PIVOT` spresi agregat harus berupa panggilan dari fungsi agregat yang didukung. Ekspresi kompleks di atas agregat tidak didukung. Argumen agregat tidak dapat berisi referensi ke tabel selain tabel `PIVOT` input. Referensi berkorelasi ke kueri induk juga tidak didukung. Argumen agregat mungkin berisi sub-query. Ini dapat dikorelasikan secara internal atau pada tabel `PIVOT` input.

- Nilai `PIVOT IN` daftar tidak dapat berupa referensi kolom atau sub-kueri. Setiap nilai harus tipe yang kompatibel dengan referensi `FOR` kolom.
- Jika nilai `IN` daftar tidak memiliki alias, `PIVOT` menghasilkan nama kolom default. Untuk `IN` nilai konstan seperti 'abc' atau 5 nama kolom default adalah konstanta itu sendiri. Untuk ekspresi kompleks apa pun, nama kolom adalah nama default Amazon Redshift standar seperti`?column?`.

Contoh UNPIVOT

Siapkan data sampel dan gunakan untuk menjalankan contoh berikutnya.

```
CREATE TABLE count_by_color (quality varchar, red int, green int, blue int);

INSERT INTO count_by_color VALUES ('high', 15, 20, 7);
INSERT INTO count_by_color VALUES ('normal', 35, NULL, 40);
INSERT INTO count_by_color VALUES ('low', 10, 23, NULL);
```

`UNPIVOT`pada kolom input merah, hijau, dan biru.

```
SELECT *
FROM (SELECT red, green, blue FROM count_by_color) UNPIVOT (
    cnt FOR color IN (red, green, blue)
);
```

Kueri menghasilkan output berikut.

```
 color | cnt
-------+-----
 red   |  15
 red   |  35
 red   |  10
 green |  20
 green |  23
 blue  |   7
 blue  |  40
```

Secara default, `NULL` nilai di kolom input dilewati dan tidak menghasilkan baris hasil.

Contoh berikut menunjukkan `UNPIVOT` dengan`INCLUDE NULLS`.

```
SELECT *
```

```
FROM (
    SELECT red, green, blue
    FROM count_by_color
) UNPIVOT INCLUDE NULLS (
    cnt FOR color IN (red, green, blue)
);
```

Berikut ini adalah output yang dihasilkan.

```
 color | cnt
-------+-----
 red   |  15
 red   |  35
 red   |  10
 green |  20
 green |
 green |  23
 blue  |   7
 blue  |  40
 blue  |
```

Jika `INCLUDING NULLS` parameter disetel, nilai `NULL` input menghasilkan baris hasil.

`The following query shows UNPIVOT`dengan `quality` kolom implisit.

```
SELECT *
FROM count_by_color UNPIVOT (
    cnt FOR color IN (red, green, blue)
);
```

Kueri menghasilkan output berikut.

```
 quality | color | cnt
---------+-------+-----
 high    | red   |  15
 normal  | red   |  35
 low     | red   |  10
 high    | green |  20
 low     | green |  23
 high    | blue  |   7
 normal  | blue  |  40
```

Kolom tabel input yang tidak direferensikan dalam UNPIVOT definisi ditambahkan secara implisit ke tabel hasil. Dalam contoh, ini adalah kasus untuk quality kolom.

Contoh berikut menunjukkan UNPIVOT dengan alias untuk nilai dalam IN daftar.

```
SELECT *
FROM count_by_color UNPIVOT (
    cnt FOR color IN (red AS r, green AS g, blue AS b)
);
```

Kueri sebelumnya menghasilkan output berikut.

```
 quality | color | cnt
---------+-------+-----
 high    | r     |  15
 normal  | r     |  35
 low     | r     |  10
 high    | g     |  20
 low     | g     |  23
 high    | b     |   7
 normal  | b     |  40
```

UNPIVOTOperator menerima alias opsional pada setiap nilai IN daftar. Setiap alias menyediakan kustomisasi data di setiap value kolom.

Berikut ini adalah catatan penggunaan untukUNPIVOT.

- UNPIVOTdapat diterapkan pada tabel, sub-query, dan ekspresi tabel umum (CTE). UNPIVOTtidak dapat diterapkan pada JOIN ekspresi, CTE rekursif,PIVOT, atau UNPIVOT ekspresi apa pun. Juga tidak didukung adalah ekspresi SUPER tidak bersarang dan tabel bersarang Redshift Spectrum.
- UNPIVOT INDaftar harus berisi hanya referensi kolom tabel masukan. Kol IN om daftar harus memiliki tipe umum yang semuanya kompatibel dengannya. Kol UNPIVOT om nilai memiliki tipe umum ini. Kol UNPIVOT om nama adalah tipeVARCHAR.
- Jika nilai IN daftar tidak memiliki alias, UNPIVOT gunakan nama kolom sebagai nilai default.

Contoh JOIN

Klausa SQL JOIN digunakan untuk menggabungkan data dari dua atau lebih tabel berdasarkan bidang umum. Hasilnya mungkin atau mungkin tidak berubah tergantung pada metode gabungan yang ditentukan. Untuk informasi selengkapnya tentang sintaks klausa JOIN, lihatParameter.

Contoh berikut menggunakan data dari data TICKIT sampel. Untuk informasi selengkapnya tentang skema database, lihatDatabase sampel. Untuk mempelajari cara memuat data sampel, lihat Memuat data di Panduan Memulai Amazon Redshift.

Kueri berikut adalah penggabungan dalam (tanpa kata kunci JOIN) antara tabel LISTING dan tabel SALES, di mana LISTID dari tabel LISTING adalah antara 1 dan 5. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel SALES (tabel kanan). Hasilnya menunjukkan bahwa LISTID 1, 4, dan 5 cocok dengan kriteria.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing, sales
where listing.listid = sales.listid
and listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan luar kiri. Gabungan luar kiri dan kanan mempertahankan nilai dari salah satu tabel yang digabungkan ketika tidak ada kecocokan yang ditemukan di tabel lain. Tabel kiri dan kanan adalah tabel pertama dan kedua yang tercantum dalam sintaks. Nilai NULL digunakan untuk mengisi “celah” dalam kumpulan hasil. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel SALES (tabel kanan). Hasilnya menunjukkan bahwa ListID 2 dan 3 tidak menghasilkan penjualan apa pun.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing left outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     2 | NULL   | NULL
     3 | NULL   | NULL
     4 |  76.00 |  11.40
```

```
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan luar kanan. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel SALES (tabel kanan). Hasilnya menunjukkan bahwa ListID 1, 4, dan 5 cocok dengan kriteria.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing right outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan penuh. Gabungan penuh mempertahankan nilai dari tabel yang digabungkan ketika tidak ada kecocokan yang ditemukan di tabel lain. Tabel kiri dan kanan adalah tabel pertama dan kedua yang tercantum dalam sintaks. Nilai NULL digunakan untuk mengisi “celah” dalam kumpulan hasil. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel SALES (tabel kanan). Hasilnya menunjukkan bahwa ListID 2 dan 3 tidak menghasilkan penjualan apa pun.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     2 | NULL   | NULL
     3 | NULL   | NULL
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan penuh. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel SALES (tabel kanan). Hanya baris yang tidak menghasilkan penjualan apa pun (ListID 2 dan 3) yang ada dalam hasil.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
and (listing.listid IS NULL or sales.listid IS NULL)
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     2 | NULL   | NULL
     3 | NULL   | NULL
```

Contoh berikut adalah gabungan dalam dengan klausa ON. Dalam hal ini, baris NULL tidak dikembalikan.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah penggabungan silang atau gabungan Cartesian dari tabel LISTING dan tabel SALES dengan predikat untuk membatasi hasil. Kueri ini cocok dengan nilai kolom LISTID dalam tabel SALES dan tabel LISTING untuk ListID 1, 2, 3, 4, dan 5 di kedua tabel. Hasilnya menunjukkan bahwa 20 baris cocok dengan kriteria.

```
select sales.listid as sales_listid, listing.listid as listing_listid
from sales cross join listing
where sales.listid between 1 and 5
and listing.listid between 1 and 5
```

```
order by 1,2;

sales_listid | listing_listid
-------------+---------------
1            | 1
1            | 2
1            | 3
1            | 4
1            | 5
4            | 1
4            | 2
4            | 3
4            | 4
4            | 5
5            | 1
5            | 1
5            | 2
5            | 2
5            | 3
5            | 3
5            | 4
5            | 4
5            | 5
5            | 5
```

Contoh berikut adalah gabungan alami antara dua tabel. Dalam hal ini, kolom listid, sellerid, eventid, dan dateid memiliki nama dan tipe data yang identik di kedua tabel sehingga digunakan sebagai kolom gabungan. Hasilnya dibatasi hingga lima baris.

```
select listid, sellerid, eventid, dateid, numtickets
from listing natural join sales
order by 1
limit 5;

listid | sellerid  | eventid | dateid | numtickets
-------+-----------+---------+--------+-----------
113    | 29704     | 4699    | 2075   | 22
115    | 39115     | 3513    | 2062   | 14
116    | 43314     | 8675    | 1910   | 28
118    | 6079      | 1611    | 1862   | 9
163    | 24880     | 8253    | 1888   | 14
```

Contoh berikut adalah gabungan antara dua tabel dengan klausa USING. Dalam hal ini, kolom listid dan eventid digunakan sebagai kolom gabungan. Hasilnya dibatasi hingga lima baris.

```
select listid, listing.sellerid, eventid, listing.dateid, numtickets
from listing join sales
using (listid, eventid)
order by 1
limit 5;

listid | sellerid | eventid | dateid | numtickets
-------+----------+---------+--------+-----------
1      | 36861    | 7872    | 1850   | 10
4      | 8117     | 4337    | 1970   | 8
5      | 1616     | 8647    | 1963   | 4
5      | 1616     | 8647    | 1963   | 4
6      | 47402    | 8240    | 2053   | 18
```

Kueri berikut adalah gabungan dalam dari dua subquery dalam klausa FROM. Kueri menemukan jumlah tiket yang dijual dan tidak terjual untuk berbagai kategori acara (konser dan pertunjukan). Subquery klausa FROM adalah subquery tabel; mereka dapat mengembalikan beberapa kolom dan baris.

```
select catgroup1, sold, unsold
from
(select catgroup, sum(qtysold) as sold
from category c, event e, sales s
where c.catid = e.catid and e.eventid = s.eventid
group by catgroup) as a(catgroup1, sold)
join
(select catgroup, sum(numtickets)-sum(qtysold) as unsold
from category c, event e, sales s, listing l
where c.catid = e.catid and e.eventid = s.eventid
and s.listid = l.listid
group by catgroup) as b(catgroup2, unsold)

on a.catgroup1 = b.catgroup2
order by 1;

catgroup1 |  sold  | unsold
----------+--------+--------
Concerts  | 195444 |1067199
Shows     | 149905 | 817736
```

Contoh UNNEST

UNNEST adalah parameter dalam klausa FROM yang memperluas data bersarang ke kolom yang menyimpan elemen data yang tidak bersarang. Untuk informasi tentang data yang tidak bersarang, lihatMengkueri data semi-terstruktur.

Pernyataan berikut membuat dan mengisi `orders` tabel, yang berisi `products` kolom berisi array ID produk. Contoh-contoh di bagian ini menggunakan data sampel dalam tabel ini.

```
CREATE TABLE orders (
    order_id INT,
    products SUPER
);

-- Populate table
INSERT INTO orders VALUES
(1001, JSON_PARSE('[
        {
            "product_id": "P456",
            "name": "Monitor",
            "price": 299.99,
            "quantity": 1,
            "specs": {
                "size": "27 inch",
                "resolution": "4K"
            }
        }
    ]
')),
(1002, JSON_PARSE('
    [
        {
            "product_id": "P567",
            "name": "USB Cable",
            "price": 9.99,
            "quantity": 3
        },
        {
            "product_id": "P678",
            "name": "Headphones",
            "price": 159.99,
            "quantity": 1,
            "specs": {
```

```
                "type": "Wireless",
                "battery_life": "20 hours"
            }
        }
    ]
'));
```

Berikut ini adalah beberapa contoh kueri yang tidak bersarang dengan data sampel menggunakan sintaks PartiQL.

Membuang sarang array tanpa kolom OFFSET

Kueri berikut menguraikan array SUPER di kolom produk, dengan setiap baris mewakili item dari pesanan masuk. `order_id`

```
SELECT o.order_id, unnested_products.product
FROM orders o, UNNEST(o.products) AS unnested_products(product);

 order_id |                                                                  product

----------
+-----------------------------------------------------------------------------------------------------------
     1001 | {"product_id":"P456","name":"Monitor","price":299.99,"quantity":1,"specs":
{"size":"27 inch","resolution":"4K"}}
     1002 | {"product_id":"P567","name":"USB Cable","price":9.99,"quantity":3}
     1002 |
 {"product_id":"P678","name":"Headphones","price":159.99,"quantity":1,"specs":
{"type":"Wireless","battery_life":"20 hours"}}
(3 rows)
```

Kueri berikut menemukan produk paling mahal di setiap pesanan.

```
SELECT o.order_id, MAX(unnested_products.product)
FROM orders o, UNNEST(o.products) AS unnested_products(product);

 order_id |                                                                  product

----------
+-----------------------------------------------------------------------------------------------------------
     1001 | {"product_id":"P456","name":"Monitor","price":299.99,"quantity":1,"specs":
{"size":"27 inch","resolution":"4K"}}
```

```
     1002 |
 {"product_id":"P678","name":"Headphones","price":159.99,"quantity":1,"specs":
{"type":"Wireless","battery_life":"20 hours"}}
(2 rows)
```

Membuang sarang array dengan kolom OFFSET implisit

Kueri berikut menggunakan `UNNEST ... WITH OFFSET` parameter untuk menunjukkan posisi berbasis nol dari setiap produk dalam array pesanannya.

```
SELECT o.order_id, up.product, up.offset_col
FROM orders o, UNNEST(o.products) WITH OFFSET AS up(product);

 order_id |                                                                    product
                                                 | offset_col
----------
+-----------------------------------------------------------------------------------------------------------------
+------------
     1001 | {"product_id":"P456","name":"Monitor","price":299.99,"quantity":1,"specs":
{"size":"27 inch","resolution":"4K"}}              |          0
     1002 | {"product_id":"P567","name":"USB Cable","price":9.99,"quantity":3}
                                                 |          0
     1002 |
 {"product_id":"P678","name":"Headphones","price":159.99,"quantity":1,"specs":
{"type":"Wireless","battery_life":"20 hours"}} |          1
(3 rows)
```

Karena pernyataan tidak menentukan alias untuk kolom offset, Amazon Redshift secara default menamainya. `offset_col`

Membuang sarang array dengan kolom OFFSET eksplisit

Kueri berikut juga menggunakan `UNNEST ... WITH OFFSET` parameter untuk menampilkan produk dalam array pesanan mereka. Perbedaan dalam kueri ini dibandingkan dengan kueri pada contoh sebelumnya adalah bahwa ia secara eksplisit menamai kolom offset dengan alias`idx`.

```
SELECT o.order_id, up.product, up.idx
FROM orders o, UNNEST(o.products) WITH OFFSET AS up(product, idx);

 order_id |                                                                    product
                                                 | idx
```

```
----------
+-------------------------------------------------------------------------------------------------
+-----
     1001 | {"product_id":"P456","name":"Monitor","price":299.99,"quantity":1,"specs":
{"size":"27 inch","resolution":"4K"}}             |   0
     1002 | {"product_id":"P567","name":"USB Cable","price":9.99,"quantity":3}
                                                  |   0
     1002 |
 {"product_id":"P678","name":"Headphones","price":159.99,"quantity":1,"specs":
{"type":"Wireless","battery_life":"20 hours"}} |   1
(3 rows)
```

Klausa WHERE

Klausa WHERE berisi kondisi yang menggabungkan tabel atau menerapkan predikat ke kolom dalam tabel. Tabel dapat digabungkan ke dalam dengan menggunakan sintaks yang sesuai baik dalam klausa WHERE atau klausa FROM. Kriteria gabungan luar harus ditentukan dalam klausa FROM.

Sintaksis

```
[ WHERE condition ]
```

ketentuan

Setiap kondisi pencarian dengan hasil Boolean, seperti kondisi gabungan atau predikat pada kolom tabel. Contoh berikut adalah kondisi bergabung yang valid:

```
sales.listid=listing.listid
sales.listid<>listing.listid
```

Contoh berikut adalah kondisi yang valid pada kolom dalam tabel:

```
catgroup like 'S%'
venueseats between 20000 and 50000
eventname in('Jersey Boys','Spamalot')
year=2008
length(catdesc)>25
date_part(month, caldate)=6
```

Kondisi bisa sederhana atau kompleks; untuk kondisi kompleks, Anda dapat menggunakan tanda kurung untuk mengisolasi unit logis. Dalam contoh berikut, kondisi gabungan ditutup dengan tanda kurung.

```
where (category.catid=event.catid) and category.catid in(6,7,8)
```

Catatan penggunaan

Anda dapat menggunakan alias dalam klausa WHERE untuk mereferensikan ekspresi daftar pilih.

Anda tidak dapat membatasi hasil fungsi agregat dalam klausa WHERE; gunakan klausa HAVING untuk tujuan ini.

Kolom yang dibatasi dalam klausa WHERE harus berasal dari referensi tabel dalam klausa FROM.

Contoh

Kueri berikut menggunakan kombinasi pembatasan klausa WHERE yang berbeda, termasuk kondisi gabungan untuk tabel SALES dan EVENT, predikat pada kolom EVENTNAME, dan dua predikat pada kolom STARTTIME.

```
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
where sales.eventid = event.eventid
and eventname='Hannah Montana'
and date_part(quarter, starttime) in(1,2)
and date_part(year, starttime) = 2008
order by 3 desc, 4, 2, 1 limit 10;

eventname    |      starttime      |   costperticket   | qtysold
---------------+---------------------+-------------------+---------
Hannah Montana | 2008-06-07 14:00:00 |     1706.00000000 |       2
Hannah Montana | 2008-05-01 19:00:00 |     1658.00000000 |       2
Hannah Montana | 2008-06-07 14:00:00 |     1479.00000000 |       1
Hannah Montana | 2008-06-07 14:00:00 |     1479.00000000 |       3
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       1
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       2
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       4
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       1
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       2
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       4
(10 rows)
```

Oracle-Style gabungan luar dalam klausa WHERE

Untuk kompatibilitas Oracle, Amazon Redshift mendukung operator outer-join Oracle (+) dalam kondisi gabungan klausa WHERE. Operator ini dimaksudkan untuk digunakan hanya dalam mendefinisikan kondisi outer-join; jangan mencoba menggunakannya dalam konteks lain. Penggunaan lain dari operator ini diabaikan secara diam-diam dalam banyak kasus.

Gabungan luar mengembalikan semua baris yang akan dikembalikan oleh gabungan dalam yang setara, ditambah baris yang tidak cocok dari satu atau kedua tabel. Dalam klausa FROM, Anda dapat menentukan gabungan luar kiri, kanan, dan penuh. Dalam klausa WHERE, Anda dapat menentukan gabungan luar kiri dan kanan saja.

Untuk menggabungkan tabel luar TABLE1 dan TABLE2 dan mengembalikan baris yang tidak cocok dari TABLE1 (gabungan luar kiri), tentukan `TABLE1 LEFT OUTER JOIN TABLE2` dalam klausa FROM atau terapkan operator (+) ke semua kolom penggabungan dari TABLE2 dalam klausa WHERE. Untuk semua baris di TABLE1 yang tidak memiliki baris yang cocok di TABLE2, hasil kueri berisi nol untuk setiap ekspresi daftar pilih yang berisi kolom dari TABLE2.

Untuk menghasilkan perilaku yang sama untuk semua baris di TABLE2 yang tidak memiliki baris yang cocok di TABLE1, tentukan `TABLE1 RIGHT OUTER JOIN TABLE2` dalam klausa FROM atau terapkan operator (+) ke semua kolom penggabungan dari TABLE1 dalam klausa WHERE.

Sintaks dasar

```
[ WHERE {
[ table1.column1 = table2.column1(+) ]
[ table1.column1(+) = table2.column1 ]
}
```

Kondisi pertama setara dengan:

```
from table1 left outer join table2
on table1.column1=table2.column1
```

Kondisi kedua setara dengan:

```
from table1 right outer join table2
on table1.column1=table2.column1
```

> **Note**
>
> Sintaks yang ditampilkan di sini mencakup kasus sederhana dari sebuah equijoin di atas satu pasang kolom penggabungan. Namun, jenis kondisi perbandingan lainnya dan beberapa pasang kolom penggabungan juga valid.

Misalnya, klausa WHERE berikut mendefinisikan gabungan luar di atas dua pasang kolom. Operator (+) harus dilampirkan ke tabel yang sama dalam kedua kondisi:

```
where table1.col1 > table2.col1(+)
and table1.col2 = table2.col2(+)
```

Catatan penggunaan

Jika memungkinkan, gunakan sintaks FROM klausa OUTER JOIN standar alih-alih operator (+) dalam klausa WHERE. Kueri yang berisi operator (+) tunduk pada aturan berikut:

- Anda hanya dapat menggunakan operator (+) dalam klausa WHERE, dan hanya mengacu pada kolom dari tabel atau tampilan.
- Anda tidak dapat menerapkan operator (+) ke ekspresi. Namun, ekspresi dapat berisi kolom yang menggunakan operator (+). Misalnya, kondisi gabungan berikut mengembalikan kesalahan sintaks:

  ```
  event.eventid*10(+)=category.catid
  ```

  Namun, kondisi bergabung berikut ini valid:

  ```
  event.eventid(+)*10=category.catid
  ```
- Anda tidak dapat menggunakan operator (+) dalam blok kueri yang juga berisi sintaks gabungan klausa FROM.
- Jika dua tabel digabungkan melalui beberapa kondisi gabungan, Anda harus menggunakan operator (+) di semua atau tidak satu pun dari kondisi ini. Gabungan dengan gaya sintaks campuran berjalan sebagai penggabungan dalam, tanpa peringatan.
- Operator (+) tidak menghasilkan gabungan luar jika Anda menggabungkan tabel di kueri luar dengan tabel yang dihasilkan dari kueri dalam.
- Untuk menggunakan operator (+) untuk menggabungkan tabel ke dirinya sendiri, Anda harus mendefinisikan alias tabel dalam klausa FROM dan mereferensikannya dalam kondisi gabungan:

```
select count(*)
from event a, event b
where a.eventid(+)=b.catid;

count
-------
8798
(1 row)
```

- Anda tidak dapat menggabungkan kondisi gabungan yang berisi operator (+) dengan kondisi OR atau kondisi IN. Contoh:

```
select count(*) from sales, listing
where sales.listid(+)=listing.listid or sales.salesid=0;
ERROR:  Outer join operator (+) not allowed in operand of OR or IN.
```

- Dalam klausa WHERE yang menghubungkan lebih dari dua tabel, operator (+) hanya dapat diterapkan sekali ke tabel tertentu. Dalam contoh berikut, tabel SALES tidak dapat direferensikan dengan operator (+) dalam dua gabungan berturut-turut.

```
select count(*) from sales, listing, event
where sales.listid(+)=listing.listid and sales.dateid(+)=date.dateid;
ERROR:  A table may be outer joined to at most one other table.
```

- Jika kondisi outter-join klausa WHERE membandingkan kolom dari TABLE2 dengan konstanta, terapkan operator (+) ke kolom. Jika Anda tidak menyertakan operator, baris yang bergabung di luar dari TABLE1, yang berisi nol untuk kolom terbatas, dihilangkan. Lihat bagian Contoh di bawah ini.

Contoh

Kueri gabungan berikut menentukan gabungan luar kiri tabel SALES dan LISTING di atas kolom LISTID mereka:

```
select count(*)
from sales, listing
where sales.listid = listing.listid(+);

count
--------
```

```
172456
(1 row)
```

Kueri ekuivalen berikut menghasilkan hasil yang sama tetapi menggunakan sintaks gabungan klausa FROM:

```
select count(*)
from sales left outer join listing on sales.listid = listing.listid;

count
--------
172456
(1 row)
```

Tabel PENJUALAN tidak berisi catatan untuk semua listing di tabel LISTING karena tidak semua listing menghasilkan penjualan. Kueri berikut outter-bergabung dengan SALES dan LISTING dan mengembalikan baris dari LISTING bahkan ketika tabel SALES melaporkan tidak ada penjualan untuk ID daftar tertentu. Kolom PRICE dan COMM, yang berasal dari tabel SALES, berisi nol dalam set hasil untuk baris yang tidak cocok tersebut.

```
select listing.listid, sum(pricepaid) as price,
sum(commission) as comm
from listing, sales
where sales.listid(+) = listing.listid and listing.listid between 1 and 5
group by 1 order by 1;

listid | price  |  comm
--------+--------+--------
1 | 728.00 | 109.20
2 |        |
3 |        |
4 |  76.00 |  11.40
5 | 525.00 |  78.75
(5 rows)
```

Perhatikan bahwa ketika operator gabungan klausa WHERE digunakan, urutan tabel dalam klausa FROM tidak masalah.

Contoh kondisi gabungan luar yang lebih kompleks dalam klausa WHERE adalah kasus di mana kondisi terdiri dari perbandingan antara dua kolom tabel dan perbandingan dengan konstanta:

```
where category.catid=event.catid(+) and eventid(+)=796;
```

Perhatikan bahwa operator (+) digunakan di dua tempat: pertama dalam perbandingan kesetaraan antara tabel dan kedua dalam kondisi perbandingan untuk kolom EVENTID. Hasil dari sintaks ini adalah pelestarian baris yang bergabung di luar ketika pembatasan pada EVENTID dievaluasi. Jika Anda menghapus operator (+) dari pembatasan EVENTID, kueri memperlakukan pembatasan ini sebagai filter, bukan sebagai bagian dari kondisi outer-join. Pada gilirannya, baris yang bergabung di luar yang berisi nol untuk EVENTID dihilangkan dari kumpulan hasil.

Berikut adalah kueri lengkap yang menggambarkan perilaku ini:

```
select catname, catgroup, eventid
from category, event
where category.catid=event.catid(+) and eventid(+)=796;

catname | catgroup | eventid
-----------+----------+---------
Classical | Concerts |
Jazz | Concerts |
MLB | Sports   |
MLS | Sports   |
Musicals | Shows    | 796
NBA | Sports   |
NFL | Sports   |
NHL | Sports   |
Opera | Shows    |
Plays | Shows    |
Pop | Concerts |
(11 rows)
```

Kueri setara menggunakan sintaks klausa FROM adalah sebagai berikut:

```
select catname, catgroup, eventid
from category left join event
on category.catid=event.catid and eventid=796;
```

Jika Anda menghapus operator kedua (+) dari versi klausa WHERE dari kueri ini, ia mengembalikan hanya 1 baris (baris di mana`eventid`=796).

```
select catname, catgroup, eventid
```

```
from category, event
where category.catid=event.catid(+) and eventid=796;

catname | catgroup | eventid
-----------+----------+---------
Musicals | Shows    | 796
(1 row)
```

Klausa GROUP BY

Klausa GROUP BY mengidentifikasi kolom pengelompokan untuk kueri. Ini digunakan untuk mengelompokkan baris dalam tabel yang memiliki nilai yang sama di semua kolom yang tercantum. Urutan kolom terdaftar tidak masalah. Hasilnya adalah menggabungkan setiap set baris yang memiliki nilai umum ke dalam satu baris grup yang mewakili semua baris dalam grup. Gunakan GROUP BY untuk menghilangkan redundansi dalam output dan untuk menghitung agregat yang berlaku untuk grup. Kolom pengelompokan harus dideklarasikan ketika kueri menghitung agregat dengan fungsi standar seperti SUM, AVG, dan COUNT. Untuk informasi selengkapnya, lihat Fungsi agregat.

Sintaks

```
[ GROUP BY  expression [, ...] | ALL | aggregation_extension  ]
```

dimana aggregation_extension adalah salah satu dari berikut ini:

```
GROUPING SETS ( () | aggregation_extension [, ...] ) |
ROLLUP ( expr [, ...] ) |
CUBE ( expr [, ...] )
```

Parameter

ekspresi

Daftar kolom atau ekspresi harus cocok dengan daftar ekspresi non-agregat dalam daftar kueri yang dipilih. Misalnya, pertimbangkan kueri sederhana berikut.

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by listid, eventid
```

```
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

Dalam kueri ini, daftar pilih terdiri dari dua ekspresi agregat. Yang pertama menggunakan fungsi SUM dan yang kedua menggunakan fungsi COUNT. Dua kolom yang tersisa, LISTID dan EVENTID, harus dinyatakan sebagai kolom pengelompokan.

Ekspresi dalam klausa GROUP BY juga dapat mereferensikan daftar pilih dengan menggunakan nomor ordinal. Misalnya, contoh sebelumnya dapat disingkat sebagai berikut.

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by 1,2
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

SEMUA

SEMUA menunjukkan untuk mengelompokkan oleh semua kolom yang ditentukan dalam daftar SELECT kecuali yang digabungkan. Misalnya, pertimbangkan kueri berikut yang mengelompokkan dengan `col1` dan `col2` tanpa harus menentukannya secara individual dalam klausa GROUP BY. Kolom `col3` adalah argumen `SUM` fungsi dan dengan demikian tidak dikelompokkan.

```
SELECT col1, col2 sum(col3) FROM testtable GROUP BY ALL
```

Jika Anda MENGECUALIKAN kolom dalam daftar SELECT, klausa GROUP BY ALL tidak mengelompokkan hasil berdasarkan kolom tertentu.

```
SELECT * EXCLUDE col3 FROM testtable GROUP BY ALL
```

ekstensi_agregasi

Anda dapat menggunakan ekstensi agregasi GROUPING SETS, ROLLUP, dan CUBE untuk melakukan pekerjaan beberapa operasi GROUP BY dalam satu pernyataan. Untuk informasi selengkapnya tentang ekstensi agregasi dan fungsi terkait, lihatEkstensi agregasi.

Contoh

Contoh berikut menggunakan tabel SALES yang berisi kolom: salesid, listid, sellerid, buyerid, eventid, dateid, qtysold, pricepaid, commission, dan saletime. Untuk informasi selengkapnya tentang tabel PENJUALAN, lihatDatabase sampel.

Contoh kueri berikut mengelompok `salesid` kan oleh dan `listid` tanpa harus menentukannya secara individual dalam klausa GROUP BY. Kolom `qtysold` adalah argumen `SUM` fungsi dan dengan demikian tidak dikelompokkan.

```
SELECT salesid, listid, sum(qtysold) FROM sales GROUP BY ALL;

salesid | listid  | sum
--------+---------+------
33095   | 36572   | 2
88268   | 100813  | 4
110917  | 127048  | 1
...
```

Contoh kueri berikut mengecualikan beberapa kolom dalam daftar SELECT, jadi GROUP BY ALL hanya mengelompokkan salesid dan listid.

```
SELECT * EXCLUDE sellerid, buyerid, eventid, dateid, qtysold, pricepaid, commission,
 saletime
FROM sales GROUP BY ALL;

salesid | listid
```

```
--------+---------
33095   | 36572
88268   | 100813
110917  | 127048
...
```

Ekstensi agregasi

Amazon Redshift mendukung ekstensi agregasi untuk melakukan pekerjaan beberapa operasi GROUP BY dalam satu pernyataan.

Contoh untuk ekstensi agregasi menggunakan `orders` tabel, yang menyimpan data penjualan untuk perusahaan elektronik. Anda dapat membuat `orders` dengan yang berikut ini.

```
CREATE TABLE ORDERS (
    ID INT,
    PRODUCT CHAR(20),
    CATEGORY CHAR(20),
    PRE_OWNED CHAR(1),
    COST DECIMAL
);

INSERT INTO ORDERS VALUES
    (0, 'laptop',       'computers',    'T', 1000),
    (1, 'smartphone',   'cellphones',   'T', 800),
    (2, 'smartphone',   'cellphones',   'T', 810),
    (3, 'laptop',       'computers',    'F', 1050),
    (4, 'mouse',        'computers',    'F', 50);
```

PENGELOMPOKAN SET

Menghitung satu atau lebih set pengelompokan dalam satu pernyataan. Kumpulan pengelompokan adalah kumpulan klausa GROUP BY tunggal, kumpulan 0 atau lebih kolom yang dengannya Anda dapat mengelompokkan kumpulan hasil kueri. GROUP BY GROUPING SET setara dengan menjalankan kueri UNION ALL pada satu set hasil yang dikelompokkan oleh kolom yang berbeda. Misalnya, GROUP BY GROUPING SET ((a), (b)) setara dengan GROUP BY a UNION ALL GROUP BY b.

Contoh berikut mengembalikan biaya produk tabel pesanan yang dikelompokkan menurut kategori produk dan jenis produk yang dijual.

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY GROUPING SETS(category, product);

       category       |       product        | total
----------------------+----------------------+-------
 computers            |                      |  2100
 cellphones           |                      |  1610
                      | laptop               |  2050
                      | smartphone           |  1610
                      | mouse                |    50

(5 rows)
```

MENGGULUNG

Mengasumsikan hierarki di mana kolom sebelumnya dianggap sebagai induk dari kolom berikutnya. ROLLUP mengelompokkan data berdasarkan kolom yang disediakan, mengembalikan baris subtotal tambahan yang mewakili total di semua tingkat kolom pengelompokan, selain baris yang dikelompokkan. Misalnya, Anda dapat menggunakan GROUP BY ROLLUP ((a), (b)) untuk mengembalikan set hasil yang dikelompokkan pertama oleh a, kemudian dengan b sambil mengasumsikan bahwa b adalah subbagian dari a. ROLLUP juga mengembalikan baris dengan seluruh hasil set tanpa mengelompokkan kolom.

GROUP BY ROLLUP ((a), (b)) setara dengan GROUP BY GROUPING SET ((a, b), (a), ()).

Contoh berikut mengembalikan biaya produk tabel pesanan yang dikelompokkan pertama berdasarkan kategori dan kemudian produk, dengan produk sebagai subdivisi kategori.

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY ROLLUP(category, product) ORDER BY 1,2;

       category       |       product        | total
----------------------+----------------------+-------
 cellphones           | smartphone           |  1610
 cellphones           |                      |  1610
 computers            | laptop               |  2050
 computers            | mouse                |    50
 computers            |                      |  2100
                      |                      |  3710
(6 rows)
```

KUBUS

Mengelompokkan data berdasarkan kolom yang disediakan, mengembalikan baris subtotal tambahan yang mewakili total di semua tingkat kolom pengelompokan, selain baris yang dikelompokkan. CUBE mengembalikan baris yang sama dengan ROLLUP, sambil menambahkan baris subtotal tambahan untuk setiap kombinasi kolom pengelompokan yang tidak tercakup oleh ROLLUP. Misalnya, Anda dapat menggunakan GROUP BY CUBE ((a), (b)) untuk mengembalikan set hasil yang dikelompokkan pertama oleh a, kemudian dengan b sambil mengasumsikan bahwa b adalah subbagian dari a, lalu oleh b saja. CUBE juga mengembalikan baris dengan seluruh hasil set tanpa mengelompokkan kolom.

GROUP BY CUBE ((a), (b)) setara dengan GRUP DENGAN MENGELOMPOKKAN SET ((a, b), (a), (b), ()).

Contoh berikut mengembalikan biaya produk tabel pesanan yang dikelompokkan pertama berdasarkan kategori dan kemudian produk, dengan produk sebagai subdivisi kategori. Tidak seperti contoh sebelumnya untuk ROLLUP, pernyataan mengembalikan hasil untuk setiap kombinasi kolom pengelompokan.

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 1,2;

       category       |       product        | total
----------------------+----------------------+-------
 cellphones           | smartphone           |  1610
 cellphones           |                      |  1610
 computers            | laptop               |  2050
 computers            | mouse                |    50
 computers            |                      |  2100
                      | laptop               |  2050
                      | mouse                |    50
                      | smartphone           |  1610
                      |                      |  3710
(9 rows)
```

GROUPING/GROUPINGFungsi _ID

ROLLUP dan CUBE menambahkan nilai NULL ke set hasil untuk menunjukkan baris subtotal. Misalnya, GROUP BY ROLLUP ((a), (b)) mengembalikan satu atau lebih baris yang memiliki nilai

NULL di kolom pengelompokan b untuk menunjukkan bahwa mereka adalah subtotal bidang di kolom pengelompokan a. Nilai NULL ini hanya berfungsi untuk memenuhi format tupel yang dikembalikan.

Ketika Anda menjalankan operasi GROUP BY dengan ROLLUP dan CUBE pada relasi yang menyimpan nilai NULL itu sendiri, ini dapat menghasilkan kumpulan hasil dengan baris yang tampaknya memiliki kolom pengelompokan yang identik. Kembali ke contoh sebelumnya, jika kolom pengelompokan b berisi nilai NULL yang disimpan, GROUP BY ROLLUP ((a), (b)) mengembalikan baris dengan nilai NULL di kolom pengelompokan b yang bukan subtotal.

Untuk membedakan antara nilai NULL yang dibuat oleh ROLLUP dan CUBE, dan nilai NULL yang disimpan dalam tabel itu sendiri, Anda dapat menggunakan fungsi GROUPING, atau alias GROUPING_ID. GROUPING mengambil satu set pengelompokan sebagai argumennya, dan untuk setiap baris dalam kumpulan hasil mengembalikan nilai bit 0 atau 1 yang sesuai dengan kolom pengelompokan di posisi itu, dan kemudian mengubah nilai itu menjadi bilangan bulat. Jika nilai di posisi itu adalah nilai NULL yang dibuat oleh ekstensi agregasi, GROUPING mengembalikan 1. Ini mengembalikan 0 untuk semua nilai lainnya, termasuk nilai NULL yang disimpan.

Misalnya, GRUPING (kategori, produk) dapat mengembalikan nilai-nilai berikut untuk baris tertentu, tergantung pada nilai kolom pengelompokan untuk baris tersebut. Untuk tujuan contoh ini, semua nilai NULL dalam tabel adalah nilai NULL yang dibuat oleh ekstensi agregasi.

kategori kolom	kolom produk	Nilai bit fungsi PENGELOMPOKAN	Nilai desimal
bukan NULL	bukan NULL	00	0
bukan NULL	NULL	01	1
NULL	bukan NULL	10	2
NULL	NULL	11	3

Fungsi GROUPING muncul di bagian daftar SELECT dari kueri dalam format berikut.

```
SELECT ... [GROUPING( expr )...] ...
  GROUP BY ... {CUBE | ROLLUP| GROUPING SETS} ( expr ) ...
```

Contoh berikut adalah sama dengan contoh sebelumnya untuk CUBE, tetapi dengan penambahan fungsi GRUPING untuk kumpulan pengelompokan.

```
SELECT category, product,
       GROUPING(category) as grouping0,
       GROUPING(product) as grouping1,
       GROUPING(category, product) as grouping2,
       sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 3,1,2;

       category       |       product        | grouping0 | grouping1 | grouping2 |
 total
----------------------+----------------------+-----------+-----------+-----------
+-------
 cellphones           | smartphone           |         0 |         0 |         0 |
 1610
 cellphones           |                      |         0 |         1 |         1 |
 1610
 computers            | laptop               |         0 |         0 |         0 |
 2050
 computers            | mouse                |         0 |         0 |         0 |
 50
 computers            |                      |         0 |         1 |         1 |
 2100
                      | laptop               |         1 |         0 |         2 |
 2050
                      | mouse                |         1 |         0 |         2 |
 50
                      | smartphone           |         1 |         0 |         2 |
 1610
                      |                      |         1 |         1 |         3 |
 3710
(9 rows)
```

ROLLUP dan CUBE sebagian

Anda dapat menjalankan operasi ROLLUP dan CUBE hanya dengan sebagian dari subtotal.

Sintaks untuk operasi ROLLUP dan CUBE sebagian adalah sebagai berikut.

```
GROUP BY expr1, { ROLLUP | CUBE }(expr2, [, ...])
```

Di sini, klausa GROUP BY hanya membuat baris subtotal pada tingkat expr2 dan seterusnya.

Contoh berikut menunjukkan operasi ROLLUP dan CUBE sebagian pada tabel pesanan, mengelompokkan terlebih dahulu berdasarkan apakah suatu produk sudah dimiliki sebelumnya dan kemudian menjalankan ROLLUP dan CUBE pada kolom kategori dan produk.

```
SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY pre_owned, ROLLUP(category, product) ORDER BY 4,1,2,3;

 pre_owned |       category       |       product        | group_id | total
-----------+----------------------+----------------------+----------+-------
 F         | computers            | laptop               |        0 |  1050
 F         | computers            | mouse                |        0 |    50
 T         | cellphones           | smartphone           |        0 |  1610
 T         | computers            | laptop               |        0 |  1000
 F         | computers            |                      |        2 |  1100
 T         | cellphones           |                      |        2 |  1610
 T         | computers            |                      |        2 |  1000
 F         |                      |                      |        6 |  1100
 T         |                      |                      |        6 |  2610
(9 rows)

SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY pre_owned, CUBE(category, product) ORDER BY 4,1,2,3;

 pre_owned |       category       |       product        | group_id | total
-----------+----------------------+----------------------+----------+-------
 F         | computers            | laptop               |        0 |  1050
 F         | computers            | mouse                |        0 |    50
 T         | cellphones           | smartphone           |        0 |  1610
 T         | computers            | laptop               |        0 |  1000
 F         | computers            |                      |        2 |  1100
 T         | cellphones           |                      |        2 |  1610
 T         | computers            |                      |        2 |  1000
 F         |                      | laptop               |        4 |  1050
 F         |                      | mouse                |        4 |    50
 T         |                      | laptop               |        4 |  1000
```

```
 T          |                      | smartphone           |        4 |  1610
 F          |                      |                      |        6 |  1100
 T          |                      |                      |        6 |  2610
(13 rows)
```

Karena kolom bekas tidak disertakan dalam operasi ROLLUP dan CUBE, tidak ada baris total keseluruhan yang mencakup semua baris lainnya.

Pengelompokan gabungan

Anda dapat menggabungkan beberapa SETS/ROLLUP/CUBE klausa GRUPING untuk menghitung tingkat subtotal yang berbeda. Pengelompokan gabungan mengembalikan produk Cartesian dari kumpulan pengelompokan yang disediakan.

Sintaks untuk menggabungkan SETS/ROLLUP/CUBE klausa GROUPING adalah sebagai berikut.

```
GROUP BY {ROLLUP|CUBE|GROUPING SETS}(expr1[, ...]),
         {ROLLUP|CUBE|GROUPING SETS}(expr1[, ...])[, ...]
```

Pertimbangkan contoh berikut untuk melihat bagaimana pengelompokan gabungan kecil dapat menghasilkan kumpulan hasil akhir yang besar.

```
SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY CUBE(category, product), GROUPING SETS(pre_owned, ())
ORDER BY 4,1,2,3;

 pre_owned |       category       |       product        | group_id | total
-----------+----------------------+----------------------+----------+-------
 F         | computers            | laptop               |        0 |  1050
 F         | computers            | mouse                |        0 |    50
 T         | cellphones           | smartphone           |        0 |  1610
 T         | computers            | laptop               |        0 |  1000
           | cellphones           | smartphone           |        1 |  1610
           | computers            | laptop               |        1 |  2050
           | computers            | mouse                |        1 |    50
 F         | computers            |                      |        2 |  1100
 T         | cellphones           |                      |        2 |  1610
 T         | computers            |                      |        2 |  1000
           | cellphones           |                      |        3 |  1610
           | computers            |                      |        3 |  2100
```

```
 F         |                      | laptop               |        4 |  1050
 F         |                      | mouse                |        4 |    50
 T         |                      | laptop               |        4 |  1000
 T         |                      | smartphone           |        4 |  1610
           |                      | laptop               |        5 |  2050
           |                      | mouse                |        5 |    50
           |                      | smartphone           |        5 |  1610
 F         |                      |                      |        6 |  1100
 T         |                      |                      |        6 |  2610
           |                      |                      |        7 |  3710
(22 rows)
```

Pengelompokan bersarang

Anda dapat menggunakan SETS/ROLLUP/CUBE operasi GRUPING sebagai GROUPING SETS expr untuk membentuk pengelompokan bersarang. Sub pengelompokan di dalam GROUPING SET bersarang diratakan.

Sintaks untuk pengelompokan bersarang adalah sebagai berikut.

```
GROUP BY GROUPING SETS({ROLLUP|CUBE|GROUPING SETS}(expr[, ...])[, ...])
```

Pertimbangkan contoh berikut.

```
SELECT category, product, pre_owned,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY GROUPING SETS(ROLLUP(category), CUBE(product, pre_owned))
ORDER BY 4,1,2,3;

       category       |       product        | pre_owned | group_id | total
----------------------+----------------------+-----------+----------+-------
 cellphones           |                      |           |        3 |  1610
 computers            |                      |           |        3 |  2100
                      | laptop               | F         |        4 |  1050
                      | laptop               | T         |        4 |  1000
                      | mouse                | F         |        4 |    50
                      | smartphone           | T         |        4 |  1610
                      | laptop               |           |        5 |  2050
                      | mouse                |           |        5 |    50
                      | smartphone           |           |        5 |  1610
                      |                      | F         |        6 |  1100
```

```
                    |                    | T          |        6 |  2610
                    |                    |            |        7 |  3710
                    |                    |            |        7 |  3710
(13 rows)
```

Perhatikan bahwa karena ROLLUP (kategori) dan CUBE (product, pre_own) berisi set pengelompokan (), baris yang mewakili total keseluruhan diduplikasi.

Catatan penggunaan

- Klausa GROUP BY mendukung hingga 64 set pengelompokan. Dalam kasus ROLLUP dan CUBE, atau beberapa kombinasi GROUPING SET, ROLLUP, dan CUBE, batasan ini berlaku untuk jumlah set pengelompokan yang tersirat. Misalnya, GROUP BY CUBE ((a), (b)) dihitung sebagai 4 set pengelompokan, bukan 2.
- Anda tidak dapat menggunakan konstanta sebagai kolom pengelompokan saat menggunakan ekstensi agregasi.
- Anda tidak dapat membuat kumpulan pengelompokan yang berisi kolom duplikat.

Klausa HAVING

Klausa HAVING menerapkan kondisi ke kumpulan hasil yang dikelompokkan perantara yang dikembalikan oleh kueri.

Sintaksis

```
[ HAVING condition ]
```

Misalnya, Anda dapat membatasi hasil fungsi SUM:

```
having sum(pricepaid) >10000
```

Kondisi HAVING diterapkan setelah semua kondisi klausa WHERE diterapkan dan operasi GROUP BY selesai.

Kondisi itu sendiri mengambil bentuk yang sama dengan kondisi klausa WHERE.

Catatan penggunaan

- Setiap kolom yang direferensikan dalam kondisi klausa HAVING harus berupa kolom pengelompokan atau kolom yang mengacu pada hasil dari fungsi agregat.

- Dalam klausa HAVING, Anda tidak dapat menentukan:
  - Nomor urut yang mengacu pada item daftar pilihan. Hanya klausa GROUP BY dan ORDER BY yang menerima nomor urut.

Contoh

Kueri berikut menghitung total penjualan tiket untuk semua acara berdasarkan nama, kemudian menghilangkan acara di mana total penjualan kurang dari $800.000. Kondisi HAVING diterapkan pada hasil fungsi agregat dalam daftar pilih:`sum(pricepaid)`.

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(pricepaid) > 800000
order by 2 desc, 1;

eventname        |    sum
-----------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
```

Kueri berikut menghitung kumpulan hasil yang serupa. Namun, dalam kasus ini, kondisi HAVING diterapkan ke agregat yang tidak ditentukan dalam daftar pilih:`sum(qtysold)`. Acara yang tidak menjual lebih dari 2.000 tiket dihilangkan dari hasil akhir.

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(qtysold) >2000
order by 2 desc, 1;

eventname        |    sum
-----------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
```

```
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
Chicago          |  790993.00
Spamalot         |  714307.00
```

Kueri berikut menghitung total penjualan tiket untuk semua acara berdasarkan nama, kemudian menghilangkan acara di mana total penjualan kurang dari $800.000. Kondisi HAVING diterapkan pada hasil fungsi agregat dalam daftar pilih menggunakan alias pp for`sum(pricepaid)`.

```
select eventname, sum(pricepaid) as pp
from sales join event on sales.eventid = event.eventid
group by 1
having pp > 800000
order by 2 desc, 1;

eventname        |    pp
-----------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
```

Klausul QUALIFY

Klausa QUALIFY memfilter hasil fungsi jendela yang dihitung sebelumnya sesuai dengan kondisi pencarian yang ditentukan pengguna. Anda dapat menggunakan klausa untuk menerapkan kondisi pemfilteran pada hasil fungsi jendela tanpa menggunakan subquery.

Hal ini mirip dengan klausa HAVING, yang menerapkan kondisi untuk menyaring baris lebih lanjut dari klausa WHERE. Perbedaan antara QUALIFY dan HAVING adalah bahwa hasil yang difilter dari klausa QUALIFY dapat didasarkan pada hasil menjalankan fungsi jendela pada data. Anda dapat menggunakan klausa QUALIFY dan HAVING dalam satu kueri.

Sintaksis

```
QUALIFY condition
```

> **Note**
>
> Jika Anda menggunakan klausa QUALIFY langsung setelah klausa FROM, nama relasi FROM harus memiliki alias yang ditentukan sebelum klausa QUALIFY.

Contoh

Contoh di bagian ini menggunakan data sampel di bawah ini.

```
create table store_sales (ss_sold_date date, ss_sold_time time,
               ss_item text, ss_sales_price float);
insert into store_sales values ('2022-01-01', '09:00:00', 'Product 1', 100.0),
                               ('2022-01-01', '11:00:00', 'Product 2', 500.0),
                               ('2022-01-01', '15:00:00', 'Product 3', 20.0),
                               ('2022-01-01', '17:00:00', 'Product 4', 1000.0),
                               ('2022-01-01', '18:00:00', 'Product 5', 30.0),
                               ('2022-01-02', '10:00:00', 'Product 6', 5000.0),
                               ('2022-01-02', '16:00:00', 'Product 7', 5.0);
```

Contoh berikut menunjukkan bagaimana menemukan dua barang paling mahal yang dijual setelah pukul 12:00 setiap hari.

```
SELECT *
FROM store_sales ss
WHERE ss_sold_time > time '12:00:00'
QUALIFY row_number()
OVER (PARTITION BY ss_sold_date ORDER BY ss_sales_price DESC) <= 2


 ss_sold_date | ss_sold_time |  ss_item  | ss_sales_price
--------------+--------------+-----------+----------------
 2022-01-01   | 17:00:00     | Product 4 |           1000
 2022-01-01   | 18:00:00     | Product 5 |             30
 2022-01-02   | 16:00:00     | Product 7 |              5
```

Anda kemudian dapat menemukan item terakhir yang terjual setiap hari.

```
SELECT *
FROM store_sales ss
QUALIFY last_value(ss_item)
OVER (PARTITION BY ss_sold_date ORDER BY ss_sold_time ASC
```

```
      ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) = ss_item;

ss_sold_date | ss_sold_time |  ss_item  | ss_sales_price
-------------+--------------+-----------+---------------
 2022-01-01   | 18:00:00     | Product 5 |             30
 2022-01-02   | 16:00:00     | Product 7 |              5
```

Contoh berikut mengembalikan catatan yang sama dengan kueri sebelumnya, item terakhir yang terjual setiap hari, tetapi tidak menggunakan klausa QUALIFY.

```
SELECT * FROM (
  SELECT *,
  last_value(ss_item)
  OVER (PARTITION BY ss_sold_date ORDER BY ss_sold_time ASC
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) ss_last_item
  FROM store_sales ss
)
WHERE ss_last_item = ss_item;

 ss_sold_date | ss_sold_time |  ss_item  | ss_sales_price | ss_last_item
-------------+--------------+-----------+----------------+--------------
 2022-01-02   | 16:00:00     | Product 7 |              5 | Product 7
 2022-01-01   | 18:00:00     | Product 5 |             30 | Product 5
```

UNION, INTERSECT, dan KECUALI

Topik

- Sintaksis
- Parameter
- Urutan evaluasi untuk operator set
- Catatan penggunaan
- Contoh kueri UNION
- Contoh kueri UNION ALL
- Contoh kueri INTERSECT
- Contoh KUERI KECUALI

Operator set UNION, INTERSECT, dan EXCEP T digunakan untuk membandingkan dan menggabungkan hasil dari dua ekspresi kueri terpisah. Misalnya, jika Anda ingin mengetahui

pengguna situs web mana yang merupakan pembeli dan penjual tetapi nama pengguna mereka disimpan dalam kolom atau tabel terpisah, Anda dapat menemukan per simpangan kedua jenis pengguna ini. Jika Anda ingin mengetahui pengguna situs web mana yang pembeli tetapi bukan penjual, Anda dapat menggunakan operator EXCEPT untuk menemukan perbedaan antara dua daftar pengguna. Jika Anda ingin membuat daftar semua pengguna, terlepas dari perannya, Anda dapat menggunakan operator UNION.

Sintaksis

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
query
```

Parameter

query

Ekspresi kueri yang sesuai, dalam bentuk daftar pilihannya, dengan ekspresi kueri kedua yang mengikuti operator UNION, INTERSECT, atau EXCEPT. Kedua ekspresi harus berisi jumlah kolom keluaran yang sama dengan tipe data yang kompatibel; jika tidak, kedua set hasil tidak dapat dibandingkan dan digabungkan. Operasi set tidak mengizinkan konversi implisit antara berbagai kategori tipe data; untuk informasi selengkapnya, lihatKompatibilitas dan konversi tipe.

Anda dapat membuat kueri yang berisi ekspresi kueri dalam jumlah tak terbatas dan menghubungkannya dengan operator UNION, INTERSECT, dan EXCEPT dalam kombinasi apa pun. Misalnya, struktur kueri berikut ini valid, dengan asumsi bahwa tabel T1, T2, dan T3 berisi kumpulan kolom yang kompatibel:

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

UNION

Tetapkan operasi yang mengembalikan baris dari dua ekspresi kueri, terlepas dari apakah baris berasal dari satu atau kedua ekspresi.

BERPOTONGAN

Tetapkan operasi yang mengembalikan baris yang berasal dari dua ekspresi kueri. Baris yang tidak dikembalikan oleh kedua ekspresi akan dibuang.

KECUALI | MINUS

Tetapkan operasi yang mengembalikan baris yang berasal dari salah satu dari dua ekspresi kueri. Agar memenuhi syarat untuk hasil, baris harus ada di tabel hasil pertama tetapi tidak yang kedua. MINUS dan EXCEPT adalah sinonim yang tepat.

SEMUA

Kata kunci ALL mempertahankan baris duplikat yang dihasilkan oleh UNION. Perilaku default saat kata kunci ALL tidak digunakan adalah membuang duplikat ini. INTERSECT ALL, EXCEPT ALL, dan MINUS ALL tidak didukung.

Urutan evaluasi untuk operator set

Operator himpunan UNION dan EXCEPT bersifat asosiatif kiri. Jika tanda kurung tidak ditentukan untuk mempengaruhi urutan prioritas, kombinasi operator himpunan ini dievaluasi dari kiri ke kanan. Misalnya, dalam kueri berikut, UNION T1 dan T2 dievaluasi terlebih dahulu, kemudian operasi EXCEPT dilakukan pada hasil UNION:

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

Operator INTERSECT diutamakan daripada operator UNION dan EXCEPT ketika kombinasi operator digunakan dalam kueri yang sama. Misalnya, kueri berikut mengevaluasi persimpangan T2 dan T3, kemudian menyatukan hasilnya dengan T1:

```
select * from t1
union
select * from t2
intersect
select * from t3
order by c1;
```

Dengan menambahkan tanda kurung, Anda dapat menerapkan urutan evaluasi yang berbeda. Dalam kasus berikut, hasil penyatuan T1 dan T2 berpotongan dengan T3, dan kueri cenderung menghasilkan hasil yang berbeda.

```
(select * from t1
union
select * from t2)
intersect
(select * from t3)
order by c1;
```

Catatan penggunaan

- Nama kolom yang dikembalikan dalam hasil kueri operasi set adalah nama kolom (atau alias) dari tabel dalam ekspresi kueri pertama. Karena nama kolom ini berpotensi menyesatkan, karena nilai dalam kolom berasal dari tabel di kedua sisi operator set, Anda mungkin ingin memberikan alias yang berarti untuk kumpulan hasil.
- Ekspresi query yang mendahului operator set tidak boleh berisi klausa ORDER BY. Klausa ORDER BY menghasilkan hasil yang diurutkan yang berarti hanya ketika digunakan di akhir kueri yang berisi operator set. Dalam hal ini, klausa ORDER BY berlaku untuk hasil akhir dari semua operasi yang ditetapkan. Kueri terluar juga dapat berisi klausa LIMIT dan OFFSET standar.
- Ketika kueri operator set mengembalikan hasil desimal, kolom hasil yang sesuai dipromosikan untuk mengembalikan presisi dan skala yang sama. Misalnya, dalam kueri berikut, di mana T1.REVENUE kolom DECIMAL (10,2) dan T2.REVENUE merupakan kolom DECIMAL (8,4), hasil desimal dipromosikan ke DECIMAL (12,4):

  ```
  select t1.revenue union select t2.revenue;
  ```

  Skal 4 anya karena itu adalah skala maksimum dari dua kolom. Ketepatan 12 karena T1.REVENUE membutuhkan 8 digit di sebelah kiri titik desimal (12 - 4 = 8). Promosi jenis ini memastikan bahwa semua nilai dari kedua sisi UNION cocok dengan hasilnya. Untuk nilai 64-bit, presisi hasil maksimum adalah 19 dan skala hasil maksimum adalah 18. Untuk nilai 128-bit, presisi hasil maksimum adalah 38 dan skala hasil maksimum adalah 37.

  Jika tipe data yang dihasilkan melebihi batas presisi dan skala Amazon Redshift, kueri mengembalikan kesalahan.
- Untuk operasi set, dua baris diperlakukan identik jika, untuk setiap pasangan kolom yang sesuai, dua nilai data sama atau keduanya NULL. Misalnya, jika tabel T1 dan T2 keduanya berisi satu

kolom dan satu baris, dan baris itu adalah NULL di kedua tabel, operasi INTERSECT di atas tabel tersebut mengembalikan baris itu.

Contoh kueri UNION

Dalam kueri UNION berikut, baris dalam tabel PENJUALAN digabungkan dengan baris dalam tabel LISTING. Tiga kolom yang kompatibel dipilih dari setiap tabel; dalam hal ini, kolom yang sesuai memiliki nama dan tipe data yang sama.

Kumpulan hasil akhir diurutkan berdasarkan kolom pertama dalam tabel LISTING dan terbatas pada 5 baris dengan nilai LISTID tertinggi.

```
select listid, sellerid, eventid from listing
union select listid, sellerid, eventid from sales
order by listid, sellerid, eventid desc limit 5;

listid | sellerid | eventid
--------+----------+---------
1 |    36861 |    7872
2 |    16002 |    4806
3 |    21461 |    4256
4 |     8117 |    4337
5 |     1616 |    8647
(5 rows)
```

Contoh berikut menunjukkan bagaimana Anda dapat menambahkan nilai literal ke output kueri UNION sehingga Anda dapat melihat ekspresi kueri mana yang menghasilkan setiap baris dalam kumpulan hasil. Kueri mengidentifikasi baris dari ekspresi kueri pertama sebagai “B” (untuk pembeli) dan baris dari ekspresi kueri kedua sebagai “S” (untuk penjual).

Kueri mengidentifikasi pembeli dan penjual untuk transaksi tiket yang harganya $10.000 atau lebih. Satu-satunya perbedaan antara dua ekspresi kueri di kedua sisi operator UNION adalah kolom penggabungan untuk tabel SALES.

```
select listid, lastname, firstname, username,
pricepaid as price, 'S' as buyorsell
from sales, users
where sales.sellerid=users.userid
and pricepaid >=10000
union
select listid, lastname, firstname, username, pricepaid,
```

```
'B' as buyorsell
from sales, users
where sales.buyerid=users.userid
and pricepaid >=10000
order by 1, 2, 3, 4, 5;

listid | lastname | firstname | username |   price   | buyorsell
--------+----------+-----------+----------+-----------+-----------
209658 | Lamb     | Colette   | VOR15LYI |  10000.00 | B
209658 | West     | Kato      | ELU81XAA |  10000.00 | S
212395 | Greer    | Harlan    | GXO71KOC |  12624.00 | S
212395 | Perry    | Cora      | YWR73YNZ |  12624.00 | B
215156 | Banks    | Patrick   | ZNQ69CLT |  10000.00 | S
215156 | Hayden   | Malachi   | BBG56AKU |  10000.00 | B
(6 rows)
```

Contoh berikut menggunakan operator UNION ALL karena baris duplikat, jika ditemukan, perlu dipertahankan dalam hasil. Untuk rangkaian ID peristiwa tertentu, kueri mengembalikan 0 baris atau lebih untuk setiap penjualan yang terkait dengan setiap peristiwa, dan 0 atau 1 baris untuk setiap daftar acara tersebut. ID peristiwa unik untuk setiap baris dalam tabel LISTING dan EVENT, tetapi mungkin ada beberapa penjualan untuk kombinasi ID acara dan daftar yang sama di tabel PENJUALAN.

Kolom ketiga dalam kumpulan hasil mengidentifikasi sumber baris. Jika berasal dari tabel PENJUALAN, itu ditandai “Ya” di kolom SALESROW. (SALESROW adalah alias untuk SALES.LISTID.) Jika baris berasal dari tabel LISTING, itu ditandai “Tidak” di kolom SALESROW.

Dalam hal ini, set hasil terdiri dari tiga baris penjualan untuk listing 500, acara 7787. Dengan kata lain, tiga transaksi berbeda terjadi untuk daftar dan kombinasi acara ini. Dua listing lainnya, 501 dan 502, tidak menghasilkan penjualan apa pun, jadi satu-satunya baris yang dihasilkan kueri untuk ID daftar ini berasal dari tabel LISTING (SALESROW = 'No').

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
```

```
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(6 rows)
```

Jika Anda menjalankan kueri yang sama tanpa kata kunci ALL, hasilnya hanya mempertahankan satu transaksi penjualan.

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(4 rows)
```

Contoh kueri UNION ALL

Contoh berikut menggunakan operator UNION ALL karena baris duplikat, jika ditemukan, perlu dipertahankan dalam hasil. Untuk rangkaian ID peristiwa tertentu, kueri mengembalikan 0 baris atau lebih untuk setiap penjualan yang terkait dengan setiap peristiwa, dan 0 atau 1 baris untuk setiap daftar acara tersebut. ID peristiwa unik untuk setiap baris dalam tabel LISTING dan EVENT, tetapi mungkin ada beberapa penjualan untuk kombinasi ID acara dan daftar yang sama di tabel PENJUALAN.

Kolom ketiga dalam kumpulan hasil mengidentifikasi sumber baris. Jika berasal dari tabel PENJUALAN, itu ditandai “Ya” di kolom SALESROW. (SALESROW adalah alias untuk SALES.LISTID.) Jika baris berasal dari tabel LISTING, itu ditandai “Tidak” di kolom SALESROW.

Dalam hal ini, set hasil terdiri dari tiga baris penjualan untuk listing 500, acara 7787. Dengan kata lain, tiga transaksi berbeda terjadi untuk daftar dan kombinasi acara ini. Dua listing lainnya, 501 dan 502, tidak menghasilkan penjualan apa pun, jadi satu-satunya baris yang dihasilkan kueri untuk ID daftar ini berasal dari tabel LISTING (SALESROW = 'No').

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(6 rows)
```

Jika Anda menjalankan kueri yang sama tanpa kata kunci ALL, hasilnya hanya mempertahankan satu transaksi penjualan.

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
```

```
(4 rows)
```

Contoh kueri INTERSECT

Bandingkan contoh berikut dengan contoh UNION pertama. Satu-satunya perbedaan antara kedua contoh adalah operator set yang digunakan, tetapi hasilnya sangat berbeda. Hanya satu baris yang sama:

```
235494 |    23875 |    8771
```

Ini adalah satu-satunya baris dalam hasil terbatas dari 5 baris yang ditemukan di kedua tabel.

```
select listid, sellerid, eventid from listing
intersect
select listid, sellerid, eventid from sales
order by listid desc, sellerid, eventid
limit 5;

listid | sellerid | eventid
--------+----------+---------
235494 |    23875 |    8771
235482 |     1067 |    2667
235479 |     1589 |    7303
235476 |    15550 |     793
235475 |    22306 |    7848
(5 rows)
```

Kueri berikut menemukan acara (yang tiketnya dijual) yang terjadi di tempat-tempat di New York City dan Los Angeles pada bulan Maret. Perbedaan antara dua ekspresi kueri adalah batasan pada kolom VENUECITY.

```
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='Los Angeles'
intersect
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='New York City'
order by eventname asc;

eventname
----------------------------
```

```
A Streetcar Named Desire
Dirty Dancing
Electra
Running with Annalise
Hairspray
Mary Poppins
November
Oliver!
Return To Forever
Rhinoceros
South Pacific
The 39 Steps
The Bacchae
The Caucasian Chalk Circle
The Country Girl
Wicked
Woyzeck
(16 rows)
```

Contoh KUERI KECUALI

Tabel KATEGORI dalam database TICKIT berisi 11 baris berikut:

```
 catid | catgroup |  catname  |                  catdesc
-------+----------+-----------+--------------------------------------------
   1   | Sports   | MLB       | Major League Baseball
   2   | Sports   | NHL       | National Hockey League
   3   | Sports   | NFL       | National Football League
   4   | Sports   | NBA       | National Basketball Association
   5   | Sports   | MLS       | Major League Soccer
   6   | Shows    | Musicals  | Musical theatre
   7   | Shows    | Plays     | All non-musical theatre
   8   | Shows    | Opera     | All opera and light opera
   9   | Concerts | Pop       | All rock and pop music concerts
  10   | Concerts | Jazz      | All jazz singers and bands
  11   | Concerts | Classical | All symphony, concerto, and choir concerts
(11 rows)
```

Asumsikan bahwa tabel CATEGORY_STAGE (tabel pementasan) berisi satu baris tambahan:

```
 catid | catgroup |  catname  |                  catdesc
-------+----------+-----------+--------------------------------------------
1 | Sports   | MLB       | Major League Baseball
```

```
2 | Sports   | NHL       | National Hockey League
3 | Sports   | NFL       | National Football League
4 | Sports   | NBA       | National Basketball Association
5 | Sports   | MLS       | Major League Soccer
6 | Shows    | Musicals  | Musical theatre
7 | Shows    | Plays     | All non-musical theatre
8 | Shows    | Opera     | All opera and light opera
9 | Concerts | Pop       | All rock and pop music concerts
10 | Concerts | Jazz      | All jazz singers and bands
11 | Concerts | Classical | All symphony, concerto, and choir concerts
12 | Concerts | Comedy    | All stand up comedy performances
(12 rows)
```

Kembalikan perbedaan antara kedua tabel. Dengan kata lain, kembalikan baris yang ada di tabel CATEGORY_STAGE tetapi tidak di tabel CATEGORY:

```
select * from category_stage
except
select * from category;

catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
12 | Concerts | Comedy  | All stand up comedy performances
(1 row)
```

Kueri setara berikut menggunakan sinonim MINUS.

```
select * from category_stage
minus
select * from category;

catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
12 | Concerts | Comedy  | All stand up comedy performances
(1 row)
```

Jika Anda membalikkan urutan ekspresi SELECT, kueri tidak mengembalikan baris.

Klausa ORDER BY

Topik

- Sintaksis

- Parameter
- Catatan penggunaan
- Contoh dengan ORDER BY

Klausa ORDER BY mengurutkan kumpulan hasil kueri.

Sintaksis

```
[ ORDER BY expression [ ASC | DESC ] ]
[ NULLS FIRST | NULLS LAST ]
[ LIMIT { count | ALL } ]
[ OFFSET start ]
```

Parameter

ekspresi

Ekspresi yang mendefinisikan urutan urutan kumpulan hasil kueri, biasanya dengan menentukan satu atau lebih kolom dalam daftar pilih. Hasil dikembalikan berdasarkan urutan bin UTF-8 er. Anda juga dapat menentukan yang berikut:

- Kolom yang tidak ada dalam daftar pilih
- Ekspresi dibentuk dari satu atau lebih kolom yang ada dalam tabel yang direferensikan oleh query
- Nomor urut yang mewakili posisi entri daftar pilih (atau posisi kolom dalam tabel jika tidak ada daftar pilih)
- Alias yang menentukan entri daftar pilih

Ketika klausa ORDER BY berisi beberapa ekspresi, kumpulan hasil diurutkan sesuai dengan ekspresi pertama, kemudian ekspresi kedua diterapkan ke baris yang memiliki nilai yang cocok dari ekspresi pertama, dan seterusnya.

ASC | DESKRIPSI

Opsi yang mendefinisikan urutan untuk ekspresi, sebagai berikut:

- ASC: naik (misalnya, rendah ke tinggi untuk nilai numerik dan 'A' ke 'Z' untuk string karakter). Jika tidak ada opsi yang ditentukan, data diurutkan dalam urutan menaik secara default.
- DESC: menurun (tinggi ke rendah untuk nilai numerik; 'Z' ke 'A' untuk string).

NULLS FIRST | NULL TERAKHIR

Opsi yang menentukan apakah nilai NULL harus diurutkan terlebih dahulu, sebelum nilai non-null, atau terakhir, setelah nilai non-null. Secara default, nilai NULL diurutkan dan diberi peringkat terakhir dalam urutan ASC, dan diurutkan dan diberi peringkat pertama dalam urutan DESC.

Nomor BATAS | SEMUA

Opsi yang mengontrol jumlah baris yang diurutkan yang dikembalikan oleh kueri. Angka LIMIT harus berupa bilangan bulat positif; nilai maksimumnya adalah2147483647.

LIMIT 0 mengembalikan tidak ada baris. Anda dapat menggunakan sintaks ini untuk tujuan pengujian: untuk memeriksa apakah kueri berjalan (tanpa menampilkan baris apa pun) atau untuk mengembalikan daftar kolom dari tabel. Klausa ORDER BY berlebihan jika Anda menggunakan LIMIT 0 untuk mengembalikan daftar kolom. Defaultnya adalah LIMIT ALL.

Mulai OFFSET

Opsi yang menentukan untuk melewati jumlah baris sebelum mulai sebelum mulai mengembalikan baris. Angka OFFSET harus berupa bilangan bulat positif; nilai maksimumnya adalah2147483647. Saat digunakan dengan opsi LIMIT, baris OFFSET dilewati sebelum mulai menghitung baris LIMIT yang dikembalikan. Jika opsi LIMIT tidak digunakan, jumlah baris dalam set hasil dikurangi dengan jumlah baris yang dilewati. Baris yang dilewati oleh klausa OFFSET masih harus dipindai, jadi mungkin tidak efisien untuk menggunakan nilai OFFSET yang besar.

Catatan penggunaan

Perhatikan perilaku yang diharapkan berikut dengan klausa ORDER BY:

- Nilai NULL dianggap “lebih tinggi” dari semua nilai lainnya. Dengan urutan pengurutan menaik default, nilai NULL diurutkan di akhir. Untuk mengubah perilaku ini, gunakan opsi NULLS FIRST.
- Ketika kueri tidak berisi klausa ORDER BY, sistem mengembalikan set hasil tanpa urutan baris yang dapat diprediksi. Kueri yang sama dijalankan dua kali mungkin mengembalikan hasil yang ditetapkan dalam urutan yang berbeda.
- Opsi LIMIT dan OFFSET dapat digunakan tanpa klausa ORDER BY; namun, untuk mengembalikan serangkaian baris yang konsisten, gunakan opsi ini bersama dengan ORDER BY.
- Dalam sistem paralel apa pun seperti Amazon Redshift, ketika ORDER BY tidak menghasilkan urutan yang unik, urutan baris tidak deterministik. Artinya, jika ekspresi ORDER BY menghasilkan

nilai duplikat, urutan pengembalian baris tersebut mungkin berbeda dari sistem lain atau dari satu proses Amazon Redshift ke yang berikutnya.

- Amazon Redshift tidak mendukung literal string dalam klausa ORDER BY.

Contoh dengan ORDER BY

Kembalikan semua 11 baris dari tabel CATEGORY, diurutkan berdasarkan kolom kedua, CATGROUP. Untuk hasil yang memiliki nilai CATGROUP yang sama, urutkan nilai kolom CATDESC dengan panjang string karakter. Kemudian urutkan berdasarkan kolom CATID dan CATNAME.

```
select * from category order by 2, length(catdesc), 1, 3;

catid | catgroup |  catname  |                 catdesc
------+----------+-----------+--------------------------------------
10    | Concerts | Jazz      | All jazz singers and bands
9     | Concerts | Pop       | All rock and pop music concerts
11    | Concerts | Classical | All symphony, concerto, and choir conce
6     | Shows    | Musicals  | Musical theatre
7     | Shows    | Plays     | All non-musical theatre
8     | Shows    | Opera     | All opera and light opera
5     | Sports   | MLS       | Major League Soccer
1     | Sports   | MLB       | Major League Baseball
2     | Sports   | NHL       | National Hockey League
3     | Sports   | NFL       | National Football League
4     | Sports   | NBA       | National Basketball Association
(11 rows)
```

Kembalikan kolom yang dipilih dari tabel SALES, diurutkan berdasarkan nilai QTYSOLD tertinggi. Batasi hasilnya ke 10 baris teratas:

```
select salesid, qtysold, pricepaid, commission, saletime from sales
order by qtysold, pricepaid, commission, salesid, saletime desc
limit 10;

salesid | qtysold | pricepaid | commission |     saletime
--------+---------+-----------+------------+---------------------
15401   |       8 |    272.00 |      40.80 | 2008-03-18 06:54:56
61683   |       8 |    296.00 |      44.40 | 2008-11-26 04:00:23
90528   |       8 |    328.00 |      49.20 | 2008-06-11 02:38:09
74549   |       8 |    336.00 |      50.40 | 2008-01-19 12:01:21
130232  |       8 |    352.00 |      52.80 | 2008-05-02 05:52:31
```

```
55243   |       8 |    384.00 |      57.60 | 2008-07-12 02:19:53
16004   |       8 |    440.00 |      66.00 | 2008-11-04 07:22:31
489     |       8 |    496.00 |      74.40 | 2008-08-03 05:48:55
4197    |       8 |    512.00 |      76.80 | 2008-03-23 11:35:33
16929   |       8 |    568.00 |      85.20 | 2008-12-19 02:59:33
(10 rows)
```

Kembalikan daftar kolom dan tidak ada baris dengan menggunakan sintaks LIMIT 0:

```
select * from venue limit 0;
venueid | venuename | venuecity | venuestate | venueseats
---------+-----------+-----------+------------+------------
(0 rows)
```

CONNECT BY klausa

Klausa CONNECT BY menentukan hubungan antara baris dalam hierarki. Anda dapat menggunakan CONNECT BY untuk memilih baris dalam urutan hierarkis dengan menggabungkan tabel ke dirinya sendiri dan memproses data hierarkis. Misalnya, Anda dapat menggunakannya untuk mengulang secara rekursif melalui bagan organisasi dan daftar data.

Proses kueri hierarkis dalam urutan sebagai berikut:

1. Jika klausa FROM memiliki gabungan, itu diproses terlebih dahulu.
2. Klausa CONNECT BY dievaluasi.
3. Klausul WHERE dievaluasi.

Sintaksis

```
[START WITH start_with_conditions]
CONNECT BY connect_by_conditions
```

> **Note**
>
> Meskipun START dan CONNECT bukan kata yang dicadangkan, gunakan pengidentifikasi yang dibatasi (tanda kutip ganda) atau AS jika Anda menggunakan START dan CONNECT sebagai alias tabel dalam kueri Anda untuk menghindari kegagalan saat runtime.

```
SELECT COUNT(*)
FROM Employee "start"
CONNECT BY PRIOR id = manager_id
START WITH name = 'John'
```

```
SELECT COUNT(*)
FROM Employee AS start
CONNECT BY PRIOR id = manager_id
START WITH name = 'John'
```

Parameter

mulai_dengan_ketentuan

Kondisi yang menentukan baris akar hierarki

sambungkan dengan syarat-syarat

Kondisi yang menentukan hubungan antara baris induk dan baris anak dari hierarki. Setidaknya satu kondisi harus memenuhi syarat dengan operator unary yang digunakan untuk merujuk ke baris induk.

```
PRIOR column = expression
-- or
expression > PRIOR column
```

Operator

Anda dapat menggunakan operator berikut dalam kueri CONNECT BY.

TINGKAT

Pseudokolom yang mengembalikan tingkat baris saat ini dalam hierarki. Mengembalikan 1 untuk baris akar, 2 untuk anak dari baris akar, dan seterusnya.

SEBELUMNYA

Operator unary yang mengevaluasi ekspresi untuk baris induk dari baris saat ini dalam hierarki.

Contoh

Contoh berikut adalah query CONNECT BY yang mengembalikan jumlah karyawan yang melaporkan secara langsung atau tidak langsung ke John, tidak lebih dalam dari 4 level.

```
SELECT id, name, manager_id
FROM employee
WHERE LEVEL < 4
START WITH name = 'John'
CONNECT BY PRIOR id = manager_id;
```

Berikut ini adalah hasil dari query.

```
id      name      manager_id
------+----------+--------------
  101     John        100
  102     Jorge       101
  103     Kwaku       101
  110     Liu         101
  201     Sofía       102
  106     Mateo       102
  110     Nikki       103
  104     Paulo       103
  105     Richard     103
  120     Saanvi      104
  200     Shirley     104
  205     Zhang       104
```

Definisi tabel untuk contoh ini:

```
CREATE TABLE employee (
   id INT,
   name VARCHAR(20),
   manager_id INT
   );
```

Berikut ini adalah baris yang dimasukkan ke dalam tabel.

```
INSERT INTO employee(id, name, manager_id)  VALUES
(100, 'Carlos', null),
(101, 'John', 100),
```

```
(102, 'Jorge', 101),
(103, 'Kwaku', 101),
(110, 'Liu', 101),
(106, 'Mateo', 102),
(110, 'Nikki', 103),
(104, 'Paulo', 103),
(105, 'Richard', 103),
(120, 'Saanvi', 104),
(200, 'Shirley', 104),
(201, 'Sofía', 102),
(205, 'Zhang', 104);
```

Berikut ini adalah bagan organisasi untuk departemen John.

Contoh subquery

Contoh berikut menunjukkan cara yang berbeda di mana subkueri masuk ke kueri SELECT. Lihat Contoh JOIN contoh lain dari penggunaan subquery.

PILIH daftar subkueri

Contoh berikut berisi subquery dalam daftar SELECT. Subkueri ini adalah skalar: hanya mengembalikan satu kolom dan satu nilai, yang diulang dalam hasil untuk setiap baris yang dikembalikan dari kueri luar. Kueri membandingkan nilai Q1SALES yang dihitung subkueri dengan nilai penjualan untuk dua kuartal lainnya (2 dan 3) pada tahun 2008, seperti yang didefinisikan oleh kueri luar.

```
select qtr, sum(pricepaid) as qtrsales,
(select sum(pricepaid)
from sales join date on sales.dateid=date.dateid
where qtr='1' and year=2008) as q1sales
from sales join date on sales.dateid=date.dateid
where qtr in('2','3') and year=2008
group by qtr
order by qtr;

qtr  |  qtrsales   |   q1sales
-------+-------------+-------------
2     | 30560050.00 | 24742065.00
3     | 31170237.00 | 24742065.00
(2 rows)
```

Subkueri klausa WHERE

Contoh berikut berisi subquery tabel dalam klausa WHERE. Subquery ini menghasilkan beberapa baris. Dalam hal ini, baris hanya berisi satu kolom, tetapi subquery tabel dapat berisi beberapa kolom dan baris, seperti tabel lainnya.

Permintaan menemukan 10 penjual teratas dalam hal tiket maksimum yang terjual. Daftar 10 teratas dibatasi oleh subquery, yang menghapus pengguna yang tinggal di kota-kota di mana ada tempat tiket. Kueri ini dapat ditulis dengan cara yang berbeda; misalnya, subquery dapat ditulis ulang sebagai gabungan dalam kueri utama.

```
select firstname, lastname, city, max(qtysold) as maxsold
from users join sales on users.userid=sales.sellerid
where users.city not in(select venuecity from venue)
group by firstname, lastname, city
order by maxsold desc, city desc
limit 10;

firstname | lastname  |      city      | maxsold
-----------+-----------+----------------+---------
Noah       | Guerrero | Worcester      |       8
Isadora    | Moss     | Winooski       |       8
Kieran     | Harrison | Westminster    |       8
Heidi      | Davis    | Warwick        |       8
Sara       | Anthony  | Waco           |       8
Bree       | Buck     | Valdez         |       8
Evangeline | Sampson  | Trenton        |       8
Kendall    | Keith    | Stillwater     |       8
Bertha     | Bishop   | Stevens Point  |       8
Patricia   | Anderson | South Portland |       8
(10 rows)
```

DENGAN subkueri klausa

Lihat DENGAN klausa.

Subkueri yang berkorelasi

Contoh berikut berisi subquery ber korelasi dalam klausa WHERE; jenis subquery ini berisi satu atau lebih korelasi antara kolom dan kolom yang dihasilkan oleh query luar. Dalam hal ini, korelasinya adalah`where s.listid=l.listid`. Untuk setiap baris yang dihasilkan kueri luar, subkueri dijalankan untuk memenuhi syarat atau mendiskualifikasi baris.

```
select salesid, listid, sum(pricepaid) from sales s
where qtysold=
(select max(numtickets) from listing l
where s.listid=l.listid)
group by 1,2
order by 1,2
limit 5;

salesid | listid |   sum
--------+--------+----------
 27     |     28 | 111.00
 81     |    103 | 181.00
 142    |    149 | 240.00
 146    |    152 | 231.00
 194    |    210 | 144.00
(5 rows)
```

Pola subquery berkorelasi yang tidak didukung

Perencana kueri menggunakan metode penulisan ulang kueri yang disebut subquery decorrelations untuk mengoptimalkan beberapa pola subquery berkorelasi untuk eksekusi di lingkungan MPP. Beberapa jenis subkueri yang berkorelasi mengikuti pola yang tidak dapat didekorasi dan tidak didukung oleh Amazon Redshift. Kueri yang berisi referensi korelasi berikut mengembalikan kesalahan:

- Referensi korelasi yang melewati blok kueri, juga dikenal sebagai “referensi korelasi tingkat lewat.” Misalnya, dalam kueri berikut, blok yang berisi referensi korelasi dan blok yang dilewati dihubungkan oleh predikat NOT EXISTS:

  ```
  select event.eventname from event
  where not exists
  (select * from listing
  where not exists
  (select * from sales where event.eventid=sales.eventid));
  ```

  Blok yang dilewati dalam hal ini adalah subkueri terhadap tabel LISTING. Referensi korelasi mengkorelasikan tabel EVENT dan SALES.

- Referensi korelasi dari subkueri yang merupakan bagian dari klausa ON dalam kueri luar:

  ```
  select * from category
  ```

```
left join event
on category.catid=event.catid and eventid =
(select max(eventid) from sales where sales.eventid=event.eventid);
```

Klausa ON berisi referensi korelasi dari SALES di subquery ke EVENT di kueri luar.

- Null-sensitive referensi korelasi ke tabel sistem Amazon Redshift. Contoh:

```
select attrelid
from stv_locks sl, pg_attribute
where sl.table_id=pg_attribute.attrelid and 1 not in
(select 1 from pg_opclass where sl.lock_owner = opcowner);
```

- Referensi korelasi dari dalam subquery yang berisi fungsi jendela.

```
select listid, qtysold
from sales s
where qtysold not in
(select sum(numtickets) over() from listing l where s.listid=l.listid);
```

- Referensi dalam kolom GROUP BY ke hasil subquery yang berkorelasi. Contoh:

```
select listing.listid,
(select count (sales.listid) from sales where sales.listid=listing.listid) as list
from listing
group by list, listing.listid;
```

- Referensi korelasi dari subquery dengan fungsi agregat dan klausa GROUP BY, terhubung ke kueri luar dengan predikat IN. (Pembatasan ini tidak berlaku untuk fungsi agregat MIN dan MAX.) Contoh:

```
select * from listing where listid in
(select sum(qtysold)
from sales
where numtickets>4
group by salesid);
```

PILIH KE

Memilih baris yang ditentukan oleh kueri apa pun dan menyisipkannya ke dalam tabel baru. Anda dapat menentukan apakah akan membuat tabel sementara atau persisten.

Sintaksis

```
[ WITH with_subquery [, ...] ]
SELECT
[ TOP number | [ ALL | DISTINCT ]
* | expression [ AS output_name ] [, ...] ]
[ EXCLUDE column_list ]
INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ [ START WITH expression ] CONNECT BY expression ]
[ GROUP BY ALL | expression [, ...] ]
[ HAVING condition ]
[ QUALIFY condition ]
[ { UNION | ALL | INTERSECT | EXCEPT | MINUS } query ]
[ ORDER BY expression [ ASC | DESC ] ]
[ LIMIT { number | ALL } ]
[ OFFSET start ]
```

Untuk detail tentang parameter perintah ini, lihatSELECT.

Contoh

Pilih semua baris dari tabel EVENT dan buat tabel NEWEVENT:

```
select * into newevent from event;
```

Pilih hasil dari kueri agregat ke dalam tabel sementara yang disebut PROFITS:

```
select username, lastname, sum(pricepaid-commission) as profit
into temp table profits
from sales, users
where sales.sellerid=users.userid
group by 1, 2
order by 3 desc;
```

SET

Menetapkan nilai parameter konfigurasi server. Gunakan perintah SET untuk mengganti setelan selama sesi atau transaksi saat ini saja.

Gunakan MENGATUR ULANG perintah untuk mengembalikan parameter ke nilai defaultnya.

Anda dapat mengubah parameter konfigurasi server dengan beberapa cara. Untuk informasi selengkapnya, lihat Memodifikasi konfigurasi server.

Sintaks

```
SET { [ SESSION | LOCAL ]
{ SEED | parameter_name } { TO | = }
{ value | 'value' | DEFAULT } |
SEED TO value }
```

Pernyataan berikut menetapkan nilai variabel konteks sesi.

```
SET { [ SESSION | LOCAL ]
variable_name { TO | = }
{ value | 'value'  }
```

Parameter

SESI

Menentukan bahwa pengaturan ini valid untuk sesi saat ini. Nilai default.

nama_variabel

Menentukan nama variabel konteks yang ditetapkan untuk sesi.

Konvensi penamaan adalah nama dua bagian yang dipisahkan oleh titik, misalnya identifi er.identifier. Hanya satu pemisah titik yang diperbolehkan. Gunakan pengi dentifikasi yang mengikuti aturan pengidentifikasi standar untuk Amazon Redshift Untuk informasi selengkapnya, lihatNama dan pengidentifikasi. Pengidentifikasi yang dibatasi tidak diperbolehkan.

LOKAL

Menentukan bahwa pengaturan tersebut valid untuk transaksi saat ini.

Nilai SEED TO

Menetapkan benih internal yang akan digunakan oleh fungsi RANDOM untuk pembuatan angka acak.

SET SEED mengambil nilai numerik antara 0 dan 1, dan mengalikan angka ini dengan (2^{31} -1) untuk digunakan dengan Fungsi RANDOM fungsi. Jika Anda menggunakan SET SEED sebelum

melakukan beberapa panggilan ACAK, RANDOM menghasilkan angka dalam urutan yang dapat diprediksi.

nama_parameter

Nama parameter yang akan diatur. Lihat Memodifikasi konfigurasi server untuk informasi tentang parameter.

nilai

Nilai parameter baru. Gunakan tanda kutip tunggal untuk mengatur nilai ke string tertentu. Jika menggunakan SET SEED, parameter ini berisi nilai SEED.

DEFAULT

Menetapkan parameter ke nilai default.

Contoh

Mengubah parameter untuk sesi saat ini

Contoh berikut menetapkan gaya tanggal:

```
set datestyle to 'SQL,DMY';
```

Mengatur grup kueri untuk manajemen beban kerja

Jika grup kueri terdaftar dalam definisi antrian sebagai bagian dari konfigurasi WLM cluster, Anda dapat mengatur parameter QUERY_GROUP ke nama grup kueri yang terdaftar. Kueri berikutnya ditetapkan ke antrian kueri terkait. Pengaturan QUERY_GROUP tetap berlaku selama durasi sesi atau sampai perintah RESET QUERY_GROUP ditemui.

Contoh ini menjalankan dua kueri sebagai bagian dari grup kueri 'prioritas', lalu mengatur ulang grup kueri.

```
set query_group to 'priority';
select tbl, count(*)from stv_blocklist;
select query, elapsed, substring from svl_qlog order by query desc limit 5;
reset query_group;
```

Untuk informasi selengkapnya, lihat Manajemen beban kerja.

Mengubah namespace identitas default untuk sesi

Pengguna database dapat mengaturdefault_identity_namespace. Contoh ini menunjukkan cara menggunakan SET SESSION untuk mengganti pengaturan selama sesi saat ini dan kemudian menampilkan nilai penyedia identitas baru. Ini paling umum digunakan ketika Anda menggunakan penyedia identitas dengan Redshift dan IAM Identity Center. Untuk informasi selengkapnya tentang menggunakan penyedia identitas dengan Redshift, lihat Menghubun gkan Redshift dengan Pusat Identitas IAM untuk memberi pengguna pengalaman masuk tunggal.

```
SET SESSION default_identity_namespace = 'MYCO';

SHOW default_identity_namespace;
```

Setelah menjalankan perintah, Anda dapat menjalankan pernyataan GRANT atau pernyataan CREATE seperti berikut:

```
GRANT SELECT ON TABLE mytable TO alice;

GRANT UPDATE ON TABLE mytable TO salesrole;

CREATE USER bob password 'md50c983d1a624280812631c5389e60d48c';
```

Dalam contoh ini, efek pengaturan namespace identitas default setara dengan mengawali setiap identitas dengan namespace. Dalam contoh alice ini, diganti denganMYCO:alice. Untuk informasi selengkapnya tentang pengaturan yang berkaitan dengan konfigurasi Redshift dengan Pusat Identitas IAM, lihat dan. MENGUBAH SISTEM UBAH PENYEDIA IDENTITAS

Menetapkan label untuk sekelompok kueri

Parameter QUERY_GROUP mendefinisikan label untuk satu atau lebih kueri yang dijalankan dalam sesi yang sama setelah perintah SET. Pada gilirannya, label ini dicatat saat kueri dijalankan dan dapat digunakan untuk membatasi hasil yang dikembalikan dari tabel sistem STL_QUERY dan STV_INFLIGHT dan tampilan SVL_QLOG.

```
show query_group;
query_group
-------------
unset
(1 row)

set query_group to '6 p.m.';
```

```
show query_group;
query_group
-------------
6 p.m.
(1 row)

select * from sales where salesid=500;
salesid | listid | sellerid | buyerid | eventid | dateid | ...
---------+--------+----------+---------+---------+--------+-----
500 |    504 |     3858 |    2123 |    5871 |   2052 | ...
(1 row)

reset query_group;

select query, trim(label) querygroup, pid, trim(querytxt) sql
from stl_query
where label ='6 p.m.';
query | querygroup |  pid  |                  sql
-------+------------+-------+---------------------------------------
57 | 6 p.m.     | 30711 | select * from sales where salesid=500;
(1 row)
```

Label grup kueri adalah mekanisme yang berguna untuk mengisolasi kueri individu atau kelompok kueri yang dijalankan sebagai bagian dari skrip. Anda tidak perlu mengidentifikasi dan melacak kueri berdasarkan ID mereka; Anda dapat melacaknya dengan label mereka.

Menetapkan nilai benih untuk pembuatan angka acak

Contoh berikut menggunakan opsi SEED dengan SET untuk menyebabkan fungsi RANDOM menghasilkan angka dalam urutan yang dapat diprediksi.

Pertama, kembalikan tiga bilangan bulat ACAK tanpa menetapkan nilai SEED terlebih dahulu:

```
select cast (random() * 100 as int);
int4
------
6
(1 row)

select cast (random() * 100 as int);
int4
------
```

```
68
(1 row)

select cast (random() * 100 as int);
int4
------
56
(1 row)
```

Sekarang, atur nilai SEED ke `.25`, dan kembalikan tiga angka RANDOM lagi:

```
set seed to .25;

select cast (random() * 100 as int);
int4
------
21
(1 row)

select cast (random() * 100 as int);
int4
------
79
(1 row)

select cast (random() * 100 as int);
int4
------
12
(1 row)
```

Terakhir, atur ulang nilai SEED ke `.25`, dan verifikasi bahwa RANDOM mengembalikan hasil yang sama dengan tiga panggilan sebelumnya:

```
set seed to .25;

select cast (random() * 100 as int);
int4
------
21
(1 row)

select cast (random() * 100 as int);
```

```
int4
------
79
(1 row)

select cast (random() * 100 as int);
int4
------
12
(1 row)
```

Contoh berikut menetapkan variabel konteks yang disesuaikan.

```
SET app_context.user_id TO 123;
SET app_context.user_id TO 'sample_variable_value';
```

MENGATUR OTORISASI SESI

Menetapkan nama pengguna untuk sesi saat ini.

Anda dapat menggunakan perintah SET SESSION AUTHORIZATION, misalnya, untuk menguji akses database dengan menjalankan sesi atau transaksi sementara sebagai pengguna yang tidak memiliki hak istimewa. Anda harus menjadi superuser database untuk menjalankan perintah ini.

Sintaksis

```
SET [ LOCAL ] SESSION AUTHORIZATION { user_name | DEFAULT }
```

Parameter

LOKAL

Menentukan bahwa pengaturan tersebut valid untuk transaksi saat ini. Menghilangkan parameter ini menentukan bahwa pengaturan valid untuk sesi saat ini.

nama pengguna

Nama pengguna yang akan diatur. Nama pengguna dapat ditulis sebagai pengidentifikasi atau string literal.

DEFAULT

Menetapkan nama pengguna sesi ke nilai default.

Contoh

Contoh berikut menetapkan nama pengguna untuk sesi saat ini menjadi`dwuser`:

```
SET SESSION AUTHORIZATION 'dwuser';
```

Contoh berikut menetapkan nama pengguna untuk transaksi saat ini menjadi`dwuser`:

```
SET LOCAL SESSION AUTHORIZATION 'dwuser';
```

Contoh ini menetapkan nama pengguna untuk sesi saat ini ke nama pengguna default:

```
SET SESSION AUTHORIZATION DEFAULT;
```

MENGATUR KARAKTERISTIK SESI

Perintah ini tidak digunakan lagi.

MENUNJUKKAN

Menampilkan nilai parameter konfigurasi server saat ini. Nilai ini mungkin spesifik untuk sesi saat ini jika perintah SET berlaku. Untuk daftar parameter konfigurasi, lihatReferensi konfigurasi.

Sintaksis

```
SHOW { parameter_name | ALL }
```

Pernyataan berikut menampilkan nilai variabel konteks sesi saat ini. Jika variabel tidak ada, Amazon Redshift menampilkan kesalahan.

```
SHOW variable_name
```

Parameter

nama_parameter

Menampilkan nilai saat ini dari parameter yang ditentukan.

SEMUA

Menampilkan nilai saat ini dari semua parameter.

nama_variabel

Menampilkan nilai variabel yang ditentukan saat ini.

Contoh

Contoh berikut menampilkan nilai untuk parameter query_group:

```
show query_group;

query_group

unset
(1 row)
```

Contoh berikut menampilkan daftar semua parameter dan nilainya:

```
show all;
name        |   setting
--------------------+--------------
datestyle          | ISO, MDY
extra_float_digits | 0
query_group        | unset
search_path        | $user,public
statement_timeout  | 0
```

Contoh berikut menampilkan nilai saat ini dari variabel yang ditentukan.

```
SHOW app_context.user_id;
```

TAMPILKAN PENGUNCIAN PENGGUNA

Gunakan SHOW USER LOCKOUT untuk mengaudit pengguna mana yang saat ini terkunci, menentukan apakah setiap lockout dilakukan secara otomatis atau manual, dan meninjau aktivitas login gagal baru-baru ini sebelum Anda membuka kunci pengguna. Pengguna federasi tidak disertakan. Perintah ini hanya tersedia untuk pengguna super. Untuk informasi selengkapnya, lihat upayan_login_gagal maksimal.

Hak istimewa yang diperlukan

Untuk menggunakan SHOW USER LOCKOUT, Anda harus menjadi superuser.

Sintaksis

```
SHOW USER LOCKOUT
```

Kolom keluaran

user_name

Nama pengguna database.

di_terkunci

Menentukan apakah pengguna saat ini terkunci.

lock_reason

Alasan pengguna dikunci. Nilainya adalah `automatic` saat Amazon Redshift mengunci pengguna setelah mencapai ambang upaya gagal, atau `manual` ketika superuser mengunci pengguna dengan ALTER USER NOLOGIN. Nilai ini adalah null ketika pengguna tidak dikunci.

terkunci_at

Stempel waktu saat pengguna dikunci.

terakhir_gagal_at

Stempel waktu dari upaya login terakhir yang gagal.

upaya_login_gagal

Jumlah upaya login gagal berturut-turut.

terkunci_oleh

ID pengguna yang mengunci pengguna ini. Nilai ini nol untuk penguncian otomatis.

Contoh

Contoh berikut menampilkan status lockout untuk semua pengguna berbasis kata sandi:

```
SHOW USER LOCKOUT;
```

```
 user_name    | is_locked | lock_reason      | locked_at           | last_failed_at
  | failed_login_attempts | locked_by
-------------+----------+-----------------+--------------------
+--------------------+----------------------+-----------
 data_analyst | true      | automatic        | 2026-08-10 14:02:11 | 2026-08-10
 14:02:11 |                     5 |
 sales_user   | true      | manual           | 2026-08-10 09:15:44 |
  |                     0 |       100
 report_user  | false     |                  |                     | 2026-08-09
 22:31:07 |                     2 |
(3 rows)
```

TAMPILKAN HIBAH KOLOM

Menampilkan hibah pada kolom dalam tabel.

Izin yang diperlukan

SHOW GRANTS untuk objek target hanya akan menampilkan hibah yang terlihat oleh pengguna saat ini. Hibah dapat dilihat oleh pengguna saat ini jika pengguna saat ini memenuhi salah satu kriteria berikut:

- Jadilah pengguna super
- Jadilah pengguna yang diberikan
- Diberikan pemilik peran yang diberikan
- Diberikan peran yang ditargetkan oleh hibah objek

Sintaksis

```
SHOW COLUMN GRANTS ON TABLE
{ database_name.schema_name.table_name | schema_name.table_name }
[FOR {username | ROLE role_name | PUBLIC}]
[LIMIT row_limit]
```

Parameter

database_name

Nama database yang berisi tabel target

schema_name

Nama skema yang berisi tabel target

table_name

Nama tabel target

nama pengguna

Hanya sertakan hibah ke nama pengguna dalam output

role_name

Hanya sertakan hibah ke role_name dalam output

PUBLIK

Hanya sertakan hibah kepada PUBLIK dalam output

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menunjukkan hibah kolom pada tabel demo_db.demo_schema.t100:

```
SHOW COLUMN GRANTS ON TABLE demo_db.demo_schema.t100;
 database_name | schema_name | table_name | column_name | object_type | privilege_type
 | identity_id | identity_name | identity_type | admin_option | privilege_scope |
 grantor_name
---------------+-------------+------------+-------------+-------------+----------------
+-------------+---------------+---------------+--------------+-----------------
+--------------
 demo_db       | demo_schema | t100       | b           | COLUMN      | UPDATE
    |         134 | bob           | user          | f            | COLUMN          |
 dbadmin
 demo_db       | demo_schema | t100       | a           | COLUMN      | SELECT
    |         130 | alice         | user          | f            | COLUMN          |
 dbadmin
 demo_db       | demo_schema | t100       | a           | COLUMN      | UPDATE
    |         130 | alice         | user          | f            | COLUMN          |
 dbadmin
```

Contoh berikut menunjukkan hibah kolom pada tabel demo_schema.t100 untuk pengguna bob:

```
SHOW COLUMN GRANTS ON TABLE demo_schema.t100 for bob;
 database_name | schema_name | table_name | column_name | object_type | privilege_type
 | identity_id | identity_name | identity_type | admin_option | privilege_scope |
 grantor_name
---------------+-------------+------------+-------------+-------------+---------------
+-------------+--------------+---------------+--------------+-----------------
+--------------
 demo_db       | demo_schema | t100       | b           | COLUMN      | UPDATE
    |         135 | bob           | user          | f             | COLUMN          |
 dbadmin
```

TAMPILKAN KOLOM

Menampilkan daftar kolom dalam tabel, bersama dengan beberapa atribut kolom.

Setiap baris keluaran terdiri dari daftar nama database yang dipisahkan koma, nama skema, nama tabel, nama kolom, posisi ordinal, default kolom, dapat dibatalkan, tipe data, panjang maksimum karakter, presisi numerik, komentar, jenis kunci pengurutan, urutan kunci urutan, kunci distribusi, pengkodean, dan pengumpulan. Untuk informasi selengkapnya tentang atribut ini, lihat SVV_ALL_KOLOM.

Jika lebih dari 10.000 kolom akan dihasilkan dari perintah SHOW COLUMNS, maka kesalahan dikembalikan.

Izin yang diperlukan

Untuk melihat kolom dalam tabel Amazon Redshift, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah pengguna super.
- Jadilah pemilik meja.
- Diberikan hak istimewa USAGE pada skema induk dan diberikan hak istimewa SELECT pada tabel atau diberikan hak istimewa SELECT pada kolom.

Sintaksis

```
SHOW COLUMNS FROM TABLE database_name.schema_name.table_name [LIKE 'filter_pattern']
 [LIMIT row_limit ]
```

Parameter

database_name

Nama database yang berisi tabel yang akan dicantumkan.

Untuk menampilkan tabel dalam sebuah AWS Glue Data Catalog, tent `awsdatacatalog` ukan () sebagai nama database, dan pastikan konfigurasi sistem `data_catalog_auto_mount` diatur `ketrue`. Untuk informasi selengkapnya, lihat MENGUBAH SISTEM.

nama skema

Nama skema yang berisi tabel yang akan dicantumkan.

Untuk menampilkan AWS Glue Data Catalog tabel, berikan nama AWS Glue database sebagai nama skema.

table_name

Nama tabel yang berisi kolom yang akan dicantumkan.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama kolom. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan setiap urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.

Jika filter_pattern tidak mengandung metacharacters, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator sama.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menunjukkan kolom dalam database Amazon Redshift bernama `sample_data_dev` yang ada dalam skema `tickit` dan tabel`event`.

```
SHOW COLUMNS FROM TABLE demo_schema.compound_sort_table;

  database_name | schema_name |     table_name      | column_name | ordinal_position
 | column_default | is_nullable |     data_type     | character_maximum_length |
 numeric_precision | numeric_scale | remarks | sort_key_type | sort_key | dist_key |
 encoding | collation
---------------+-------------+--------------------+-------------+-----------------
+----------------+-------------+-------------------+-------------------------
+------------------+--------------+---------+--------------+----------+----------
+----------+-----------
 demo_db       | demo_schema | compound_sort_table | id          |                1 |
              | YES         | integer           |                           |
     32 |             0 |         | COMPOUND      |        1 |        1 | delta32k |
 demo_db       | demo_schema | compound_sort_table | name        |                2 |
              | YES         | character varying |                      50 |
          |               |         | COMPOUND      |        2 |          | lzo      |
 default
 demo_db       | demo_schema | compound_sort_table | date_col    |                3 |
              | YES         | date              |                           |
       |               |         |               |        0 |          | delta    |
 demo_db       | demo_schema | compound_sort_table | amount      |                4 |
              | YES         | numeric           |                           |
     10 |             2 |         |               |        0 |          | mostly16 |
```

Contoh berikut menunjukkan tabel dalam AWS Glue Data Catalog database bernama `awsdatacatalog` yang ada dalam skema `batman` dan tabel`nation`. Output terbatas pada 2 baris.

```
SHOW COLUMNS FROM TABLE second_db.public.t22;

 database_name | schema_name | table_name | column_name | ordinal_position |
 column_default | is_nullable |          data_type          | character_maximum_length
 | numeric_precision | numeric_scale | remarks | sort_key_type | sort_key | dist_key |
 encoding | collation
---------------+-------------+------------+-------------+------------------
+----------------+-------------+-----------------------------
+-------------------------+-------------------+--------------+---------
+--------------+----------+----------+----------+-----------
```

```
second_db     | public      | t22        | col1        |               1 |
    | YES         | integer                     |                           |
     32 |             0 |         | INTERLEAVED   |       -1 |          | mostly8  |
second_db     | public      | t22        | col2        |               2 |
     | YES         | character varying           |                     100 |
          |               |         | INTERLEAVED   |        2 |          | text255  |
default
second_db     | public      | t22        | col3        |               3 |
    | YES         | timestamp without time zone |                           |
        |               |         |               |        0 |          | raw      |
second_db     | public      | t22        | col4        |               4 |
    | YES         | numeric                     |                           |
     10 |             2 |         |               |        0 |          | az64     |
```

TUNJUKKAN KENDALA

Menampilkan daftar kendala kunci utama dan kunci asing dalam tabel.

Izin yang diperlukan

Untuk menjalankan SHOW CONSTRAINTS pada tabel, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah superuser
- Jadilah pemilik meja
- Diberikan hak istimewa USAGE pada skema induk dan hak pilih pada tabel

Sintaksis

```
SHOW CONSTRAINTS {PRIMARY KEYS | FOREIGN KEYS [EXPORTED]}
FROM TABLE
{ database_name.schema_name.table_name | schema_name.table_name }
[LIMIT row_limit]
```

Parameter

database_name

Nama database yang berisi tabel target

nama skema

Nama skema yang berisi tabel target

table_name

Nama tabel target

DIEKSPOR

Ketika EXPORTED ditentukan, maka daftarkan semua kunci asing dari tabel lain yang mereferensikan tabel target.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menunjukkan batasan kunci utama dari tabel demo_db.demo_schema.pk1:

```
SHOW CONSTRAINTS PRIMARY KEYS FROM TABLE demo_db.demo_schema.pk1;
 database_name | schema_name | table_name | pk_name  | column_name | key_seq
---------------+-------------+------------+----------+-------------+---------
 demo_db       | demo_schema | pk1        | pk1_pkey | i           |       1
 demo_db       | demo_schema | pk1        | pk1_pkey | j           |       2
 demo_db       | demo_schema | pk1        | pk1_pkey | c           |       3
```

Contoh berikut menunjukkan batasan kunci asing dari tabel demo_schema.fk2:

```
SHOW CONSTRAINTS FOREIGN KEYS FROM TABLE demo_schema.fk2;
 pk_database_name | pk_schema_name | pk_table_name | pk_column_name | fk_database_name
 | fk_schema_name | fk_table_name | fk_column_name | key_seq |  fk_name   | pk_name  |
 update_rule | delete_rule | deferrability
------------------+----------------+---------------+----------------+-----------------
+----------------+---------------+----------------+---------+------------+----------
+-------------+-------------+--------------
 demo_db          | demo_schema    | pk1           | i              | demo_db
 | demo_schema    | fk2           | i              |       1 | fk2_i_fkey | pk1_pkey |
            |             |
 demo_db          | demo_schema    | pk1           | j              | demo_db
 | demo_schema    | fk2           | j              |       2 | fk2_i_fkey | pk1_pkey |
            |             |
```

```
 demo_db           | demo_schema     | pk1           | c               | demo_db
 | demo_schema     | fk2           | c               |       3 | fk2_i_fkey | pk1_pkey |
            |             |
```

Contoh berikut menunjukkan batasan kunci asing yang diekspor dari tabel demo_schema.pk1:

```
SHOW CONSTRAINTS FOREIGN KEYS EXPORTED FROM TABLE demo_schema.pk1;
 pk_database_name | pk_schema_name | pk_table_name | pk_column_name | fk_database_name
 | fk_schema_name | fk_table_name | fk_column_name | key_seq |     fk_name     |
 pk_name  | update_rule | delete_rule | deferrability
------------------+----------------+---------------+----------------+-----------------
+----------------+---------------+----------------+---------+----------------
+----------+-------------+-------------+---------------
 demo_db           | demo_schema     | pk1           | i               | demo_db
    | demo_schema     | fk2           | i               |       1 | fk2_i_fkey      |
 pk1_pkey |             |             |
 demo_db           | demo_schema     | pk1           | j               | demo_db
    | demo_schema     | fk2           | j               |       2 | fk2_i_fkey      |
 pk1_pkey |             |             |
 demo_db           | demo_schema     | pk1           | c               | demo_db
    | demo_schema     | fk2           | c               |       3 | fk2_i_fkey      |
 pk1_pkey |             |             |
 demo_db           | demo_schema     | pk1           | i               | demo_db
    | demo_schema     | other_fk      | i               |       1 | other_fk_i_fkey |
 pk1_pkey |             |             |
 demo_db           | demo_schema     | pk1           | j               | demo_db
    | demo_schema     | other_fk      | j               |       2 | other_fk_i_fkey |
 pk1_pkey |             |             |
 demo_db           | demo_schema     | pk1           | c               | demo_db
    | demo_schema     | other_fk      | c               |       3 | other_fk_i_fkey |
 pk1_pkey |             |             |
```

TAMPILKAN TABEL EKSTERNAL

Menunjukkan definisi tabel eksternal, termasuk atribut tabel dan atribut kolom. Anda dapat menggunakan output dari pernyataan SHOW EXTERNAL TABLE untuk membuat ulang tabel.

Untuk informasi selengkapnya tentang pembuatan tabel eksternal, lihatCREATE EXTERNAL TABLE.

Sintaksis

```
SHOW EXTERNAL TABLE [external_database].external_schema.table_name [ PARTITION ]
```

Parameter

database eksternal

Nama database eksternal terkait. Parameter ini bersifat opsional.

skema eksternal

Nama skema eksternal terkait.

table_name

Nama tabel yang akan ditampilkan.

PARTITION

Menampilkan pernyataan ALTER TABLE untuk menambahkan partisi ke definisi tabel.

Contoh

Contoh berikut didasarkan pada tabel eksternal yang didefinisikan sebagai berikut:

```
CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (
     csmallint smallint,
     cint int,
     cbigint bigint,
     cfloat float4,
     cdouble float8,
     cchar char(10),
     cvarchar varchar(255),
     cdecimal_small decimal(18,9),
     cdecimal_big decimal(30,15),
     ctimestamp TIMESTAMP,
     cboolean boolean,
     cstring varchar(16383)
)
PARTITIONED BY (cdate date, ctime TIMESTAMP)
STORED AS PARQUET
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';
```

Berikut ini adalah contoh perintah SHOW EXTERNAL TABLE dan output untuk tabel`my_schema.alldatatypes_parquet_test_partitioned`.

```
SHOW EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned;
```

```
"CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (
    csmallint smallint,
    cint int,
    cbigint bigint,
    cfloat float4,
    cdouble float8,
    cchar char(10),
    cvarchar varchar(255),
    cdecimal_small decimal(18,9),
    cdecimal_big decimal(30,15),
    ctimestamp timestamp,
    cboolean boolean,
    cstring varchar(16383)
)
PARTITIONED BY (cdate date, ctime timestamp)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';"
```

Berikut ini adalah contoh perintah SHOW EXTERNAL TABLE dan output untuk tabel yang sama, tetapi dengan database juga ditentukan dalam parameter.

```
SHOW EXTERNAL TABLE my_database.my_schema.alldatatypes_parquet_test_partitioned;
```

```
"CREATE EXTERNAL TABLE my_database.my_schema.alldatatypes_parquet_test_partitioned (
    csmallint smallint,
    cint int,
    cbigint bigint,
    cfloat float4,
    cdouble float8,
    cchar char(10),
    cvarchar varchar(255),
    cdecimal_small decimal(18,9),
    cdecimal_big decimal(30,15),
    ctimestamp timestamp,
    cboolean boolean,
    cstring varchar(16383)
)
```

```
PARTITIONED BY (cdate date, ctime timestamp)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';"
```

Berikut ini adalah contoh perintah SHOW EXTERNAL TABLE dan output saat menggunakan PARTITION parameter. Output berisi pernyataan ALTER TABLE untuk menambahkan partisi ke definisi tabel.

```
SHOW EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned PARTITION;
```

```
"CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (
    csmallint smallint,
    cint int,
    cbigint bigint,
    cfloat float4,
    cdouble float8,
    cchar char(10),
    cvarchar varchar(255),
    cdecimal_small decimal(18,9),
    cdecimal_big decimal(30,15),
    ctimestamp timestamp,
    cboolean boolean,
    cstring varchar(16383)
)
PARTITIONED BY (cdate date)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';
ALTER TABLE my_schema.alldatatypes_parquet_test_partitioned ADD IF NOT
 EXISTS PARTITION (cdate='2021-01-01') LOCATION 's3://amzn-s3-demo-bucket/
alldatatypes_parquet_partitioned2/cdate=2021-01-01';
ALTER TABLE my_schema.alldatatypes_parquet_test_partitioned ADD IF NOT
 EXISTS PARTITION (cdate='2021-01-02') LOCATION 's3://amzn-s3-demo-bucket/
alldatatypes_parquet_partitioned2/cdate=2021-01-02';"
```

TAMPILKAN DATABASE

Menampilkan database dari Katalog Data atau gudang data Amazon Redshift. SHOW DATABASE mencantumkan semua database yang dapat diakses, seperti, dalam gudang data, AWS Glue Data Catalog database (awsdatacatalog), database berbagi data, dan database Lake Formation.

Izin yang diperlukan

Semua database dapat dilihat oleh pengguna kecuali:

- Untuk database yang dibuat dari datashare dengan izin agar terlihat, pengguna saat ini harus diberikan izin USAGE pada database.

Sintaksis

Untuk menampilkan database dari gudang data Amazon Redshift:

```
SHOW DATABASES
[ LIKE '<expression>' ]
[ LIMIT row_limit ]
```

Untuk menampilkan database dari Katalog Data:

```
SHOW DATABASES FROM DATA CATALOG
[ ACCOUNT  '<id1>', '<id2>', ... ]
[ LIKE '<expression>' ]
[ IAM_ROLE default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' ]
[ LIMIT row_limit ]
```

Parameter

AKUN '<id1>', '<id2>',...

AWS Glue Data Catalog Akun dari mana untuk membuat daftar database. Menghilangkan parameter ini menunjukkan bahwa Amazon Redshift harus menampilkan database dari akun yang memiliki cluster.

SUKA '<expression>'

Memfilter daftar database ke database yang cocok dengan ekspresi yang Anda tentukan. Parameter ini mendukung pola yang menggunakan karakter wildcard% (persen) dan _ (garis bawah).

<account-id><role-name>IAM_ROLE default | 'SESI' | 'arn:aws:iam: :role/ '

Jika Anda menentukan peran IAM yang terkait dengan cluster saat menjalankan perintah SHOW DATABASE, Amazon Redshift akan menggunakan kredenSIAL peran saat Anda menjalankan kueri pada database.

Menentukan `default` kata kunci berarti menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster.

Gunakan `'SESSION'` jika Anda terhubung ke cluster Amazon Redshift menggunakan identitas federasi dan mengakses tabel dari database eksternal yang dibuat menggunakan perintahthe section called “BUAT BASIS DATA”. Untuk contoh penggunaan identitas federasi, lihat Menggunakan identitas federasi untuk mengelola akses Amazon Redshift ke sumber daya lokal dan tabel eksternal Amazon Redshift Spectrum, yang menjelaskan cara mengonfigurasi identitas federasi.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Minimal, peran IAM harus memiliki izin untuk melakukan operasi LIST pada bucket Amazon S3 untuk diakses dan operasi GET pada objek Amazon S3 yang berisi bucket. Untuk mempelajari lebih lanjut tentang database yang dibuat dari AWS Glue Data Catalog for datashares dan menggunakan IAM_ROLE, lihat Bek erja dengan berbagi data Lake Formation-managed sebagai konsumen.

Berikut ini menunjukkan sintaks untuk string parameter IAM_ROLE untuk satu ARN.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

Anda dapat mengrantai peran sehingga cluster Anda dapat mengambil peran IAM lain, mungkin milik akun lain. Anda dapat merangkai hingga 10 peran. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Amazon Redshift Spectrum.

Untuk peran IAM ini, lampirkan kebijakan izin IAM yang serupa dengan berikut ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:my-rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk langkah-langkah membuat peran IAM untuk digunakan dengan kueri federasi, lihat. Membuat peran rahasia dan IAM untuk menggunakan kueri federasi

> **Note**
>
> Jangan sertakan spasi dalam daftar peran yang dirantai.

Berikut ini menunjukkan sintaks untuk rantai tiga peran.

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-
account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

BATASI batas _baris

Klausa untuk MEMBATASI jumlah baris yang dikembalikan. Di mana row_limit adalah jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menampilkan semua database Katalog Data dari ID akun 123456789012.

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012'

  catalog_id  | database_name |                        database_arn
 |     type     |                                          target_database
                            | location | parameters
--------------+---------------+-----------------------------------------------------
+--------------
+-------------------------------------------------------------------------------------
+----------+------------
 123456789012 |   database1   | arn:aws:glue:us-east-1:123456789012:database/database1
 | Data Catalog |
                            |          |
 123456789012 |   database2   | arn:aws:glue:us-east-1:123456789012:database/database2
 | Data Catalog | arn:aws:redshift:us-
east-1:123456789012:datashare:035c45ea-61ce-86f0-8b75-19ac6102c3b7/database2 |
 |
```

Berikut ini adalah contoh yang menunjukkan cara menampilkan semua database Katalog Data dari ID akun 123456789012 saat menggunakan kredenSIAL peran IAM.

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012' IAM_ROLE default;
```

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012' IAM_ROLE <iam-role-arn>;
```

Contoh berikut menampilkan semua database di gudang data Amazon Redshift yang terhubung.

```
SHOW DATABASES
```

```
database_name  | database_owner | database_type        | database_acl | parameters |
 database_isolation_level
---------------+----------------+----------------------+--------------+------------
+--------------------
awsdatacatalog | 1              | auto mounted catalog | NULL         | UNKNOWN    |
 UNKNOWN
dev            | 1              | local                | NULL         | NULL       |
 Snapshot Isolation
```

TAMPILKAN FUNGSI

Menampilkan daftar fungsi dalam skema, bersama dengan informasi tentang objek yang terdaftar.

Setiap baris keluaran memiliki kolom database_name, schema_name, function_name, number_of_, argument_list, return_type, comments.

Jika lebih dari 10.000 baris akan dihasilkan dari SHOW FUNCTIONS, maka perintah tersebut menimbulkan kesalahan.

Izin yang diperlukan

Untuk melihat fungsi dalam skema Redshift, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah pengguna super
- Jadilah pemilik fungsi
- Diberikan hak istimewa USAGE pada skema induk dan diberikan EXECUTE pada fungsi

Sintaksis

```
SHOW FUNCTIONS FROM SCHEMA
[database_name.]schema_name
[LIKE 'filter_pattern'] [LIMIT row_limit]
```

Parameter

database_name

Nama database yang berisi fungsi yang akan dicantumkan.

nama skema

Nama skema yang berisi fungsi yang akan dicantumkan.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama fungsi. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan urutan nol atau lebih karakter
_	Cocok dengan karakter tunggal apa pun

Perhatikan bahwa filter_pattern hanya cocok dengan nama fungsi.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menunjukkan fungsi dari skema demo_db.demo_schema:

```
SHOW FUNCTIONS FROM SCHEMA demo_db.demo_schema;
 database_name | schema_name |    function_name     | number_of_arguments |
                      argument_list                                    |    return_type
  | remarks
---------------+-------------+----------------------+---------------------
+-----------------------------------------------------------------------------
+------------------+---------
 demo_db       | demo_schema | f2                   |                   6 | integer,
 character varying, numeric, date, timestamp without time zone, boolean | character
 varying |
 demo_db       | demo_schema | f_calculate_discount |                   2 | numeric,
 integer                                                                  | numeric
     |
 demo_db       | demo_schema | f_days_between       |                   2 | date, date
                                                                         | integer
  |
```

Contoh berikut menunjukkan fungsi dari skema demo_schema dengan nama yang diakhiri dengan 'diskon':

```
SHOW FUNCTIONS FROM SCHEMA demo_schema like '%discount';
 database_name | schema_name |    function_name     | number_of_arguments |
 argument_list   | return_type | remarks
---------------+-------------+----------------------+---------------------
+-----------------+-------------+---------
 demo_db       | demo_schema | f_calculate_discount |                   2 | numeric,
 integer | numeric     |
```

TUNJUKKAN HIBAH

Menampilkan hibah untuk pengguna, peran, atau objek. Objek dapat berupa database, skema, tabel, fungsi, atau template. Ketika Anda menentukan objek, seperti tabel atau fungsi, Anda harus memenuhi syarat dengan notasi dua bagian atau tiga bagian. Misalnya, `schema_name.table_name` atau `database_name.schema_name.table_name`.

Jika lebih dari 10.000 baris akan dihasilkan dari SHOW GRANTS, maka perintah tersebut menimbulkan kesalahan.

Izin yang diperlukan

Untuk menjalankan SHOW GRANTS untuk pengguna atau peran target, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah superuser
- Jadilah target pengguna
- Jadilah pemilik peran target
- Diberikan peran

SHOW GRANTS untuk objek target hanya akan menampilkan hibah yang terlihat oleh pengguna saat ini. Hibah dapat dilihat oleh pengguna saat ini jika pengguna saat ini memenuhi salah satu kriteria berikut:

- Jadilah pengguna super
- Jadilah target pengguna
- Diberikan pemilik peran yang diberikan

- Diberikan peran yang ditargetkan oleh hibah objek

Sintaksis

Berikut ini adalah sintaks untuk menampilkan hibah pada objek. Perhatikan bahwa cara kedua untuk menentukan fungsi hanya valid untuk skema eksternal dan database yang dibuat dari datashare.

```
SHOW GRANTS ON
{
 DATABASE database_name |
 FUNCTION {database_name.schema_name.function_name | schema_name.function_name }
 ( [ [ argname ] argtype [, ...] ] ) |
 FUNCTION {database_name.schema_name.function_name | schema_name.function_name } |
 SCHEMA {database_name.schema_name | schema_name} |
 { TABLE {database_name.schema_name.table_name | schema_name.table_name} | table_name }
 TEMPLATE {database_name.schema_name.template_name | template_name}
}
[FOR {username | ROLE role_name | PUBLIC}]
[LIMIT row_limit]
```

Berikut ini adalah sintaks untuk menampilkan hibah untuk pengguna atau peran.

```
SHOW GRANTS FOR
{username | ROLE role_name}
[FROM DATABASE database_name]
[LIMIT row_limit]
```

Parameter

database_name

Nama database untuk menampilkan hibah.

function_name

Nama fungsi untuk menampilkan hibah.

nama templat

Nama template untuk menampilkan hibah.

nama skema

Nama skema untuk menampilkan hibah.

table_name

Nama tabel untuk menunjukkan hibah.

UNTUK nama pengguna

Menunjukkan menunjukkan hibah untuk pengguna.

UNTUK ROLE role_name

Menunjukkan menunjukkan hibah untuk peran.

UNTUK UMUM

Menunjukkan menunjukkan hibah untuk PUBLIK.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menampilkan semua hibah pada database bernama`dev`.

```
SHOW GRANTS on database demo_db;

  database_name | privilege_type | identity_id | identity_name | identity_type |
 admin_option | privilege_scope | grantor_name
---------------+----------------+-------------+---------------+---------------
+--------------+-----------------+--------------
 demo_db       | ALTER          |         112 | alice         | user          | f
      | TABLES          | dbadmin
 demo_db       | TRUNCATE       |         112 | alice         | user          | f
      | TABLES          | dbadmin
 demo_db       | DROP           |         112 | alice         | user          | f
      | TABLES          | dbadmin
 demo_db       | INSERT         |         112 | alice         | user          | f
      | TABLES          | dbadmin
 demo_db       | TEMP           |           0 | public        | public        | f
      | DATABASE        | dbadmin
 demo_db       | SELECT         |         112 | alice         | user          | f
      | TABLES          | dbadmin
 demo_db       | UPDATE         |         112 | alice         | user          | f
      | TABLES          | dbadmin
```

```
 demo_db       | DELETE         |         112 | alice         | user          | f
      | TABLES          | dbadmin
 demo_db       | REFERENCES     |         112 | alice         | user          | f
      | TABLES          | dbadmin
```

Perintah berikut menunjukkan semua hibah pada skema bernamademo.

```
SHOW GRANTS ON SCHEMA demo_schema;

 schema_name | object_name | object_type | privilege_type | identity_id | identity_name
 | identity_type | admin_option | privilege_scope | database_name | grantor_name
-------------+-------------+-------------+----------------+-------------
+---------------+---------------+--------------+-----------------+---------------
+--------------
 demo_schema | demo_schema | SCHEMA      | ALTER          |         112 | alice
 | user          | f            | SCHEMA          | db1           | dbadmin
 demo_schema | demo_schema | SCHEMA      | DROP           |         112 | alice
 | user          | f            | SCHEMA          | db1           | dbadmin
 demo_schema | demo_schema | SCHEMA      | USAGE          |         112 | alice
 | user          | f            | SCHEMA          | db1           | dbadmin
 demo_schema | demo_schema | SCHEMA      | CREATE         |         112 | alice
 | user          | f            | SCHEMA          | db1           | dbadmin
```

Perintah berikut menunjukkan semua hibah untuk pengguna bernamaalice.

```
SHOW GRANTS FOR alice;

 database_name | schema_name | object_name | object_type | privilege_type | identity_id
 | identity_name | identity_type | privilege_scope | grantor_name
---------------+-------------+-------------+-------------+----------------
+-------------+---------------+---------------+-----------------+-------------
 demo_db       |             |             | DATABASE    | INSERT         |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       |             |             | DATABASE    | SELECT         |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       |             |             | DATABASE    | UPDATE         |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       |             |             | DATABASE    | DELETE         |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       |             |             | DATABASE    | REFERENCES     |         124
 | alice         | user          | TABLES          | dbadmin
```

```
 demo_db       |             |             | DATABASE    | DROP           |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       |             |             | DATABASE    | TRUNCATE       |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       |             |             | DATABASE    | ALTER          |         124
 | alice         | user          | TABLES          | dbadmin
 demo_db       | demo_schema |             | SCHEMA      | USAGE          |         124
 | alice         | user          | SCHEMA          | dbadmin
 demo_db       | demo_schema |             | SCHEMA      | CREATE         |         124
 | alice         | user          | SCHEMA          | dbadmin
 demo_db       | demo_schema |             | SCHEMA      | DROP           |         124
 | alice         | user          | SCHEMA          | dbadmin
 demo_db       | demo_schema |             | SCHEMA      | ALTER          |         124
 | alice         | user          | SCHEMA          | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | INSERT         |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | SELECT         |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | UPDATE         |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | DELETE         |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | RULE           |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | REFERENCES     |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | TRIGGER        |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | DROP           |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | TRUNCATE       |         124
 | alice         | user          | TABLE           | dbadmin
 demo_db       | demo_schema | t1          | TABLE       | ALTER          |         124
 | alice         | user          | TABLE           | dbadmin
```

```
SHOW GRANTS FOR alice FROM DATABASE second_db;
 database_name | schema_name | object_name | object_type | privilege_type | identity_id
 | identity_name | identity_type | privilege_scope | grantor_name
---------------+-------------+-------------+-------------+----------------
+-------------+---------------+---------------+-----------------+--------------
 second_db     | public      | t22         | TABLE       | SELECT         |         101
 | alice         | user          | TABLE           | dbadmin
```

Perintah berikut menunjukkan semua hibah pada tabel yang dinam `t3` ai untuk pengguna bernama`alice`. Perhatikan bahwa Anda dapat menggunakan notasi dua bagian atau tiga bagian untuk menentukan nama tabel.

```
SHOW GRANTS ON TABLE demo_db.demo_schema.t3 FOR ALICE;
 schema_name | object_name | object_type | privilege_type | identity_id | identity_name
 | identity_type | admin_option | privilege_scope | database_name | grantor_name
-------------+-------------+-------------+----------------+-------------
+---------------+---------------+--------------+-----------------+---------------
+--------------
 demo_schema | t3          | TABLE       | ALTER          |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | TRUNCATE       |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | DROP           |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | TRIGGER        |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | SELECT         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | INSERT         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | UPDATE         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | DELETE         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | RULE           |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | REFERENCES     |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin


SHOW GRANTS ON TABLE demo_schema.t3 FOR ALICE;
 schema_name | object_name | object_type | privilege_type | identity_id | identity_name
 | identity_type | admin_option | privilege_scope | database_name | grantor_name
-------------+-------------+-------------+----------------+-------------
+---------------+---------------+--------------+-----------------+---------------
+--------------
 demo_schema | t3          | TABLE       | ALTER          |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | TRUNCATE       |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
```

```
 demo_schema | t3          | TABLE       | DROP           |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | TRIGGER        |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | SELECT         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | INSERT         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | UPDATE         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | DELETE         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | RULE           |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 demo_schema | t3          | TABLE       | REFERENCES     |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
```

Contoh berikut menunjukkan semua hibah pada tabel bernama`t4`. Perhatikan berbagai cara yang dapat Anda tentukan nama tabel.

```
SHOW GRANTS ON t4;
 schema_name | object_name | object_type | privilege_type | identity_id | identity_name
 | identity_type | admin_option | privilege_scope | database_name | grantor_name
-------------+-------------+-------------+----------------+-------------
+---------------+---------------+--------------+-----------------+---------------
+--------------
 public      | t4          | TABLE       | ALTER          |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | TRUNCATE       |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | DROP           |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | TRIGGER        |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | SELECT         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | INSERT         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | UPDATE         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | DELETE         |         130 | alice
 | user          | f            | TABLE           | demo_db       | dbadmin
```

```
 public      | t4          | TABLE       | RULE            |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | REFERENCES      |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin

SHOW GRANTS ON TABLE public.t4;
 schema_name | object_name | object_type | privilege_type | identity_id | identity_name
 | identity_type | admin_option | privilege_scope | database_name | grantor_name
-------------+-------------+-------------+----------------+-------------
+---------------+---------------+--------------+-----------------+---------------
+--------------
 public      | t4          | TABLE       | ALTER           |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | TRUNCATE        |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | DROP            |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | TRIGGER         |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | SELECT          |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | INSERT          |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | UPDATE          |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | DELETE          |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | RULE            |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
 public      | t4          | TABLE       | REFERENCES      |          130 | alice
 | user          | f             | TABLE           | demo_db       | dbadmin
```

TUNJUKKAN MODEL

Menampilkan informasi yang berguna tentang model pembelajaran mesin, termasuk statusnya, parameter yang digunakan untuk membuatnya dan fungsi prediksi dengan jenis argumen inputnya. Anda dapat menggunakan informasi dari SHOW MODEL untuk membuat ulang model. Jika tabel dasar telah berubah, menjalankan CREATE MODEL dengan pernyataan SQL yang sama menghasilkan model yang berbeda. Informasi yang dikembalikan oleh SHOW MODEL berbeda untuk pemilik model dan pengguna dengan hak istimewa EXECUTE. SHOW MODEL menunjukkan output yang berbeda ketika model dilatih dari Amazon Redshift atau ketika model adalah model BYOM.

Sintaksis

```
SHOW MODEL ( ALL | model_name )
```

Parameter

SEMUA

Mengembalikan semua model yang dapat digunakan pengguna dan skema mereka.

nama_model

Nama modul. Nama model dalam skema harus unik.

Catatan penggunaan

Perintah SHOW MODEL mengembalikan yang berikut:

- Nama model.
- Skema tempat model dibuat.
- Pemilik model.
- Waktu pembuatan model.
- Status model, seperti READY, TRAINING, atau FAILED.
- Pesan alasan untuk model yang gagal.
- Kesalahan validasi jika model telah menyelesaikan pelatihan.
- Perkiraan biaya yang diperlukan untuk mendapatkan model untuk pendekatan non-BYOM. Hanya pemilik model yang dapat melihat informasi ini.
- Daftar parameter yang ditentukan pengguna dan nilainya, khususnya sebagai berikut:
  - Kolom TARGET yang ditentukan.
  - Jenis model, AUTO atau XGBoost.
  - Jenis masalah, seperti REGRESI, BINARY_CLASSIFICATION, MULTICLASS_CLASSIFICATION. Parameter ini khusus untuk AUTO.
  - Nama pekerjaan pelatihan Amazon SageMaker AI atau pekerjaan Amazon SageMaker AI Autopilot yang menciptakan model. Anda dapat menggunakan nama pekerjaan ini untuk menemukan informasi lebih lanjut tentang model di Amazon SageMaker AI.

- Tujuannya, seperti MSE, F1, Akurasi. Parameter ini khusus untuk AUTO.
- Nama fungsi yang dibuat.
- Jenis inferensi, lokal atau jarak jauh.
- Argumen input fungsi prediksi.
- Jenis argumen input fungsi prediksi untuk model yang tidak membawa model Anda sendiri (BYOM).
- Jenis pengembalian fungsi prediksi. Parameter ini khusus untuk BYOM.
- Nama titik akhir Amazon SageMaker AI untuk model BYOM dengan inferensi jarak jauh.
- Peran IAM. Hanya pemilik model yang dapat melihat ini.
- Bucket S3 digunakan. Hanya pemilik model yang dapat melihat ini.
- Kun AWS KMS cinya, jika ada yang disediakan. Hanya pemilik model yang dapat melihat ini.
- Waktu maksimum yang dapat dijalankan model.
- Jika tipe model bukan AUTO, maka Amazon Redshift juga menunjukkan daftar hyperparameter yang disediakan dan nilainya.

Anda juga dapat melihat beberapa informasi yang disediakan oleh SHOW MODEL di tabel katalog lain, seperti pg_proc. Amazon Redshift mengembalikan informasi tentang fungsi prediksi yang terdaftar di tabel katalog pg_proc. Informasi ini mencakup nama argumen input dan tipenya untuk fungsi prediksi. Amazon Redshift mengembalikan informasi yang sama dalam perintah SHOW MODEL.

```
SELECT * FROM pg_proc WHERE proname ILIKE '%<function_name>%';
```

Contoh

Contoh berikut menunjukkan output model show.

```
SHOW MODEL ALL;

Schema Name |  Model Name
------------+---------------
 public     | customer_churn
```

Pemilik customer_churn dapat melihat output berikut. Pengguna yang hanya memiliki hak istimewa EXECUTE tidak dapat melihat peran IAM, bucket Amazon S3, dan perkiraan biaya mode.

```
SHOW MODEL customer_churn;

       Key                  |           Value
----------------------------+-----------------------------------
 Model Name                 | customer_churn
 Schema Name                | public
 Owner                      | 'owner'
 Creation Time              | Sat, 15.01.2000 14:45:20
 Model State                | READY
 validation:F1              | 0.855
 Estimated Cost             | 5.7
                            |
 TRAINING DATA:             |
 Table                      | customer_data
 Target Column              | CHURN
                            |
 PARAMETERS:                |
 Model Type                 | auto
 Problem Type               | binary_classification
 Objective                  | f1
 Function Name              | predict_churn
 Function Parameters        | age zip average_daily_spend average_daily_cases
 Function Parameter Types   | int int float float
 IAM Role                   | 'iam_role'
 KMS Key                    | 'kms_key'
 Max Runtime                | 36000
```

TAMPILKAN DATA SHARE

Menampilkan saham inbound dan outbound dalam cluster baik dari akun yang sama atau di seluruh akun. Jika Anda tidak menentukan nama datashare, maka Amazon Redshift menampilkan semua data share di semua database di cluster. Pengguna yang memiliki hak istimewa ALTER dan SHARE dapat melihat saham yang mereka miliki hak istimewa.

Sintaksis

```
SHOW DATASHARES [ LIKE 'namepattern' ]
```

Parameter

SUKA

Klausa opsional yang membandingkan pola nama yang ditentukan dengan deskripsi datashare. Saat klausa ini digunakan, Amazon Redshift hanya menampilkan data share dengan nama yang cocok dengan pola nama yang ditentukan.

namepattern

Nama data yang diminta atau bagian dari nama yang akan dicocokkan menggunakan karakter wildcard.

Contoh

Contoh berikut menampilkan saham inbound dan outbound dalam sebuah cluster.

```
SHOW DATASHARES;
SHOW DATASHARES LIKE 'sales%';

share_name   | share_owner | source_database | consumer_database | share_type |
 createdate          | is_publicaccessible | share_acl | producer_account |
 producer_namespace
-------------+-------------+-----------------+-------------------+------------
+---------------------+---------------------+-----------+------------------
+--------------------------------------
'salesshare' | 100         | dev             |                   | outbound
 | 2020-12-09 01:22:54.| False               |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

TAMPILKAN PARAMETER

Menampilkan daftar parameter untuk a function/procedure, bersama dengan beberapa informasi tentang parameter.

Setiap baris keluaran memiliki kolom database_name, schema_name, procedure_name atau function_name, parameter_name, ordinal_position, parameter_type (), data_type, character_maximum_length, numeric_precision, numeric_scale, dan komentar. IN/OUT

Izin yang diperlukan

Untuk melihat a function/procedure dalam skema pergeseran merah, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah pengguna super
- Jadilah pemilik fungsi
- Diberikan hak istimewa USAGE pada skema induk dan diberikan EXECUTE pada fungsi

Sintaksis

```
SHOW PARAMETERS OF {FUNCTION| PROCEDURE}
[database_name.]schema_name.function_name(argtype [, ...] )
[LIKE 'filter_pattern'];
```

Parameter

database_name

Nama database yang berisi fungsi untuk daftar.

nama skema

Nama skema yang berisi fungsi untuk daftar.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama parameter. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan urutan nol atau lebih karakter
_	Cocok dengan karakter tunggal apa pun

Contoh

Contoh berikut menunjukkan parameter prosedur demo_db.demo_schema.f1:

```
SHOW PARAMETERS OF PROCEDURE demo_db.demo_schema.f1(VARCHAR, DECIMAL, DECIMAL,
 DECIMAL);
 database_name | schema_name | procedure_name |  parameter_name  | ordinal_position
 | parameter_type |          data_type          | character_maximum_length |
 numeric_precision | numeric_scale
---------------+-------------+----------------+------------------+------------------
+----------------+-----------------------------+--------------------------
+-------------------+--------------
 demo_db       | demo_schema | f1             | operation        |                1 |
 IN             | character varying           |                       10 |
      |
 demo_db       | demo_schema | f1             | value1           |                2 |
 IN             | numeric                     |                          |
   18 |             0
 demo_db       | demo_schema | f1             | value2           |                3 |
 IN             | numeric                     |                          |
   18 |             0
 demo_db       | demo_schema | f1             | result           |                4 |
 INOUT          | numeric                     |                          |
   18 |             0
 demo_db       | demo_schema | f1             | operation_status |                5 |
 OUT            | character varying           |                       50 |
      |
 demo_db       | demo_schema | f1             | calculation_time |                6 |
 OUT            | timestamp without time zone |                          |
      |
 demo_db       | demo_schema | f1             | is_successful    |                7 |
 OUT            | boolean                     |                          |
      |
```

Contoh berikut menunjukkan parameter prosedur demo_schema.f1 dengan nama yang dimulai dengan 'val':

```
SHOW PARAMETERS OF PROCEDURE demo_schema.f1(VARCHAR, DECIMAL, DECIMAL, DECIMAL) like
 'val%';
 database_name | schema_name | procedure_name | parameter_name | ordinal_position
 | parameter_type | data_type | character_maximum_length | numeric_precision |
 numeric_scale
---------------+-------------+----------------+----------------+------------------
+----------------+-----------+--------------------------+------------------
+---------------
 demo_db       | demo_schema | f1             | value1         |                2 | IN
            | numeric   |                          |                18 |             0
```

```
 demo_db       | demo_schema | f1             | value2         |               3 | IN
           | numeric   |                          |               18 |             0
```

Contoh berikut menunjukkan parameter fungsi demo_schema.f2:

```
SHOW PARAMETERS OF FUNCTION demo_schema.f2(INT, VARCHAR, DECIMAL, DATE, TIMESTAMP,
 BOOLEAN);
 database_name | schema_name | function_name | parameter_name  | ordinal_position
 | parameter_type |          data_type          | character_maximum_length |
 numeric_precision | numeric_scale
--------------+------------+--------------+-----------------+-----------------
+---------------+-----------------------------+-------------------------
+------------------+--------------
 demo_db       | demo_schema | f2           |                 |               0 |
 RETURN         | character varying           |                      -1 |
      |
 demo_db       | demo_schema | f2           | int_param       |               1 | IN
            | integer                    |                          |
 32 |             0
 demo_db       | demo_schema | f2           | varchar_param   |               2 | IN
           | character varying          |                      -1 |
  |
 demo_db       | demo_schema | f2           | decimal_param   |               3 | IN
           | numeric                    |                          |
  |
 demo_db       | demo_schema | f2           | date_param      |               4 | IN
           | date                       |                          |
  |
 demo_db       | demo_schema | f2           | timestamp_param |               5 | IN
           | timestamp without time zone |                          |
  |
 demo_db       | demo_schema | f2           | boolean_param   |               6 | IN
           | boolean                    |                          |
  |
```

TAMPILKAN KEBIJAKAN

Menampilkan kebijakan keamanan tingkat baris (RLS) dan penyamaran data dinamis (DDM) yang ditentukan dalam database, serta kebijakan RLS dan DDM yang diterapkan pada hubungan tertentu. Hanya pengguna super atau pengguna dengan `sys:secadmin` peran di database yang dapat melihat hasil kebijakan ini.

Sintaksis

```
SHOW { RLS | MASKING } POLICIES
[
    ON { database_name.schema_name.relation_name
       | schema_name.relation_name
       }
    [ FOR { user_name | ROLE role_name | PUBLIC } ]
  |
    FROM DATABASE database_name
]
[ LIMIT row_limit ];
```

Parameter

database_name

Nama database untuk menampilkan kebijakan.

nama skema

Nama skema relasi untuk menampilkan kebijakan terlampir.

nama relasi

Nama relasi untuk menampilkan kebijakan terlampir.

nama_pengguna

Nama pengguna yang kebijakan dilampirkan pada relasi.

role_name

Nama peran yang dilampirkan kebijakan pada hubungan.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

> **Note**
>
> Menampilkan kebijakan dari database yang berbeda dari database yang terhubung didukung di Katalog Izin Federasi Amazon Redshift. Perintah TAMPILKAN KEBIJAKAN mendukung kueri lintas database untuk semua database di gudang dengan izin federasi Amazon Redshift

Contoh

Perintah berikut menunjukkan kebijakan RLS dari database yang terhubung.

```
SHOW RLS POLICIES;

  policy_name   | policy_alias |                          policy_atts
                |
 policy_qual                                                                    |
 policy_enabled | policy_modified_by |    policy_modified_time
----------------+--------------
+-------------------------------------------------------------
+-------------------------------------------------------------------------------------------------
+----------------+--------------------+---------------------------
 policy_america | rls_table    | [{"colname":"region","type":"character varying(10)"}]
           | (("rls_table"."region" = CAST('USA' AS TEXT)) OR ("rls_table"."region" =
 CAST('CANADA' AS TEXT)) OR ("rls_table"."region" = CAST('Mexico' AS TEXT))) | t
       | admin              | 2025-11-07 14:57:27
```

Perintah berikut menunjukkan kebijakan masking dari database “sales_db.finance-catalog”;

```
SHOW MASKING POLICIES FROM DATABASE "sales_db@finance-catalog";

  policy_name  |                          input_columns                           |
                                        policy_expression
                         | policy_modified_by |    policy_modified_time
--------------+-----------------------------------------------------------------
+-------------------------------------------------------------------------------------------------
+--------------------+----------------------------
 hash_credit   | [{"colname":"credit_card","type":"character varying(256)"}]      |
 [{"expr":"SHA2((\"masked_table\".\"credit_card\" + CAST('testSalt' AS TEXT)), CAST(256
 AS INT4))","type":"text"}]   | admin              | 2025-11-07 16:05:54
 hash_username | [{"colname":"username","type":"character varying(256)"}]
 | [{"expr":"SHA2((\"masked_table\".\"username\" + CAST('otherTestSalt' AS TEXT)),
 CAST(256 AS INT4))","type":"text"}] | admin              | 2025-11-07 16:07:08
(2 rows)
```

Perintah berikut menunjukkan kebijakan RLS yang dilampirkan pada relasi sales_table;

```
SHOW RLS POLICIES ON sales_schema.sales_table;

  policy_name   | schema_name  | relation_name | relation_kind | grantor  |
 grantee          | grantee_kind | is_policy_on | is_rls_on | rls_conjunction_type
```

```
---------------+-------------+--------------+--------------+----------
+-------------------------+-------------+-------------+-----------
+----------------------
 policy_global  | sales_schema | sales_table   | table         | admin    |
 sales_analyst_role_global | role         | t            | t         | and
 policy_america | sales_schema | sales_table   | table         | admin    |
 sales_analyst_usa         | user         | t            | t         | and
```

Perintah berikut menunjukkan kebijakan masking yang dilampirkan pada relasi transaction_table dari database “sales_db.finance-catalog”.

```
SHOW MASKING POLICIES ON "sales_db@finance-catalog".sales_schema.transaction_table
 LIMIT 1;

  policy_name  | schema_name  |   relation_name   | relation_type | grantor  |
 grantee          | grantee_type | priority |   input_columns   |   output_columns
---------------+-------------+------------------+--------------+----------
+-------------------------+-------------+----------+------------------
+------------------
 hash_username | sales_schema | transaction_table | table         | admin    |
 transaction_analyst_role | role         |      100 | ["user_name"]     | ["user_name"]
```

Perintah berikut menunjukkan kebijakan RLS yang dilampirkan pada relasi sales_table dari database “sales_db.finance-catalog” untuk pengguna "_analyst_usa”. IAMR:sales

```
SHOW RLS POLICIES ON "sales_db@finance-catalog".sales_schema.sales_table FOR
 "IAMR:sales_analyst_usa";

  policy_name   | schema_name  | relation_name | relation_kind | grantor  |
 grantee           | grantee_kind | is_policy_on | is_rls_on | rls_conjunction_type
---------------+-------------+--------------+--------------+----------
+-----------------------+-------------+-------------+-----------
+----------------------
 policy_america | sales_schema | sales_table   | table         | admin    |
 IAMR:sales_analyst_usa | user         | t            | t         | and
```

Perintah berikut menunjukkan kebijakan RLS yang dilampirkan pada relasi transaction_table dari database “sales_db.finance-catalog” untuk peran transaction_analyst_role.

```
SHOW MASKING POLICIES ON sales_schema.transaction_table FOR ROLE
 transaction_analyst_role;
```

```
  policy_name  | schema_name  |   relation_name   | relation_type | grantor  |
 grantee          | grantee_type | priority | input_columns | output_columns
--------------+-------------+------------------+--------------+----------
+-------------------------+-------------+----------+---------------+---------------
 hash_username | sales_schema | transaction_table | table         | admin    |
 transaction_analyst_role | role         |      100 | ["user_name"] | ["user_name"]
```

TAMPILKAN PROSEDUR

Menunjukkan definisi prosedur tersimpan yang diberikan, termasuk tanda tangannya. Anda dapat menggunakan output dari SHOW PROCEDURE untuk membuat ulang prosedur yang disimpan.

Sintaksis

```
SHOW PROCEDURE sp_name [( [ [ argname ] [ argmode ] argtype [, ...] ] )]
```

Parameter

sp_nama

Nama prosedur yang akan ditampilkan.

[argname] [argmode] argtype

Masukan jenis argumen untuk mengidentifikasi prosedur tersimpan. Secara opsional, Anda dapat menyertakan tipe data argumen lengkap, termasuk argumen OUT. Bagian ini opsional jika nama prosedur yang disimpan unik (yaitu, tidak kelebihan beban).

Contoh

Contoh berikut menunjukkan definisi prosedur`test_spl2`.

```
show procedure test_sp2(int, varchar);
                                        Stored Procedure Definition
-------------------------------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE public.test_sp2(f1 integer, INOUT f2 character varying, OUT
 character varying)
LANGUAGE plpgsql
AS $_$
DECLARE
```

```
out_var alias for $3;
loop_var int;
BEGIN
IF f1 is null OR f2 is null THEN
RAISE EXCEPTION 'input cannot be null';
END IF;
CREATE TEMP TABLE etl(a int, b varchar);
FOR loop_var IN 1..f1 LOOP
insert into etl values (loop_var, f2);
f2 := f2 || '+' || f2;
END LOOP;
SELECT INTO out_var count(*) from etl;
END;
$_$

(1 row)
```

TUNJUKKAN PROSEDUR

Menampilkan daftar prosedur dalam skema, bersama dengan informasi tentang objek yang terdaftar.

Setiap baris keluaran memiliki kolom`database_name`,`schema_name`,`procedure_name`,`number_of_arguments`,`argument_list`,`retu` komentar.

Jika lebih dari 10.000 baris akan dihasilkan dari SHOW PROCEDURES, maka perintah tersebut menimbulkan kesalahan.

Izin yang diperlukan

Untuk melihat prosedur dalam skema Redshift, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah pengguna super
- Jadilah pemilik prosedur
- Diberikan hak istimewa USAGE pada skema induk dan diberikan EXECUTE pada prosedur

Sintaksis

```
SHOW PROCEDURES FROM SCHEMA
```

```
[database_name.]schema_name
[LIKE 'filter_pattern'] [LIMIT row_limit]
```

Parameter

database_name

Nama database yang berisi prosedur yang akan dicantumkan.

schema_name

Nama skema yang berisi prosedur yang akan dicantumkan.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama prosedur. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan urutan nol atau lebih karakter
_	Cocok dengan karakter tunggal apa pun

Perhatikan bahwa filter_pattern hanya cocok dengan nama prosedur.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menunjukkan prosedur dari skema demo_db.demo_schema:

```
SHOW PROCEDURES FROM SCHEMA demo_db.demo_schema;
 database_name | schema_name |  procedure_name   | number_of_arguments |
 argument_list                 |                          return_type
           | remarks
---------------+-------------+-------------------+---------------------
+-----------------------------------------
+-------------------------------------------------------------------+---------
```

```
 demo_db       | demo_schema | f1                |                   4 | character
 varying, numeric, numeric, numeric | numeric, character varying, timestamp without
 time zone, boolean |
 demo_db       | demo_schema | sp_get_result_set |                   2 | integer,
 refcursor                          | refcursor
                 |
 demo_db       | demo_schema | sp_process_data   |                   2 | numeric,
 numeric                            | numeric, character varying
                 |
```

Contoh berikut menunjukkan prosedur dari skema demo_schema dengan nama yang diakhiri dengan 'data':

```
SHOW PROCEDURES FROM SCHEMA demo_schema like '%data';
 database_name | schema_name | procedure_name  | number_of_arguments |  argument_list
 |        return_type         | remarks
---------------+-------------+-----------------+---------------------
+------------------+----------------------------+---------
 demo_db       | demo_schema | sp_process_data |                   2 | numeric, numeric
 | numeric, character varying |
```

TAMPILKAN SKEMA

Menampilkan daftar skema dalam database, bersama dengan beberapa atribut skema.

Setiap baris keluaran terdiri dari nama database, nama skema, pemilik skema, tipe skema, skema ACL, database sumber, dan opsi skema. Untuk informasi selengkapnya tentang atribut ini, lihat SVV_ALL_SCHEMAS.

Jika lebih dari 10.000 skema dapat dihasilkan dari perintah SHOW SCHEMAS, maka kesalahan dikembalikan.

Izin yang diperlukan

Untuk melihat skema dalam tabel Amazon Redshift, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah pengguna super.
- Jadilah pemilik skema.
- Diberikan hak istimewa USAGE pada skema.

Sintaksis

```
SHOW SCHEMAS FROM DATABASE database_name [LIKE 'filter_pattern'] [LIMIT row_limit ]
```

Parameter

database_name

Nama database yang berisi tabel yang akan dicantumkan.

Untuk menampilkan tabel dalam sebuah AWS Glue Data Catalog, tent `awsdatacatalog` ukan () sebagai nama database, dan pastikan konfigurasi sistem `data_catalog_auto_mount` diatur `ketrue`. Untuk informasi selengkapnya, lihat MENGUBAH SISTEM.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama skema. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
`%`	Cocokkan setiap urutan nol atau lebih karakter.
`_`	Cocokkan dengan karakter tunggal.

Jika filter_pattern tidak mengandung metacharacters, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator sama.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

Contoh berikut menunjukkan skema dari database Amazon Redshift bernama. `dev`

```
SHOW SCHEMAS FROM DATABASE dev;
```

```
 database_name |     schema_name      | schema_owner | schema_type |         schema_acl
          | source_database | schema_option
---------------+----------------------+--------------+-------------
+----------------------------+-----------------+--------------
 dev           | pg_automv            |            1 | local       |
          |                 |
 dev           | pg_catalog           |            1 | local       | jpuser=UC/
jpuser~=U/jpuser  |                 |
 dev           | public               |            1 | local       | jpuser=UC/
jpuser~=UC/jpuser |                 |
 dev           | information_schema   |            1 | local       | jpuser=UC/
jpuser~=U/jpuser  |                 |
 dev           | schemad79cd6d93bf043 |            1 | local       |
          |                 |
```

Contoh berikut menunjukkan skema dalam AWS Glue Data Catalog database bernamaawsdatacatalog. Jumlah maksimum baris keluaran adalah5.

```
SHOW SCHEMAS FROM DATABASE awsdatacatalog LIMIT 5;

 database_name  |     schema_name      | schema_owner | schema_type | schema_acl |
 source_database | schema_option
----------------+----------------------+--------------+-------------+------------
+-----------------+---------------
 awsdatacatalog | 000_too_many_glue_db |              | EXTERNAL    |            |
            |
 awsdatacatalog | 123_default          |              | EXTERNAL    |            |
            |
 awsdatacatalog | adhoc                |              | EXTERNAL    |            |
            |
 awsdatacatalog | all_shapes_10mb      |              | EXTERNAL    |            |
            |
 awsdatacatalog | all_shapes_1g        |              | EXTERNAL    |            |
            |
```

TAMPILKAN TABEL

Menunjukkan definisi tabel, termasuk atribut tabel, batasan tabel, atribut kolom, kolasi kolom dan batasan kolom. Anda dapat menggunakan output dari pernyataan SHOW TABLE untuk membuat ulang tabel.

Untuk informasi lebih lanjut tentang pembuatan tabel, lihatCREATE TABLE.

Sintaksis

```
SHOW TABLE [schema_name.]table_name
```

Parameter

nama skema

(Opsional) Nama skema terkait.

table_name

Nama tabel yang akan ditampilkan.

Contoh

Berikut ini adalah contoh output SHOW TABLE untuk tabel`sales`.

```
show table sales;
```

```
CREATE TABLE public.sales (
salesid integer NOT NULL ENCODE az64,
listid integer NOT NULL ENCODE az64 distkey,
sellerid integer NOT NULL ENCODE az64,
buyerid integer NOT NULL ENCODE az64,
eventid integer NOT NULL ENCODE az64,
dateid smallint NOT NULL,
qtysold smallint NOT NULL ENCODE az64,
pricepaid numeric(8,2) ENCODE az64,
commission numeric(8,2) ENCODE az64,
saletime timestamp without time zone ENCODE az64
)
DISTSTYLE KEY SORTKEY ( dateid );
```

Berikut ini adalah contoh output SHOW TABLE untuk tabel `category` dalam skema`public`. Kolasi database adalah CASE_SENSITIVE.

```
show table public.category;
```

```
CREATE TABLE public.category (
```

```
catid smallint NOT NULL distkey,
catgroup character varying(10) ENCODE lzo COLLATE case_sensitive,
catname character varying(10) ENCODE lzo COLLATE case_sensitive,
catdesc character varying(50) ENCODE lzo COLLATE case_sensitive
)
DISTSTYLE KEY SORTKEY ( catid );
```

Contoh berikut membuat tabel `foo` dengan kunci primer.

```
create table foo(a int PRIMARY KEY, b int);
```

Hasil SHOW TABLE menampilkan pernyataan create dengan semua properti `foo` tabel.

```
show table foo;
```

```
CREATE TABLE public.foo (
a integer NOT NULL ENCODE az64,
b integer ENCODE az64, PRIMARY KEY (a)
)
DISTSTYLE AUTO;
```

Contoh ini membuat tabel di mana kolom a mewarisi kolasi CASE_SENSITIVE default database, sementara b dan secara eksplisit disetel ke kolasi CASE_ c INSENSITIVE.

```
CREATE TABLE public.foo (
a CHAR,
b VARCHAR(10) COLLATE CASE_INSENSITIVE,
c SUPER COLLATE CASE_INSENSITIVE
);
```

Hasil SHOW TABLE menampilkan pernyataan create dengan semua properti `foo` tabel.

```
show table public.foo;
```

```
CREATE TABLE public.foo (
a character(1) ENCODE lzo COLLATE case_sensitive,
b character varying(10) ENCODE lzo COLLATE case_insensitive,
c super COLLATE case_insensitive
)
```

```
DISTSTYLE AUTO;
```

TAMPILKAN TABEL

Menampilkan daftar tabel dalam skema, bersama dengan beberapa atribut tabel.

Setiap baris keluaran terdiri dari nama database, nama skema, nama tabel, tipe tabel, ACL tabel, komentar, pemilik tabel, waktu terakhir diubah, waktu modifikasi terakhir, dist_style, dan sub-tipe tabel. Untuk informasi selengkapnya tentang atribut ini, lihat SVV_ALL_TABEL.

Stempel waktu modifikasi dan perubahan dapat tertinggal dari pembaruan tabel pada interval tetap sekitar 5 menit.

Jika lebih dari 10.000 tabel akan dihasilkan dari perintah SHOW TABLES, maka kesalahan dikembalikan.

Izin yang diperlukan

Untuk melihat tabel dalam skema Amazon Redshift, pengguna saat ini harus memenuhi salah satu kriteria berikut:

- Jadilah pengguna super.
- Jadilah pemilik meja.
- Diberikan hak istimewa USAGE pada skema induk dan diberikan hak istimewa SELECT pada tabel atau diberikan hak istimewa SELECT pada kolom mana pun dalam tabel.

Sintaksis

```
SHOW TABLES FROM SCHEMA database_name.schema_name [LIKE 'filter_pattern']
 [LIMIT row_limit ]
```

Parameter

database_name

Nama database yang berisi tabel yang akan dicantumkan.

Untuk menampilkan tabel dalam sebuah AWS Glue Data Catalog, tent `awsdatacatalog` ukan () sebagai nama database, dan pastikan konfigurasi sistem `data_catalog_auto_mount` diatur `ketrue`. Untuk informasi selengkapnya, lihat MENGUBAH SISTEM.

nama skema

Nama skema yang berisi tabel yang akan dicantumkan.

Untuk menampilkan AWS Glue Data Catalog tabel, berikan nama AWS Glue database sebagai nama skema.

filter_pattern

Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama tabel. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan setiap urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.

Jika filter_pattern tidak mengandung metacharacters, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator sama.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Row_limit bisa 0-10.000.

Contoh

```
SHOW TABLES FROM SCHEMA s1;

 database_name | schema_name |    table_name     | table_type |              table_acl
             | remarks | owner |     last_altered_time      |     last_modified_time
  | dist_style |   table_subtype
---------------+-------------+-------------------+------------
+-----------------------------------+---------+-------+---------------------------
+---------------------------+------------+-------------------
 dev           | s1          | late_binding_view | VIEW       | alice=arwdRxtDPA/
alice~bob=d/alice  |         | alice |                            |
        |            | LATE BINDING VIEW
 dev           | s1          | manual_mv         | VIEW       | alice=arwdRxtDPA/
alice~bob=P/alice  |         | alice |                            |
        |            | MATERIALIZED VIEW
```

```
 dev           | s1          | regular_view      | VIEW       | alice=arwdRxtDPA/
alice~bob=r/alice  |         | alice |                              |
        |            | REGULAR VIEW
 dev           | s1          | test_table        | TABLE      | alice=arwdRxtDPA/
alice~bob=rw/alice |         | alice | 2025-11-18 15:52:00.010452 | 2025-11-18
 15:44:34.856073 | AUTO (ALL) | REGULAR TABLE
```

```
SHOW TABLES FROM SCHEMA dev.s1 LIKE '%view' LIMIT 1;

 database_name | schema_name |    table_name     | table_type |              table_acl
              | remarks | owner | last_altered_time | last_modified_time | dist_style |
   table_subtype
---------------+-------------+-------------------+------------
+--------------------------------------+---------+-------+-------------------
+--------------------+------------+-------------------
 dev           | s1          | late_binding_view | VIEW       | {alice=arwdRxtDPA/
alice,bob=d/alice} |         | alice |                   |                    |
    | LATE BINDING VIEW
```

TAMPILKAN TEMPLATE

Menampilkan definisi lengkap template, termasuk nama yang sepenuhnya memenuhi syarat (database, skema, dan nama template) dan semua parameter. Outputnya adalah pernyataan CREATE TEMPLATE valid yang dapat Anda gunakan untuk membuat ulang template atau membuat template serupa dengan modifikasi.

Untuk informasi selengkapnya tentang pembuatan template, lihatBUAT TEMPLAT.

Izin yang diperlukan

Untuk melihat definisi template, Anda harus memiliki salah satu dari berikut ini:

- Hak istimewa pengguna super
- Hak istimewa PENGGUNAAN pada template dan hak istimewa PENGGUNAAN pada skema yang berisi template

Sintaksis

```
SHOW TEMPLATE [database_name.][schema_name.]template_name;
```

Parameter

database_name

(Opsional) Nama database tempat template dibuat. Jika tidak ditentukan, database saat ini digunakan.

nama skema

(Opsional) Nama skema tempat template dibuat. Jika tidak ditentukan, template dicari di jalur pencarian saat ini.

nama_templat

Nama templat.

Contoh

Berikut ini adalah contoh output SHOW TEMPLATE untuk template`test_template`:

```
CREATE TEMPLATE test_template FOR COPY AS NOLOAD DELIMITER ',' ENCODING UTF16
 ENCRYPTED;
```

```
SHOW TEMPLATE test_template;

CREATE OR REPLACE TEMPLATE dev.public.test_template FOR COPY AS ENCRYPTED NOLOAD
 ENCODING UTF16 DELIMITER ',';
```

Contoh berikut membuat template `demo_template` dalam skema`demo_schema`.

```
CREATE OR REPLACE TEMPLATE demo_schema.demo_template FOR COPY AS
ACCEPTANYDATE ACCEPTINVCHARS DATEFORMAT 'DD-MM-YYYY' EXPLICIT_IDS ROUNDEC
TIMEFORMAT  AS 'DD.MM.YYYY HH:MI:SS' TRUNCATECOLUMNS NULL  AS 'null_string';
```

```
SHOW TEMPLATE demo_schema.demo_template;

CREATE OR REPLACE TEMPLATE dev.demo_schema.demo_template FOR COPY AS TRUNCATECOLUMNS
 NULL 'null_string' EXPLICIT_IDS TIMEFORMAT 'DD.MM.YYYY HH:MI:SS' ACCEPTANYDATE ROUNDEC
 ACCEPTINVCHARS DATEFORMAT 'DD-MM-YYYY';
```

TAMPILKAN TEMPLAT

Menampilkan daftar template dalam skema, bersama dengan atributnya.

Setiap baris keluaran terdiri dari nama template, id template, jenis template, pemilik template, nama database, nama skema, waktu pembuatan, waktu modifikasi terakhir, dan terakhir dimodifikasi oleh.

Untuk detail template lengkap, termasuk parameter template, lihatSYS_REDSHIFT_TEMPLATE.

Izin yang diperlukan

Untuk melihat template dalam skema Amazon Redshift, Anda harus memiliki salah satu dari berikut ini:

- Hak istimewa pengguna super
- Hak istimewa PENGGUNAAN pada skema yang berisi template

Sintaksis

```
SHOW TEMPLATES FROM SCHEMA [database_name.]schema_name [LIKE 'filter_pattern']
 [LIMIT row_limit ];
```

Parameter

database_name

(Opsional) Nama database yang berisi templat yang akan dicantumkan. Jika tidak disediakan, gunakan database saat ini.

nama skema

Nama skema yang berisi templat yang akan dicantumkan.

filter_pattern

(Opsional) Ekspresi UTF-8 karakter yang valid dengan pola untuk mencocokkan nama template. Opsi LIKE melakukan kecocokan peka huruf besar-kecil yang mendukung metakarakter pencocokan pola berikut:

Metakarakter	Deskripsi
%	Cocokkan setiap urutan nol atau lebih karakter.
_	Cocokkan dengan karakter tunggal.

Jika filter_pattern tidak mengandung metacharacters, maka pola hanya mewakili string itu sendiri; dalam hal ini LIKE bertindak sama dengan operator sama.

batas_baris

Jumlah maksimum baris yang akan dikembalikan. Rentang yang valid adalah 0 hingga batas template pada cluster (default adalah 1000).

Contoh

```
SHOW TEMPLATES FROM SCHEMA s1;

 template_name          | template_id | template_type | template_owner | database_name
 | schema_name |        create_time         |     last_modified_time     |
 last_modified_by
------------------------+-------------+---------------+----------------+--------------
+-------------+----------------------------+----------------------------
+------------------
 template_maxerror      |      107685 | COPY          | alice          | dev
 | s1          | 2025-12-16 19:31:10.514076 | 2025-12-16 19:31:10.514076 |
 100
 json_template          |      107687 | COPY          | alice          | dev
 | s1          | 2025-12-16 19:31:33.229566 | 2025-12-16 19:31:33.229567 |
 100
 noload_template        |      107686 | COPY          | alice          | dev
 | s1          | 2025-12-16 19:31:17.370547 | 2025-12-16 19:31:17.370547 |
 100
 csv_delimiter_template |      107688 | COPY          | alice          | dev
 | s1          | 2025-12-16 19:31:42.354044 | 2025-12-16 19:31:42.354045 |
 100
```

```
SHOW TEMPLATES FROM SCHEMA dev.s1 LIKE '%template' LIMIT 1;
```

```
 template_name  | template_id | template_type | template_owner | database_name
 | schema_name |        create_time         |     last_modified_time     |
 last_modified_by
-----------------+-------------+---------------+----------------+--------------
+-------------+----------------------------+----------------------------
+------------------
 noload_template |      107686 | COPY          | alice          | dev           | s1
       | 2025-12-16 19:31:17.370547 | 2025-12-16 19:31:17.370547 |             100
```

TAMPILKAN TAMPILAN

Menampilkan definisi tampilan, termasuk untuk tampilan yang terwujud dan tampilan yang mengikat terlambat. Anda dapat menggunakan output dari pernyataan SHOW VIEW untuk membuat ulang tampilan.

Sintaksis

```
SHOW VIEW [schema_name.]view_name
```

Parameter

nama skema

(Opsional) Nama skema terkait.

nama_tampilan

Nama tampilan yang akan ditampilkan.

Contoh

Berikut ini adalah definisi tampilan untuk tampilan`LA_Venues_v`.

```
create view LA_Venues_v as select * from venue where venuecity='Los Angeles';
```

Berikut ini adalah contoh perintah SHOW VIEW dan output untuk tampilan yang ditentukan sebelumnya.

```
show view LA_Venues_v;
```

```
SELECT venue.venueid,
venue.venuename,
venue.venuecity,
venue.venuestate,
venue.venueseats
FROM venue WHERE ((venue.venuecity)::text = 'Los Angeles'::text);
```

Berikut ini adalah definisi tampilan untuk tampilan `public.Sports_v` dalam skema`public`.

```
create view public.Sports_v as select * from category where catgroup='Sports';
```

Berikut ini adalah contoh perintah SHOW VIEW dan output untuk tampilan yang ditentukan sebelumnya.

```
show view public.Sports_v;
```

```
SELECT category.catid,
category.catgroup,
category.catname,
category.catdesc
FROM category WHERE ((category.catgroup)::text = 'Sports'::text);
```

START TRANSACTION

Sinonim dari fungsi BEGIN.

Lihat MULAI.

MEMOTONG

Menghapus semua baris dari tabel tanpa melakukan pemindaian tabel: operasi ini adalah alternatif yang lebih cepat untuk operasi DELETE yang tidak memenuhi syarat. Untuk menjalankan perintah TRUNCATE, Anda harus memiliki izin TRUNCATE untuk tabel, menjadi pemilik tabel, atau menjadi superuser. Untuk memberikan izin untuk memotong tabel, gunakan HIBAH perintah.

TRUNCATE jauh lebih efisien daripada DELETE dan tidak memerlukan VACUUM dan ANALYSIS. Namun, ketahuilah bahwa TRUNCATE melakukan transaksi di mana ia dijalankan.

Sintaksis

```
TRUNCATE [ TABLE ] table_name
```

Perintah ini juga bekerja pada tampilan yang terwujud.

```
TRUNCATE materialized_view_name
```

Parameter

TABEL

Kata kunci opsional.

table_name

Tabel sementara atau persisten. Hanya pemilik tabel atau pengguna super yang dapat memotongnya.

Anda dapat memotong tabel apa pun, termasuk tabel yang direferensikan dalam batasan kunci asing.

Anda tidak perlu menyedot debu meja setelah memotongnya.

nama_tampilan terwujud

Pandangan yang terwujud.

Anda dapat memotong tampilan terwujud yang digunakan untuk. Streaming konsumsi ke tampilan yang terwujud

Catatan penggunaan

- Perintah TRUNCATE melakukan transaksi di mana ia dijalankan; oleh karena itu, Anda tidak dapat memutar kembali operasi TRUNCATE, dan perintah TRUNCATE dapat melakukan operasi lain ketika melakukan itu sendiri.
- Operasi TRUNCATE memegang kunci eksklusif saat dijalankan di Amazon Redshift streaming tampilan terwujud yang terhubung ke salah satu dari berikut ini:
  - Aliran data Amazon Kinesis
  - Topik Streaming Terkelola Amazon untuk Apache Kafka

- Aliran eksternal yang didukung, seperti topik Confluent Cloud Kafka

Untuk informasi selengkapnya, lihat Streaming konsumsi ke tampilan yang terwujud.

Contoh

Gunakan perintah TRUNCATE untuk menghapus semua baris dari tabel CATEGORY:

```
truncate category;
```

Mencoba memutar kembali operasi TRUNCATE:

```
begin;

truncate date;

rollback;

select count(*) from date;
count
-------
0
(1 row)
```

Tabel DATE tetap kosong setelah perintah ROLLBACK karena perintah TRUNCATE dilakukan secara otomatis.

Contoh berikut menggunakan perintah TRUNCATE untuk menghapus semua baris dari tampilan yang terwujud.

```
truncate my_materialized_view;
```

Ini menghapus semua catatan dalam tampilan terwujud dan membiarkan tampilan terwujud dan skemanya tetap utuh. Dalam kueri, nama tampilan yang terwujud adalah sampel.

MEMBONGKAR

Client-side enkripsi untuk perintah COPY dan UNLOAD tidak akan lagi terbuka untuk pelanggan baru mulai 30 April 2025. Jika Anda menggunakan enkripsi sisi klien dengan perintah COPY dan

UNLOAD dalam 12 bulan sebelum 30 April 2025, Anda dapat terus menggunakan enkripsi sisi klien dengan perintah COPY atau UNLOAD hingga 30 April 2026. Setelah 30 April 2026, Anda tidak akan dapat menggunakan enkripsi sisi klien untuk COPY dan UNLOAD. Kami menyarankan Anda beralih menggunakan enkripsi sisi server untuk COPY dan UNLOAD sesegera mungkin. Jika Anda sudah menggunakan enkripsi sisi server untuk COPY dan UNLOAD, tidak ada perubahan dan Anda dapat terus menggunakannya tanpa mengubah kueri Anda. Untuk informasi selengkapnya tentang enkripsi untuk COPY dan UNLOAD, lihat parameter ENCRYPTED di bawah ini.

Membongkar hasil kueri ke satu atau beberapa file teks, JSON, atau Apache Parquet di Amazon S3, menggunakan enkripsi sisi server Amazon S3 (). SSE-S3 Anda juga dapat menentukan enkripsi sisi server dengan AWS Key Management Service kunci ()SSE-KMS.

Secara default, format file yang tidak dimuat adalah teks dipisahkan pipa (). |

Anda dapat mengelola ukuran file di Amazon S3, dan dengan ekstensi jumlah file, dengan mengatur parameter MAXFILESIZE. Pastikan bahwa rentang IP S3 ditambahkan ke daftar izin Anda. Untuk mempelajari selengkapnya tentang rentang IP S3 yang diperlukan, lihat Isolasi jaringan.

Anda dapat membongkar hasil kueri Amazon Redshift ke danau data Amazon S3 Anda di Apache Parquet, format penyimpanan kolom terbuka yang efisien untuk analitik. Format parket hingga 2x lebih cepat untuk dibongkar dan mengkonsumsi penyimpanan hingga 6x lebih sedikit di Amazon S3, dibandingkan dengan format teks. Ini memungkinkan Anda menyimpan transformasi dan pengayaan data yang telah Anda lakukan di Amazon S3 ke dalam data lake Amazon S3 Anda dalam format terbuka. Anda kemudian dapat menganalisis data Anda dengan Redshift Spectrum dan AWS layanan lain seperti Amazon Athena, Amazon EMR, dan Amazon AI. SageMaker

Untuk informasi selengkapnya dan contoh skenario tentang menggunakan perintah UNLOAD, lihatMembongkar data di Amazon Redshift.

Hak istimewa dan izin yang diperlukan

Agar perintah UNLOAD berhasil, setidaknya diperlukan hak istimewa SELECT pada data dalam database, bersama dengan izin untuk menulis ke lokasi Amazon S3. Untuk informasi tentang izin untuk mengakses AWS sumber daya untuk perintah UNLOAD, lihatIzin untuk mengakses lainnya AWS Sumber daya.

Untuk menerapkan izin hak istimewa terkecil, ikuti rekomendasi ini untuk hanya memberikan izin, sesuai kebutuhan, kepada pengguna yang menjalankan perintah.

- Pengguna harus memiliki hak istimewa SELECT pada data. Untuk informasi tentang cara membatasi hak akses database, lihatHIBAH.
- Pengguna memerlukan izin untuk mengambil peran IAM untuk menulis ke bucket Amazon S3 di Anda Akun AWS. Untuk membatasi akses bagi pengguna database untuk mengambil peran, lihat Membatasi akses ke peran IAM di Panduan Manajemen Amazon Redshift.
- Pengguna memerlukan akses ke bucket Amazon S3. Untuk membatasi izin menggunakan kebijakan bucket Amazon S3, lihat Kebijakan Bucket untuk Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Sintaks

```
UNLOAD ('select-statement')
TO 's3://object-path/name-prefix'
authorization
[ option, ...]

where authorization is
IAM_ROLE { default | 'arn:aws:iam::<Akun AWS-id-1>:role/<role-name>[,arn:aws:iam::<Akun
 AWS-id-2>:role/<role-name>][,...]' }

where option is
| [ FORMAT [ AS ] ] CSV | PARQUET | JSON
| PARTITION BY ( column_name [, ... ] ) [ INCLUDE ]
| MANIFEST [ VERBOSE ]
| HEADER
| DELIMITER [ AS ] 'delimiter-char'
| FIXEDWIDTH [ AS ] 'fixedwidth-spec'
| ENCRYPTED [ AUTO ]
| BZIP2
| GZIP
| ZSTD
| ADDQUOTES
| NULL [ AS ] 'null-string'
| ESCAPE
| ALLOWOVERWRITE
| CLEANPATH
| PARALLEL [ { ON | TRUE } | { OFF | FALSE } ]
| MAXFILESIZE [AS] max-size [ MB | GB ]
| ROWGROUPSIZE [AS] size [ MB | GB ]
| REGION [AS] 'aws-region' }
| EXTENSION 'extension-name'
```

Parameter

('sel eksi pernyataan')

Kueri SELECT. Hasil kueri dibongkar. Dalam kebanyakan kasus, ada baiknya membongkar data dalam urutan yang diurutkan dengan menentukan klausa ORDER BY dalam kueri. Pendekatan ini menghemat waktu yang diperlukan untuk mengurutkan data saat dimuat ulang.

Kueri harus dilampirkan dalam tanda kutip tunggal seperti yang ditunjukkan berikut:

```
('select * from venue order by venueid')
```

> **Note**
>
> Jika kueri Anda berisi tanda kutip (misalnya untuk melampirkan nilai literal), letakkan literal di antara dua set tanda kutip tunggal—Anda juga harus melampirkan kueri di antara tanda kutip tunggal:
>
> ```
> ('select * from venue where venuestate=''NV''')
> ```

UNTUK 's3://objek- path/name - awalan '

Jalur lengkap, termasuk nama bucket, ke lokasi di Amazon S3 tempat Amazon Redshift menulis objek file keluaran, termasuk file manifes jika MANIFEST ditentukan. Nama objek diawali dengan nam a-awalan. Jika Anda menggunakan`PARTITION BY`, garis miring maju (/) secara otomatis ditambahkan ke akhir nilai awalan nama jika diperlukan. Untuk keamanan tambahan, UNLOAD terhubung ke Amazon S3 menggunakan koneksi HTTPS. Secara default, UNLOAD menulis satu atau lebih file per irisan. UNLOAD menambahkan nomor irisan dan nomor bagian ke awalan nama yang ditentukan sebagai berikut:

<object-path>/*<name-prefix><slice-number>*_part_*<part-number>*.

Jika MANIFEST ditentukan, file manifes ditulis sebagai berikut:

<object_path>/*<name_prefix>*manifest.

Jika PARALLEL ditentukan OFF, file data ditulis sebagai berikut:

<object_path>/*<name_prefix><part-number>*.

UNLOAD secara otomatis membuat file terenkripsi menggunakan enkripsi sisi server Amazon S3 (SSE), termasuk file manifes jika MANIFEST digunakan. Perintah COPY secara otomatis membaca file terenkripsi sisi server selama operasi pemuatan. Anda dapat mengunduh file terenkripsi sisi server secara transparan dari bucket Anda menggunakan konsol Amazon S3 atau API. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Server-Side Enkripsi.

> **Important**
>
> REGION diperlukan jika bucket Amazon S3 tidak Wilayah AWS sama dengan database Amazon Redshift.

otorisasi

Perintah UNLOAD memerlukan otorisasi untuk menulis data ke Amazon S3. Perintah UNLOAD menggunakan parameter yang sama dengan perintah COPY untuk otorisasi. Untuk informasi selengkapnya, lihat Parameter otorisasi referensi sintaks perintah COPY.

IAM_ROLE {default | 'arn:aws:iam: :rol/ '*<Akun AWS-id-1><role-name>*

Gunakan kata kunci default agar Amazon Redshift menggunakan peran IAM yang ditetapkan sebagai default dan terkait dengan cluster saat perintah UNLOAD dijalankan.

Gunakan Nama Sumber Daya Amazon (ARN) untuk peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Jika Anda menentukan IAM_ROLE, Anda tidak dapat menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY, SESSION_TOKEN, atau CREDENTIALS. IAM_ROLE dapat dirantai. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Panduan Manajemen Amazon Redshift.

[FORMAT [SEBAGAI]] CSV | PARKET | JSON

Kata kunci untuk menentukan format bongkar muat untuk mengganti format default.

Saat CSV, bongkar muat ke file teks dalam format CSV menggunakan karakter koma (,) sebagai pembatas default. Jika bidang berisi pembatas, tanda kutip ganda, karakter baris baru, atau pengembalian kereta, maka bidang dalam file yang tidak dimuat ditutup dalam tanda kutip ganda. Tanda kutip ganda dalam bidang data dilepaskan oleh tanda kutip ganda tambahan. Saat nol baris dibongkar, Amazon Redshift mungkin menulis objek Amazon S3 kosong.

Saat PARQUET, bongkar ke file dalam format Apache Parquet versi 1.0. Secara default, setiap grup baris dikompresi menggunakan kompresi SNAPPY. Untuk informasi selengkapnya tentang format Parket Apache, lihat Parket.

Saat JSON, membongkar ke file JSON dengan setiap baris berisi objek JSON, mewakili catatan lengkap dalam hasil kueri. Amazon Redshift mendukung penulisan JSON bersarang saat hasil kueri berisi kolom SUPER. Untuk membuat objek JSON yang valid, nama setiap kolom dalam kueri harus unik. Dalam file JSON, nilai boolean dibongkar sebagai `t` atau`f`, dan nilai NULL dibongkar sebagai. `null` Saat nol baris diturunkan, Amazon Redshift tidak menulis objek Amazon S3.

Kata kunci FORMAT dan AS adalah opsional. Anda tidak dapat menggunakan CSV dengan ESCAPE, FIXEDWIDTH, atau ADDQUOTES. Anda tidak dapat menggunakan PARQUET dengan DELIMITER, FIXEDWIDTH, ADDQUOTES, ESCAPE, NULL AS, HEADER, GZIP, BZIP2, atau ZSTD. PARQUET dengan ENCRYPTED hanya didukung dengan enkripsi sisi server dengan AWS Key Management Service kunci ()SSE-KMS. Anda tidak dapat menggunakan JSON dengan DELIMITER, HEADER, FIXEDWIDTH, ADDQUOTES, ESCAPE, atau NULL AS.

PARTISI OLEH (column_name [,...]) [TERMASUK]

Menentukan kunci partisi untuk operasi bongkar muat. UNLOAD secara otomatis mempartisi file output ke folder partisi berdasarkan nilai kunci partisi, mengikuti konvensi Apache Hive. Misalnya, file Parquet milik tahun partisi 2019 dan bulan September memiliki awalan berikut:`s3://amzn-s3-demo-bucket/my_prefix/year=2019/month=September/000.parquet`.

Nilai untuk column_name harus berupa kolom dalam hasil kueri yang sedang dibongkar.

Jika Anda menentukan PARTITION BY dengan opsi INCLUDE, kolom partisi tidak dihapus dari file yang dibongkar.

Amazon Redshift tidak mendukung literal string dalam klausa PARTITION BY.

MANIFES [VERBOSE]

Membuat file manifes yang secara eksplisit mencantumkan detail untuk file data yang dibuat oleh proses UNLOAD. Manifes adalah file teks dalam format JSON yang mencantumkan URL setiap file yang ditulis ke Amazon S3.

Jika MANIFEST ditentukan dengan opsi VERBOSE, manifes menyertakan detail berikut:

- Nama kolom dan tipe data, dan untuk tipe data CHAR, VARCHAR, atau NUMERIK, dimensi untuk setiap kolom. Untuk tipe data CHAR dan VARCHAR, dimensi adalah panjang. Untuk tipe data DECIMAL atau NUMERIC, dimensinya adalah presisi dan skala.

- Jumlah baris dibongkar ke setiap file. Jika opsi HEADER ditentukan, jumlah baris menyertakan baris header.
- Ukuran file total dari semua file yang dibongkar dan jumlah baris total yang dibongkar ke semua file. Jika opsi HEADER ditentukan, jumlah baris menyertakan baris header.
- Penulis. Penulis selalu “Amazon Redshift”.

Anda dapat menentukan VERBOSE hanya mengikuti MANIFEST.

File manifes ditulis ke awalan jalur Amazon S3 yang sama dengan file pembongkaran dalam format`<object_path_prefix>manifest`. Misalnya, jika UNLOAD menentukan awalan jalur Amazon S3 '`s3://amzn-s3-demo-bucket/venue_`', lokasi file manifes adalah '`s3://amzn-s3-demo-bucket/venue_manifest`'.

TAJUK

Menambahkan baris header yang berisi nama kolom di bagian atas setiap file output. Opsi transformasi teks, seperti CSV, DELIMITER, ADDQUOTES, dan ESCAPE, juga berlaku untuk baris header. Anda tidak dapat menggunakan HEADER dengan FIXEDWIDTH.

PEMBATAS SEBAGAI 'delimiter_character'

Menentukan karakter ASCII tunggal yang digunakan untuk memisahkan bidang dalam file output, seperti karakter pipa (|), koma (,), atau tab (\ t). Pembatas default untuk file teks adalah karakter pipa. Pembatas default untuk file CSV adalah karakter koma. Kata kunci AS adalah opsional. Anda tidak dapat menggunakan DELIMITER dengan FIXEDWIDTH. Jika data berisi karakter pembatas, Anda perlu menentukan opsi ESCAPE untuk menghindari pembatas, atau menggunakan ADDQUOTES untuk melampirkan data dalam tanda kutip ganda. Atau, tentukan pembatas yang tidak terkandung dalam data.

FIXEDWIDTH 'fixedwidth_spec'

Membongkar data ke file di mana setiap lebar kolom adalah panjang tetap, bukan dipisahkan oleh pembatas. fixedwidth_spec adalah string yang menentukan jumlah kolom dan lebar kolom. Kata kunci AS adalah opsional. Karena FIXEDWIDTH tidak memotong data, spesifikasi untuk setiap kolom dalam pernyataan UNLOAD harus setidaknya sepanjang entri terpanjang untuk kolom itu. Format untuk fixedwidth_ spec ditunjukkan di bawah ini:

```
'colID1:colWidth1,colID2:colWidth2, ...'
```

Anda tidak dapat menggunakan FIXEDWIDTH dengan DELIMITER atau HEADER.

TERENKRIPSI [OTOMATIS]

Menentukan bahwa file keluaran di Amazon S3 dienkripsi menggunakan enkripsi sisi server Amazon S3. Jika MANIFEST ditentukan, file manifes juga dienkripsi. Untuk informasi selengkapnya, lihat Membongkar file data terenkripsi. Jika Anda tidak menentukan parameter ENCRYPTED, UNLOAD akan secara otomatis membuat file terenkripsi menggunakan enkripsi sisi server Amazon S3 dengan kunci enkripsi yang AWS dikelola (). SSE-S3

Untuk ENCRYPTED, Anda mungkin ingin membongkar ke Amazon S3 menggunakan enkripsi sisi server dengan AWS KMS kunci (). SSE-KMS Jika demikian, gunakan KMS_KEY_ID parameter untuk memberikan ID kunci. Anda tidak dapat menggunakan the section called “Menggunakan KREDENSIAL” parameter dengan parameter KMS_KEY_ID. Jika Anda menjalankan perintah UNLOAD untuk data menggunakan KMS_KEY_ID, Anda kemudian dapat melakukan operasi COPY untuk data yang sama tanpa menentukan kunci.

Jika ENCRYPTED AUTO digunakan, perintah UNLOAD mengambil kunci AWS KMS enkripsi default pada properti bucket Amazon S3 target dan mengenkripsi file yang ditulis ke Amazon S3 dengan kunci tersebut. AWS KMS Jika bucket tidak memiliki kunci AWS KMS enkripsi default, UNLOAD secara otomatis membuat file terenkripsi menggunakan enkripsi sisi server Amazon Redshift dengan kunci enkripsi yang AWS dikelola (). SSE-S3 Anda tidak dapat menggunakan opsi ini dengan KMS_KEY_ID, MASTER_SYMMETRIC_KEY, atau CREDENTIALS yang berisi master_symmetric_key.

KMS_KEY_ID 'id kunci'

Menentukan ID kunci untuk kunci AWS Key Management Service (AWS KMS) yang akan digunakan untuk mengenkripsi file data di Amazon S3. Untuk informasi lebih lanjut, lihat Apa itu AWS Key Management Service? Jika Anda menentukan KMS_KEY_ID, Anda harus menentukan parameter juga. ENCRYPTED Jika Anda menentukan KMS_KEY_ID, Anda tidak dapat mengotentikasi menggunakan parameter CREDENTIALS. Sebagai gantinya, gunakan salah the section called “Menggunakan IAM_ROLE” satu atauthe section called “Menggunakan ACCESS_KEY_ID dan SECRET_ACCESS_KEY”.

BZIP2

Membongkar data ke satu atau lebih file terkompresi bzip2 per irisan. Setiap file yang dihasilkan ditambahkan dengan `.bz2` ekstensi.

GZIP

Membongkar data ke satu atau beberapa file terkompresi gzip per irisan. Setiap file yang dihasilkan ditambahkan dengan `.gz` ekstensi.

ZSTD

Membongkar data ke satu atau lebih Zstandard-compressed file per irisan. Setiap file yang dihasilkan ditambahkan dengan `.zst` ekstensi.

TAMBAHKAN KUTIPAN

Menempatkan tanda kutip di sekitar setiap bidang data yang tidak dimuat, sehingga Amazon Redshift dapat membongkar nilai data yang berisi pembatas itu sendiri. Misalnya, jika pembatas adalah koma, Anda dapat membongkar dan memuat ulang data berikut dengan sukses:

```
"1","Hello, World"
```

Tanpa tanda kutip tambahan, string `Hello, World` akan diurai sebagai dua bidang terpisah.

Beberapa format output tidak mendukung ADDQUOTES.

Jika Anda menggunakan ADDQUOTES, Anda harus menentukan REMOVEQUOTES di COPY jika Anda memuat ulang data.

NULL SEBAGAI 'null-string'

Menentukan string yang mewakili nilai null dalam membongkar file. Jika opsi ini digunakan, semua file output berisi string yang ditentukan sebagai pengganti nilai nol yang ditemukan dalam data yang dipilih. Jika opsi ini tidak ditentukan, nilai null dibongkar sebagai:

- Zero-length string untuk output yang dibatasi
- String spasi untuk keluaran lebar tetap

Jika string null ditentukan untuk pembongkaran lebar tetap dan lebar kolom keluaran kurang dari lebar string null, perilaku berikut terjadi:

- Bidang kosong adalah output untuk kolom non-karakter
- Kesalahan dilaporkan untuk kolom karakter

Tidak seperti tipe data lain di mana string yang ditentukan pengguna mewakili nilai nol, Amazon Redshift mengekspor kolom data SUPER menggunakan format JSON dan merepresentasikannya sebagai null seperti yang ditentukan oleh format JSON. Akibatnya, kolom data SUPER mengabaikan opsi NULL [AS] yang digunakan dalam perintah UNLOAD.

MELARIKAN DIRI

Untuk kolom CHAR dan VARCHAR dalam file unload yang dibatasi, karakter escape (\) ditempatkan sebelum setiap kemunculan karakter berikut:

- Umpan garis: \n
- Pengembalian kereta: \r
- Karakter pembatas yang ditentukan untuk data yang dibongkar.
- Karakter pelarian: \
- Karakter tanda kutip: " atau ' (jika ESCAPE dan ADDQUOTES ditentukan dalam perintah UNLOAD).

> **Important**
>
> Jika Anda memuat data Anda menggunakan COPY dengan opsi ESCAPE, Anda juga harus menentukan opsi ESCAPE dengan perintah UNLOAD Anda untuk menghasilkan file keluaran timbal balik. Demikian pula, jika Anda UNLOAD menggunakan opsi ESCAPE, Anda perlu menggunakan ESCAPE ketika Anda MENYALIN data yang sama.

IZINKAN PENULISAN

Secara default, UNLOAD gagal jika menemukan file yang mungkin akan diganti. Jika ALLOWOVERWRITE ditentukan, UNLOAD akan mengganti file yang ada, termasuk file manifes.

JALUR BERSIH

Opsi CLEANPATH menghapus file yang ada yang terletak di jalur Amazon S3 yang ditentukan dalam klausa TO sebelum membongkar file ke lokasi yang ditentukan.

Jika Anda menyertakan klausa PARTITION BY, file yang ada dihapus hanya dari folder partisi untuk menerima file baru yang dihasilkan oleh operasi UNLOAD.

Anda harus memiliki `s3:DeleteObject` izin di bucket Amazon S3. Untuk informasi, lihat Kebijakan dan Izin di Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. File yang Anda hapus dengan menggunakan opsi CLEANPATH dihapus secara permanen dan tidak dapat dipulihkan. Jika bucket Amazon S3 target mengaktifkan pembuatan versi, UNLOAD dengan opsi CLEANPATH tidak akan menghapus versi file sebelumnya.

Anda tidak dapat menentukan opsi CLEANPATH jika Anda menentukan opsi ALLOWOVERWRITE.

PARALEL

Secara default, UNLOAD menulis data secara paralel ke beberapa file, sesuai dengan jumlah irisan dalam cluster. Opsi default adalah ON atau TRUE. Jika PARALLEL OFF atau FALSE,

UNLOAD menulis ke satu atau lebih file data secara serial, diurutkan secara mutlak sesuai dengan klausa ORDER BY, jika digunakan. Ukuran maksimum untuk file data adalah 6,2 GB. Jadi, misalnya, jika Anda membongkar data 13,4 GB, UNLOAD membuat tiga file berikut.

```
s3://amzn-s3-demo-bucket/key000    6.2 GB
s3://amzn-s3-demo-bucket/key001    6.2 GB
s3://amzn-s3-demo-bucket/key002    1.0 GB
```

> **Note**
>
> Perintah UNLOAD dirancang untuk menggunakan pemrosesan paralel. Sebaiknya biarkan PARALLEL diaktifkan untuk sebagian besar kasus, terutama jika file digunakan untuk memuat tabel menggunakan perintah COPY.

MAXFILESIZE [AS] ukuran maksimal [MB | GB]

Menentukan ukuran maksimum file yang dibuat UNLOAD di Amazon S3. Tentukan nilai desimal antara 5 MB dan 6,2 GB. Kata kunci AS adalah opsional. Unit default adalah MB. Jika MAXFILESIZE tidak ditentukan, ukuran file maksimum default adalah 6,2 GB. Ukuran file manifes, jika digunakan, tidak terpengaruh oleh MAXFILESIZE.

ROWGROUPSIZE [AS] ukuran [MB | GB]

Menentukan ukuran kelompok baris. Memilih ukuran yang lebih besar dapat mengurangi jumlah grup baris, mengurangi jumlah komunikasi jaringan. Tentukan nilai integer antara 32 MB dan 128 MB. Kata kunci AS adalah opsional. Unit default adalah MB.

Jika ROWGROUPSIZE tidak ditentukan, ukuran defaultnya adalah 32 MB. Untuk menggunakan parameter ini, format penyimpanan harus Parquet dan tipe simpul harus ra3.4xlarge, ra3.16xlarge, rg.4xlarge, rg.12xlarge, atau dc2.8xlarge.

WILAYAH [AS] 'aws- wilayah'

Men Wilayah AWS entukan lokasi bucket Amazon S3 target. REGION diperlukan untuk UNLOAD ke bucket Amazon S3 yang tidak Wilayah AWS sama dengan database Amazon Redshift.

Nilai untuk aws_region harus cocok dengan Wilayah yang tercantum dalam tabel AWS wilayah Amazon Redshift dan titik akhir di. Referensi Umum AWS

Secara default, UNLOAD mengasumsikan bahwa bucket Amazon S3 target terletak Wilayah AWS sama dengan database Amazon Redshift.

EKSTENSI 'nama ekstensi'

Menentukan ekstensi file untuk ditambahkan ke nama file yang tidak dimuat. Amazon Redshift tidak menjalankan validasi apa pun, jadi Anda harus memverifikasi bahwa ekstensi file yang ditentukan sudah benar. Jika Anda menentukan metode kompresi tanpa memberikan ekstensi, Amazon Redshift hanya menambahkan ekstensi metode kompresi ke nama file. Jika Anda tidak memberikan ekstensi apa pun dan tidak menentukan metode kompresi, Amazon Redshift tidak menambahkan apa pun ke nama file.

Catatan penggunaan

Menggunakan ESCAPE untuk semua operasi UNLOAD teks yang dibatasi

Saat Anda MEMBONGKAR menggunakan pembatas, data Anda dapat menyertakan pembatas itu atau salah satu karakter yang tercantum dalam deskripsi opsi ESCAPE. Dalam hal ini, Anda harus menggunakan opsi ESCAPE dengan pernyataan UNLOAD. Jika Anda tidak menggunakan opsi ESCAPE dengan UNLOAD, operasi COPY berikutnya menggunakan data yang tidak dimuat mungkin gagal.

> **Important**
>
> Kami sangat menyarankan agar Anda selalu menggunakan ESCAPE dengan pernyataan UNLOAD dan COPY. Pengecualiannya adalah jika Anda yakin bahwa data Anda tidak mengandung pembatas atau karakter lain yang mungkin perlu dihindari.

Hilangnya presisi floating-point

Anda mungkin mengalami kehilangan presisi untuk data floating-point yang secara berturut-turut dibongkar dan dimuat ulang.

Klausul batas

Kueri SELECT tidak dapat menggunakan klausa LIMIT di SELECT luar. Misalnya, pernyataan UNLOAD berikut gagal.

```
unload ('select * from venue limit 10')
to 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole';
```

Sebagai gantinya, gunakan klausa LIMIT bersarang, seperti pada contoh berikut.

```
unload ('select * from venue where venueid in
(select venueid from venue order by venueid desc limit 10)')
to 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole';
```

Anda juga dapat mengisi tabel menggunakan SELECT... INTO atau CREATE TABLE AS menggunakan klausa LIMIT, lalu membongkar dari tabel tersebut.

Membongkar kolom tipe data GEOMETRY

Anda hanya dapat membongkar kolom GEOMETRY ke teks atau format CSV. Anda tidak dapat membongkar data GEOMETRY dengan `FIXEDWIDTH` opsi tersebut. Data dibongkar dalam bentuk heksadesimal dari format biner terkenal yang diperluas (EWKB). Jika ukuran data EWKB lebih dari 4 MB, maka peringatan terjadi karena data nantinya tidak dapat dimuat ke dalam tabel.

Membongkar tipe data HLLSKETCH

Anda hanya dapat membongkar kolom HLLSKETCH ke teks atau format CSV. Anda tidak dapat membongkar data HLLSKETCH dengan opsi. `FIXEDWIDTH` Data dibongkar dalam format Base64 untuk HyperLogLog sketsa padat atau dalam format JSON untuk sketsa jarang HyperLogLog . Untuk informasi selengkapnya, lihat HyperLogLog fungsi.

Contoh berikut mengekspor tabel yang berisi kolom HLLSKETCH ke dalam file.

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
 (5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

UNLOAD ('select * from hll_table') TO 's3://amzn-s3-demo-bucket/unload/'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole' NULL AS 'null' ALLOWOVERWRITE
 CSV;
```

Membongkar kolom tipe data VARBYTE

Anda hanya dapat membongkar kolom VARBYTE ke teks atau format CSV. Data dibongkar dalam bentuk heksadesimal. Anda tidak dapat membongkar data VARBYTE dengan opsi. `FIXEDWIDTH`

ADDQU0TESOpsi UNLOAD ke CSV tidak didukung. Kolom VARBYTE tidak dapat berupa kolom PARTITIONED BY.

FORMAT AS klausa PARQUET

Perhatikan pertimbangan berikut saat menggunakan FORMAT AS PARQUET:

- Membongkar ke Parquet tidak menggunakan kompresi tingkat file. Setiap grup baris dikompresi dengan SNAPPY.
- Jika MAXFILESIZE tidak ditentukan, ukuran file maksimum default adalah 6,2 GB. Anda dapat menggunakan MAXFILESIZE untuk menentukan ukuran file 5 MB—6,2 GB. Ukuran file sebenarnya diperkirakan saat file sedang ditulis, jadi mungkin tidak persis sama dengan angka yang Anda tentukan.

  Untuk memaksimalkan kinerja pemindaian, Amazon Redshift mencoba membuat file Parquet yang berisi grup baris 32-MB berukuran sama. Nilai MAXFILESIZE yang Anda tentukan secara otomatis dibulatkan ke bawah ke kelipatan terdekat dari 32 MB. Misalnya, jika Anda menentukan MAXFILESIZE 200 MB, maka setiap file Parquet yang dibongkar kira-kira 192 MB (grup baris 32 MB x 6 = 192 MB).
- Jika kolom menggunakan format data TIMESTAMPTZ, hanya nilai stempel waktu yang dibongkar. Informasi zona waktu tidak dibongkar.
- Jangan tentukan awalan nama file yang dimulai dengan karakter garis bawah (_) atau titik (.). Redshift Spectrum memperlakukan file yang dimulai dengan karakter ini sebagai file tersembunyi dan mengabaikannya.

PARTISI BY klausa

Waspadai pertimbangan ini saat menggunakan PARTITION BY:

- Kolom partisi tidak termasuk dalam file output.
- Pastikan untuk menyertakan kolom partisi dalam kueri SELECT yang digunakan dalam pernyataan UNLOAD. Anda dapat menentukan sejumlah kolom partisi dalam perintah UNLOAD. Namun, ada batasan bahwa setidaknya harus ada satu kolom nonpartisi untuk menjadi bagian dari file.
- Jika nilai kunci partisi adalah null, Amazon Redshift secara otomatis membongkar data tersebut ke partisi default yang disebut. `partition_column=__HIVE_DEFAULT_PARTITION__`
- Perintah UNLOAD tidak melakukan panggilan apa pun ke katalog eksternal. Untuk mendaftarkan partisi baru Anda untuk menjadi bagian dari tabel eksternal Anda yang ada, gunakan ALTER

TABLE terpisah... TAMBAH PARTISI... perintah. Atau Anda dapat menjalankan perintah CREATE EXTERNAL TABLE untuk mendaftarkan data yang dibongkar sebagai tabel eksternal baru. Anda juga dapat menggunakan AWS Glue crawler untuk mengisi Katalog Data Anda. Untuk informasi selengkapnya, lihat Mendefinisikan Crawler di Panduan Peng AWS Glue embang.

- Jika Anda menggunakan opsi MANIFEST, Amazon Redshift hanya menghasilkan satu file manifes di folder root Amazon S3.
- Jenis data kolom yang dapat Anda gunakan sebagai kunci partisi adalah SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, BOOLEAN, CHAR, VARCHAR, DATE, dan TIMESTAMP.

Menggunakan hak istimewa ASSUMEROLE untuk memberikan akses ke peran IAM untuk operasi UNLOAD

Untuk menyediakan akses bagi pengguna dan grup tertentu ke peran IAM untuk operasi UNLOAD, superuser dapat memberikan hak istimewa ASSUMEROLE pada peran IAM kepada pengguna dan grup. Untuk informasi, lihat HIBAH.

UNLOAD tidak mendukung alias titik akses Amazon S3

Anda tidak dapat menggunakan alias titik akses Amazon S3 dengan perintah UNLOAD.

Contoh

Untuk contoh yang menunjukkan cara menggunakan perintah UNLOAD, lihatContoh UNLOAD.

Contoh UNLOAD

Contoh-contoh ini menunjukkan berbagai parameter perintah UNLOAD. Data sampel TICKIT digunakan dalam banyak contoh. Untuk informasi selengkapnya, lihat Database sampel.

> **Note**
>
> Contoh-contoh ini berisi jeda baris untuk keterbacaan. Jangan sertakan jeda baris atau spasi dalam string kredensi als-args Anda.

Bongkar VENUE ke file yang dibatasi pipa (pembatas default)

Contoh berikut membongkar tabel VENUE dan menulis data ke: `s3://amzn-s3-demo-bucket/unload/`

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Secara default, UNLOAD menulis satu atau lebih file per irisan. Dengan asumsi cluster dua simpul dengan dua irisan per node, contoh sebelumnya membuat file-file ini di: `amzn-s3-demo-bucket`

```
unload/0000_part_00
unload/0001_part_00
unload/0002_part_00
unload/0003_part_00
```

Untuk membedakan file output dengan lebih baik, Anda dapat menyertakan awalan di lokasi. Contoh berikut membongkar tabel VENUE dan menulis data ke: `s3://amzn-s3-demo-bucket/unload/venue_pipe_`

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Hasilnya adalah keempat file ini di `unload` folder, sekali lagi dengan asumsi empat irisan.

```
venue_pipe_0000_part_00
venue_pipe_0001_part_00
venue_pipe_0002_part_00
venue_pipe_0003_part_00
```

Bongkar tabel LINEITEM ke file Parquet yang dipartisi

Contoh berikut membongkar tabel LINEITEM dalam format Parquet, dipartisi oleh kolom. `l_shipdate`

```
unload ('select * from lineitem')
to 's3://amzn-s3-demo-bucket/lineitem/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
PARQUET
PARTITION BY (l_shipdate);
```

Dengan asumsi empat irisan, file Parquet yang dihasilkan secara dinamis dipartisi ke dalam berbagai folder.

```
s3://amzn-s3-demo-bucket/lineitem/l_shipdate=1992-01-02/0000_part_00.parquet
                                                0001_part_00.parquet
                                                0002_part_00.parquet
                                                0003_part_00.parquet
s3://amzn-s3-demo-bucket/lineitem/l_shipdate=1992-01-03/0000_part_00.parquet
                                                0001_part_00.parquet
                                                0002_part_00.parquet
                                                0003_part_00.parquet
s3://amzn-s3-demo-bucket/lineitem/l_shipdate=1992-01-04/0000_part_00.parquet
                                                0001_part_00.parquet
                                                0002_part_00.parquet
                                                0003_part_00.parquet
...
```

Note

Dalam beberapa kasus, perintah UNLOAD menggunakan opsi INCLUDE seperti yang ditunjukkan dalam pernyataan SQL berikut.

```
unload ('select * from lineitem')
to 's3://amzn-s3-demo-bucket/lineitem/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
PARQUET
PARTITION BY (l_shipdate) INCLUDE;
```

Dalam kasus ini, `l_shipdate` kolom juga ada dalam data dalam file Parquet. Jika tidak, data `l_shipdate` kolom tidak ada dalam file Parquet.

Bongkar tabel VENUE ke file JSON

Contoh berikut membongkar tabel VENUE dan menulis data dalam format JSON kes`3://amzn-s3-demo-bucket/unload/`.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
JSON;
```

Berikut ini adalah contoh baris dari tabel VENUE.

```
venueid | venuename                  | venuecity       | venuestate | venueseats
--------+----------------------------+-----------------+------------+-----------
      1 | Pinewood Racetrack         | Akron           | OH         | 0
      2 | Columbus "Crew" Stadium    | Columbus        | OH         | 0
      4 | Community, Ballpark, Arena | Kansas City     | KS         | 0
```

Setelah dibongkar ke JSON, format file mirip dengan berikut ini.

```
{"venueid":1,"venuename":"Pinewood
 Racetrack","venuecity":"Akron","venuestate":"OH","venueseats":0}
{"venueid":2,"venuename":"Columbus \"Crew\" Stadium
 ","venuecity":"Columbus","venuestate":"OH","venueseats":0}
{"venueid":4,"venuename":"Community, Ballpark, Arena","venuecity":"Kansas
 City","venuestate":"KS","venueseats":0}
```

Bongkar VENUE ke file CSV

Contoh berikut membongkar tabel VENUE dan menulis data dalam format CSV kes3://amzn-s3-demo-bucket/unload/.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
CSV;
```

Misalkan tabel VENUE berisi baris berikut.

```

venueid | venuename                  | venuecity       | venuestate | venueseats
--------+----------------------------+-----------------+------------+-----------
      1 | Pinewood Racetrack         | Akron           | OH         | 0
      2 | Columbus "Crew" Stadium    | Columbus        | OH         | 0
      4 | Community, Ballpark, Arena | Kansas City     | KS         | 0

```

File pembongkaran terlihat mirip dengan berikut ini.

```
1,Pinewood Racetrack,Akron,OH,0
2,"Columbus ""Crew"" Stadium",Columbus,OH,0
```

```
4,"Community, Ballpark, Arena",Kansas City,KS,0
```

Bongkar VENUE ke file CSV menggunakan pembatas

Contoh berikut membongkar tabel VENUE dan menulis data dalam format CSV menggunakan karakter pipa (|) sebagai pembatas. File yang dibongkar ditulis kes3://amzn-s3-demo-bucket/unload/. Tabel VENUE dalam contoh ini berisi karakter pipa dalam nilai baris pertama (Pinewood Race|track). Ini dilakukan untuk menunjukkan bahwa nilai dalam hasil tertutup dalam tanda kutip ganda. Tanda kutip ganda dilepaskan oleh tanda kutip ganda, dan seluruh bidang tertutup dalam tanda kutip ganda.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
CSV DELIMITER AS '|';
```

Misalkan tabel VENUE berisi baris berikut.

```
venueid | venuename                  | venuecity       | venuestate | venueseats
--------+----------------------------+-----------------+------------+-------------
      1 | Pinewood Race|track        | Akron           | OH         | 0
      2 | Columbus "Crew" Stadium    | Columbus        | OH         | 0
      4 | Community, Ballpark, Arena | Kansas City     | KS         | 0
```

File pembongkaran terlihat mirip dengan berikut ini.

```
1|"Pinewood Race|track"|Akron|OH|0
2|"Columbus ""Crew"" Stadium"|Columbus|OH|0
4|Community, Ballpark, Arena|Kansas City|KS|0
```

Bongkar VENUE dengan file manifes

Untuk membuat file manifes, sertakan opsi MANIFEST. Contoh berikut membongkar tabel VENUE dan menulis file manifes bersama dengan file data ke s3://amzn-s3-demo -: bucket/venue_pipe_

> **Important**
>
> Jika Anda membongkar file dengan opsi MANIFEST, Anda harus menggunakan opsi MANIFEST dengan perintah COPY saat Anda memuat file. Jika Anda menggunakan awalan

yang sama untuk memuat file dan tidak menentukan opsi MANIFEST, COPY gagal karena mengasumsikan file manifes adalah file data.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole'
manifest;
```

Hasilnya adalah lima file ini:

```
s3://amzn-s3-demo-bucket/venue_pipe_0000_part_00
s3://amzn-s3-demo-bucket/venue_pipe_0001_part_00
s3://amzn-s3-demo-bucket/venue_pipe_0002_part_00
s3://amzn-s3-demo-bucket/venue_pipe_0003_part_00
s3://amzn-s3-demo-bucket/venue_pipe_manifest
```

Berikut ini menunjukkan isi file manifes.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0000_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0001_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0002_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/tickit/venue_0003_part_00"}
  ]
}
```

Bongkar VENUE dengan MANIFEST VERBOSE

Saat Anda menentukan opsi MANIFEST VERBOSE, file manifes mencakup bagian berikut:

- `entries`Bagian ini mencantumkan jalur Amazon S3, ukuran file, dan jumlah baris untuk setiap file.
- `schema`Bagian ini mencantumkan nama kolom, tipe data, dan dimensi untuk setiap kolom.
- `meta`Bagian ini menunjukkan ukuran file total dan jumlah baris untuk semua file.

Contoh berikut membongkar tabel VENUE menggunakan opsi MANIFEST VERBOSE.

```
unload ('select * from venue')
```

```
to 's3://amzn-s3-demo-bucket/unload_venue_folder/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest verbose;
```

Berikut ini menunjukkan isi file manifes.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/venue_pipe_0000_part_00", "meta":
 { "content_length": 32295, "record_count": 10 }},
    {"url":"s3://amzn-s3-demo-bucket/venue_pipe_0001_part_00", "meta":
 { "content_length": 32771, "record_count": 20 }},
    {"url":"s3://amzn-s3-demo-bucket/venue_pipe_0002_part_00", "meta":
 { "content_length": 32302, "record_count": 10 }},
    {"url":"s3://amzn-s3-demo-bucket/venue_pipe_0003_part_00", "meta":
 { "content_length": 31810, "record_count": 15 }}
  ],
  "schema": {
    "elements": [
      {"name": "venueid", "type": { "base": "integer" }},
      {"name": "venuename", "type": { "base": "character varying", 25 }},
      {"name": "venuecity", "type": { "base": "character varying", 25 }},
      {"name": "venuestate", "type": { "base": "character varying", 25 }},
      {"name": "venueseats", "type": { "base": "character varying", 25 }}
    ]
  },
  "meta": {
    "content_length": 129178,
    "record_count": 55
  },
  "author": {
    "name": "Amazon Redshift",
    "version": "1.0.0"
  }
}
```

Bongkar VENUE dengan header

Contoh berikut membongkar VENUE dengan baris header.

```
unload ('select * from venue where venueseats > 75000')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
header
parallel off;
```

Berikut ini menunjukkan isi file output dengan baris header.

```
venueid|venuename|venuecity|venuestate|venueseats
6|New York Giants Stadium|East Rutherford|NJ|80242
78|INVESCO Field|Denver|CO|76125
83|FedExField|Landover|MD|91704
79|Arrowhead Stadium|Kansas City|MO|79451
```

Bongkar VENUE ke file yang lebih kecil

Secara default, ukuran file maksimum adalah 6,2 GB. Jika data bongkar lebih besar dari 6,2 GB, UNLOAD membuat file baru untuk setiap segmen data 6,2 GB. Untuk membuat file yang lebih kecil, sertakan parameter MAXFILESIZE. Dengan asumsi ukuran data dalam contoh sebelumnya adalah 20 GB, perintah UNLOAD berikut membuat 20 file, masing-masing berukuran 1 GB.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
maxfilesize 1 gb;
```

Bongkar VENUE secara serial

Untuk membongkar secara serial, tentukan PARALLEL OFF. UNLOAD kemudian menulis satu file pada satu waktu, hingga maksimum 6,2 GB per file.

Contoh berikut membongkar tabel VENUE dan menulis data secara serial ke. `s3://amzn-s3-demo-bucket/unload/`

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/venue_serial_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off;
```

Hasilnya adalah satu file bernama venue_serial_000.

Jika data bongkar lebih besar dari 6,2 GB, UNLOAD membuat file baru untuk setiap segmen data 6,2 GB. Contoh berikut membongkar tabel LINEORDER dan menulis data secara serial ke. `s3://amzn-s3-demo-bucket/unload/`

```
unload ('select * from lineorder')
to 's3://amzn-s3-demo-bucket/unload/lineorder_serial_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off gzip;
```

Hasilnya adalah rangkaian file berikut.

```
lineorder_serial_0000.gz
lineorder_serial_0001.gz
lineorder_serial_0002.gz
lineorder_serial_0003.gz
```

Untuk membedakan file output dengan lebih baik, Anda dapat menyertakan awalan di lokasi. Contoh berikut membongkar tabel VENUE dan menulis data ke: `s3://amzn-s3-demo-bucket/venue_pipe_`

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Hasilnya adalah keempat file ini di `unload` folder, sekali lagi dengan asumsi empat irisan.

```
venue_pipe_0000_part_00
venue_pipe_0001_part_00
venue_pipe_0002_part_00
venue_pipe_0003_part_00
```

Muat VENUE dari membongkar file

Untuk memuat tabel dari sekumpulan file unload, cukup balikkan proses dengan menggunakan perintah COPY. Contoh berikut membuat tabel baru, LOADVENUE, dan memuat tabel dari file data yang dibuat dalam contoh sebelumnya.

```
create table loadvenue (like venue);

copy loadvenue from 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role
 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Jika Anda menggunakan opsi MANIFEST untuk membuat file manifes dengan file bongkar muat Anda, Anda dapat memuat data menggunakan file manifes yang sama. Anda melakukannya dengan perintah COPY dengan opsi MANIFEST. Contoh berikut memuat data menggunakan file manifes.

```
copy loadvenue
from 's3://amzn-s3-demo-bucket/venue_pipe_manifest' iam_role
 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

Bongkar VENUE ke file terenkripsi

Contoh berikut membongkar tabel VENUE ke satu set file terenkripsi menggunakan AWS KMS kunci. Jika Anda menentukan file manifes dengan opsi ENCRYPTED, file manifes juga dienkripsi. Untuk informasi selengkapnya, lihat Membongkar file data terenkripsi.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_encrypt_kms'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
kms_key_id '1234abcd-12ab-34cd-56ef-1234567890ab'
manifest
encrypted;
```

Contoh berikut membongkar tabel VENUE ke satu set file terenkripsi menggunakan kunci simetris root.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_encrypt_cmk'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key 'EXAMPLEMASTERKEYtkbjk/OpCwtYSx/M4/t7DMCDIK722'
encrypted;
```

Muat VENUE dari file terenkripsi

Untuk memuat tabel dari sekumpulan file yang dibuat dengan menggunakan UNLOAD dengan opsi ENCRYPT, balikkan proses dengan menggunakan perintah COPY. Dengan perintah itu, gunakan opsi ENCRYPTED dan tentukan kunci simetris root yang sama yang digunakan untuk perintah UNLOAD. Contoh berikut memuat tabel LOADVENUE dari file data terenkripsi yang dibuat dalam contoh sebelumnya.

```
create table loadvenue (like venue);
```

```
copy loadvenue
from 's3://amzn-s3-demo-bucket/venue_encrypt_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key 'EXAMPLEMASTERKEYtkbjk/OpCwtYSx/M4/t7DMCDIK722'
manifest
encrypted;
```

Bongkar data VENUE ke file yang dibatasi tab

```
unload ('select venueid, venuename, venueseats from venue')
to 's3://amzn-s3-demo-bucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t';
```

File data output terlihat seperti ini:

```
1 Toyota Park Bridgeview IL 0
2 Columbus Crew Stadium Columbus OH 0
3 RFK Stadium Washington DC 0
4 CommunityAmerica Ballpark Kansas City KS 0
5 Gillette Stadium Foxborough MA 68756
...
```

Bongkar VENUE ke file data lebar tetap

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_fw_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth as 'venueid:3,venuename:39,venuecity:16,venuestate:2,venueseats:6';
```

File data keluaran terlihat seperti berikut ini.

```
1  Toyota Park              Bridgeview  IL0
2  Columbus Crew Stadium    Columbus    OH0
3  RFK Stadium              Washington  DC0
4  CommunityAmerica BallparkKansas City KS0
5  Gillette Stadium         Foxborough  MA68756
...
```

Bongkar VENUE ke satu set file yang dibatasi tab GZIP-compressed

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t'
gzip;
```

Bongkar VENUE ke file GZIP-compressed teks

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
extension 'txt.gz'
gzip;
```

Bongkar data yang berisi pembatas

Contoh ini menggunakan opsi ADDQUOTES untuk membongkar data yang dipisahkan koma di mana beberapa bidang data aktual berisi koma.

Pertama, buat tabel yang berisi tanda kutip.

```
create table location (id int, location char(64));

insert into location values (1,'Phoenix, AZ'),(2,'San Diego, CA'),(3,'Chicago, IL');
```

Kemudian, bongkar data menggunakan opsi ADDQUOTES.

```
unload ('select id, location from location')
to 's3://amzn-s3-demo-bucket/location_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter ',' addquotes;
```

File data yang dibongkar terlihat seperti ini:

```
1,"Phoenix, AZ"
2,"San Diego, CA"
3,"Chicago, IL"
...
```

Membongkar hasil kueri gabungan

Contoh berikut membongkar hasil kueri gabungan yang berisi fungsi jendela.

```
unload ('select venuecity, venuestate, caldate, pricepaid,
sum(pricepaid) over(partition by venuecity, venuestate
order by caldate rows between 3 preceding and 3 following) as winsum
from sales join date on sales.dateid=date.dateid
join event on event.eventid=sales.eventid
join venue on event.venueid=venue.venueid
order by 1,2')
to 's3://amzn-s3-demo-bucket/tickit/winsum'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

File output terlihat seperti ini:

```
Atlanta|GA|2008-01-04|363.00|1362.00
Atlanta|GA|2008-01-05|233.00|2030.00
Atlanta|GA|2008-01-06|310.00|3135.00
Atlanta|GA|2008-01-08|166.00|8338.00
Atlanta|GA|2008-01-11|268.00|7630.00
...
```

Bongkar menggunakan NULL AS

UNLOAD mengeluarkan nilai nol sebagai string kosong secara default. Contoh berikut menunjukkan cara menggunakan NULL AS untuk mengganti string teks dengan nol.

Untuk contoh-contoh ini, tambahkan beberapa nilai null ke tabel VENUE.

```
update venue set venuestate = NULL
where venuecity = 'Cleveland';
```

Pilih dari VENUE di mana VENUESTATE adalah null untuk memverifikasi bahwa kolom berisi NULL.

```
select * from venue where venuestate is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
      22 | Quicken Loans Arena      | Cleveland |            |          0
     101 | Progressive Field        | Cleveland |            |      43345
```

```
      72 | Cleveland Browns Stadium | Cleveland |            |      73200
```

Sekarang, BONGKAR tabel VENUE menggunakan opsi NULL AS untuk mengganti nilai null dengan string karakter '`fred`'.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
null as 'fred';
```

Contoh berikut dari file unload menunjukkan bahwa nilai null diganti dengan`fred`. Ternyata beberapa nilai untuk VENUESEATS juga nol dan diganti dengan. `fred` Meskipun tipe data untuk VENUESEATS adalah integer, UNLOAD mengonversi nilai menjadi teks dalam file unload, dan kemudian COPY mengubahnya kembali ke integer. Jika Anda membongkar ke file dengan lebar tetap, string NULL AS tidak boleh lebih besar dari lebar bidang.

```
248|Charles Playhouse|Boston|MA|0
251|Paris Hotel|Las Vegas|NV|fred
258|Tropicana Hotel|Las Vegas|NV|fred
300|Kennedy Center Opera House|Washington|DC|0
306|Lyric Opera House|Baltimore|MD|0
308|Metropolitan Opera|New York City|NY|0
  5|Gillette Stadium|Foxborough|MA|5
 22|Quicken Loans Arena|Cleveland|fred|0
101|Progressive Field|Cleveland|fred|43345
...
```

Untuk memuat tabel dari file unload, gunakan perintah COPY dengan opsi NULL AS yang sama.

> **Note**
>
> Jika Anda mencoba memuat nol ke kolom yang didefinisikan sebagai NOT NULL, perintah COPY gagal.

```
create table loadvenuenulls (like venue);

copy loadvenuenulls from 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
null as 'fred';
```

Untuk memverifikasi bahwa kolom berisi null, bukan hanya string kosong, pilih dari LOADVENUENULLS dan filter untuk null.

```
select * from loadvenuenulls where venuestate is null or venueseats is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
      72 | Cleveland Browns Stadium | Cleveland |            |      73200
     253 | Mirage Hotel             | Las Vegas | NV         |
     255 | Venetian Hotel           | Las Vegas | NV         |
      22 | Quicken Loans Arena      | Cleveland |            |          0
     101 | Progressive Field        | Cleveland |            |      43345
     251 | Paris Hotel              | Las Vegas | NV         |

...
```

Anda dapat MEMBONGKAR tabel yang berisi nol menggunakan perilaku NULL AS default dan kemudian SALIN data kembali ke tabel menggunakan perilaku NULL AS default; namun, setiap bidang non-numerik dalam tabel target berisi string kosong, bukan nol. Secara default UNLOAD mengonversi nol menjadi string kosong (spasi putih atau panjang nol). COPY mengubah string kosong menjadi NULL untuk kolom numerik, tetapi menyisipkan string kosong ke kolom non-numerik. Contoh berikut menunjukkan cara melakukan UNLOAD diikuti oleh COPY menggunakan perilaku NULL AS default.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' allowoverwrite;

truncate loadvenuenulls;
copy loadvenuenulls from 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

Dalam hal ini, saat Anda memfilter nol, hanya baris di mana VENUESEATS berisi nol. Di mana VENUESTATE berisi nol dalam tabel (VENUE), VENUESTATE di tabel target (LOADVENUENULLS) berisi string kosong.

```
select * from loadvenuenulls where venuestate is null or venueseats is null;
```

```
 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
     253 | Mirage Hotel             | Las Vegas | NV         |
     255 | Venetian Hotel           | Las Vegas | NV         |
     251 | Paris Hotel              | Las Vegas | NV         |
...
```

Untuk memuat string kosong ke kolom non-numerik sebagai NULL, sertakan opsi EMPTYASNULL atau BLANKSASNULL. Tidak apa-apa untuk menggunakan keduanya.

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' allowoverwrite;

truncate loadvenuenulls;
copy loadvenuenulls from 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' EMPTYASNULL;
```

Untuk memverifikasi bahwa kolom berisi NULL, bukan hanya spasi atau string kosong, pilih dari LOADVENUENULLS dan filter untuk null.

```
select * from loadvenuenulls where venuestate is null or venueseats is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
      72 | Cleveland Browns Stadium | Cleveland |            |      73200
     253 | Mirage Hotel             | Las Vegas | NV         |
     255 | Venetian Hotel           | Las Vegas | NV         |
      22 | Quicken Loans Arena      | Cleveland |            |          0
     101 | Progressive Field        | Cleveland |            |      43345
     251 | Paris Hotel              | Las Vegas | NV         |
     ...
```

Bongkar menggunakan parameter ALLOWOVERWRITE

Secara default, UNLOAD tidak mengganti file yang ada di bucket tujuan. Misalnya, jika Anda menjalankan pernyataan UNLOAD yang sama dua kali tanpa memodifikasi file di bucket tujuan, UNLOAD kedua gagal. Untuk mengganti file yang ada, termasuk file manifes, tentukan opsi ALLOWOVERWRITE.

```
unload ('select * from venue')
```

```
to 's3://amzn-s3-demo-bucket/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest allowoverwrite;
```

Bongkar tabel EVENT menggunakan parameter PARALLEL dan MANIFEST

Anda dapat MEMBONGKAR tabel secara paralel dan menghasilkan file manifes. File data Amazon S3 semuanya dibuat pada tingkat yang sama dan nama diberi akhiran dengan pola. `0000_part_00` File manifes berada pada tingkat folder yang sama dengan file data dan diberi akhiran dengan teks. `manifest` SQL berikut membongkar tabel EVENT dan membuat file dengan nama dasar `parallel`

```
unload ('select * from mytickit1.event')
to 's3://amzn-s3-demo-bucket/parallel'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
parallel on
manifest;
```

Daftar file Amazon S3 mirip dengan yang berikut ini.

```
 Name                        Last modified                          Size
 parallel0000_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 52.1 KB
 parallel0001_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 53.4 KB
 parallel0002_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 52.1 KB
 parallel0003_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 51.1 KB
 parallel0004_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 54.6 KB
 parallel0005_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 53.4 KB
 parallel0006_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 54.1 KB
 parallel0007_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 55.9 KB
 parallelmanifest       -   August 2, 2023, 14:54:39 (UTC-07:00) 886.0 B
```

Kon `parallelmanifest` ten file mirip dengan berikut ini.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/parallel0000_part_00", "meta": { "content_length":
 53316 }},
    {"url":"s3://amzn-s3-demo-bucket/parallel0001_part_00", "meta": { "content_length":
 54704 }},
```

```
    {"url":"s3://amzn-s3-demo-bucket/parallel0002_part_00", "meta": { "content_length":
 53326 }},
    {"url":"s3://amzn-s3-demo-bucket/parallel0003_part_00", "meta": { "content_length":
 52356 }},
    {"url":"s3://amzn-s3-demo-bucket/parallel0004_part_00", "meta": { "content_length":
 55933 }},
    {"url":"s3://amzn-s3-demo-bucket/parallel0005_part_00", "meta": { "content_length":
 54648 }},
    {"url":"s3://amzn-s3-demo-bucket/parallel0006_part_00", "meta": { "content_length":
 55436 }},
    {"url":"s3://amzn-s3-demo-bucket/parallel0007_part_00", "meta": { "content_length":
 57272 }}
  ]
}
```

Bongkar tabel EVENT menggunakan parameter PARALLEL OFF dan MANIFEST

Anda dapat MEMBONGKAR tabel secara serial (PARALLEL OFF) dan menghasilkan file manifes. File data Amazon S3 semuanya dibuat pada tingkat yang sama dan nama diberi akhiran dengan pola. `0000` File manifes berada pada tingkat folder yang sama dengan file data dan diberi akhiran dengan teks. `manifest`

```
unload ('select * from mytickit1.event')
to 's3://amzn-s3-demo-bucket/serial'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
parallel off
manifest;
```

Daftar file Amazon S3 mirip dengan yang berikut ini.

```
 Name                    Last modified                         Size
 serial0000          -   August 2, 2023, 15:54:39 (UTC-07:00) 426.7 KB
 serialmanifest      -   August 2, 2023, 15:54:39 (UTC-07:00) 120.0 B
```

Kon `serialmanifest` ten file mirip dengan berikut ini.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/serial000", "meta": { "content_length": 436991 }}
```

```
  ]
}
```

Bongkar tabel EVENT menggunakan parameter PARTITION BY dan MANIFEST

Anda dapat MEMBONGKAR tabel dengan partisi dan menghasilkan file manifes. Folder baru dibuat di Amazon S3 dengan folder partisi anak, dan file data di folder anak dengan pola nama yang mirip dengan`0000_par_00`. File manifes berada pada tingkat folder yang sama dengan folder anak dengan nama`manifest`.

```
unload ('select * from mytickit1.event')
to 's3://amzn-s3-demo-bucket/partition'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
partition by (eventname)
manifest;
```

Daftar file Amazon S3 mirip dengan yang berikut ini.

```
Name                    Type     Last modified                     Size

partition            Folder
```

Di `partition` folder tersebut adalah folder anak dengan nama partisi dan file manifes. Berikut ini adalah bagian bawah daftar folder dalam `partition` folder, mirip dengan berikut ini.

```
Name                    Type      Last modified                      Size

...
eventname=Zucchero/    Folder
eventname=Zumanity/    Folder
eventname=ZZ Top/      Folder
manifest               -     August 2, 2023, 15:54:39 (UTC-07:00) 467.6 KB
```

Di `eventname=Zucchero/` folder adalah file data yang mirip dengan berikut ini.

```
Name               Last modified                      Size
0000_part_00 -   August 2, 2023, 15:59:19 (UTC-07:00) 70.0 B
```

```
 0001_part_00 -   August 2, 2023, 15:59:16 (UTC-07:00) 106.0 B
 0002_part_00 -   August 2, 2023, 15:59:15 (UTC-07:00) 70.0 B
 0004_part_00 -   August 2, 2023, 15:59:17 (UTC-07:00) 141.0 B
 0006_part_00 -   August 2, 2023, 15:59:16 (UTC-07:00) 35.0 B
 0007_part_00 -   August 2, 2023, 15:59:19 (UTC-07:00) 108.0 B
```

Bagian bawah konten `manifest` file mirip dengan berikut ini.

```
{
  "entries": [
    ...
    {"url":"s3://amzn-s3-demo-bucket/partition/eventname=Zucchero/007_part_00", "meta":
 { "content_length": 108 }},
    {"url":"s3://amzn-s3-demo-bucket/partition/eventname=Zumanity/007_part_00", "meta":
 { "content_length": 72 }}
  ]
}
```

Bongkar tabel EVENT menggunakan parameter MAXFILESIZE, ROWGROUPSIZE, dan MANIFEST

Anda dapat MEMBONGKAR tabel secara paralel dan menghasilkan file manifes. File data Amazon S3 semuanya dibuat pada tingkat yang sama dan nama diberi akhiran dengan pola. `0000_part_00` File data Parquet yang dihasilkan dibatasi hingga 256 MB dan ukuran grup baris 128 MB. File manifes berada pada tingkat folder yang sama dengan file data dan diberi akhiran dengan. `manifest`

```
unload ('select * from mytickit1.event')
to 's3://amzn-s3-demo-bucket/eventsize'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
maxfilesize 256 MB
rowgroupsize 128 MB
parallel on
parquet
manifest;
```

Daftar file Amazon S3 mirip dengan yang berikut ini.

```
 Name                            Type      Last modified                     Size
 eventsize0000_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.5 KB
```

```
 eventsize0001_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.8 KB
 eventsize0002_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.4 KB
 eventsize0003_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.0 KB
 eventsize0004_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 25.3 KB
 eventsize0005_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.8 KB
 eventsize0006_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 25.0 KB
 eventsize0007_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 25.6 KB
 eventsizemanifest                 -       August 2, 2023, 17:35:21 (UTC-07:00) 958.0 B
```

Kon `eventsizemanifest` ten file mirip dengan berikut ini.

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/eventsize0000_part_00.parquet", "meta":
 { "content_length": 25130 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0001_part_00.parquet", "meta":
 { "content_length": 25428 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0002_part_00.parquet", "meta":
 { "content_length": 25025 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0003_part_00.parquet", "meta":
 { "content_length": 24554 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0004_part_00.parquet", "meta":
 { "content_length": 25918 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0005_part_00.parquet", "meta":
 { "content_length": 25362 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0006_part_00.parquet", "meta":
 { "content_length": 25647 }},
    {"url":"s3://amzn-s3-demo-bucket/eventsize0007_part_00.parquet", "meta":
 { "content_length": 26256 }}
  ]
}
```

UPDATE

Topik

- Sintaksis
- Parameter
- Catatan penggunaan
- Contoh pernyataan UPDATE

Memperbarui nilai dalam satu atau lebih kolom tabel ketika suatu kondisi terpenuhi.

> **Note**
>
> Ukuran maksimum untuk satu pernyataan SQL adalah 16 MB.

Sintaksis

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
            UPDATE table_name [ [ AS ] alias ] SET column = { expression | DEFAULT }
 [,...]

[ FROM fromlist ]
[ WHERE condition ]
```

Parameter

DENGAN klausa

Klausa opsional yang menentukan satu atau lebih ekspresi tab el-umum. Lihat DENGAN klausa.

table_name

Tabel sementara atau persisten. Hanya pemilik tabel atau pengguna dengan hak istimewa UPDATE pada tabel yang dapat memperbarui baris. Jika Anda menggunakan klausa FROM atau memilih dari tabel dalam ekspresi atau kondisi, Anda harus memiliki hak istimewa SELECT pada tabel tersebut. Anda tidak dapat memberikan tabel alias di sini; namun, Anda dapat menentukan alias dalam klausa FROM.

> **Note**
>
> Tabel eksternal Amazon Redshift Spectrum bersifat read-only. Anda tidak dapat MEMPERBARUI tabel eksternal.

alias

Nama alternatif sementara untuk tabel target. Alias bersifat opsional. Kata kunci AS selalu opsional.

SET kolom =

Satu atau lebih kolom yang ingin Anda modifikasi. Kolom yang tidak terdaftar mempertahankan nilainya saat ini. Jangan sertakan nama tabel dalam spesifikasi kolom target. Misalnya, `UPDATE tab SET tab.col = 1` tidak valid.

ekspresi

Sebuah ekspresi yang mendefinisikan nilai baru untuk kolom yang ditentukan.

DEFAULT

Memperbarui kolom dengan nilai default yang ditetapkan ke kolom dalam pernyataan CREATE TABLE.

DARI daftar tabel

Anda dapat memperbarui tabel dengan mereferensikan informasi di tabel lain. Buat daftar tabel lain ini dalam klausa FROM atau gunakan subquery sebagai bagian dari kondisi WHERE. Tabel yang tercantum dalam klausa FROM dapat memiliki alias. Jika Anda perlu menyertakan tabel target pernyataan UPDATE dalam daftar, gunakan alias.

Kondisi WHERE

Klausa opsional yang membatasi pembaruan ke baris yang cocok dengan suatu kondisi. Ketika kondisi kembali`true`, kolom SET yang ditentukan diperbarui. Kondisi tersebut dapat berupa predikat sederhana pada kolom atau kondisi berdasarkan hasil subquery.

Anda dapat memberi nama tabel apa pun di subquery, termasuk tabel target untuk UPDATE.

Catatan penggunaan

Setelah memperbarui sejumlah besar baris dalam tabel:

- Vakum meja untuk merebut kembali ruang penyimpanan dan mengurutkan ulang baris.
- Analisis tabel untuk memperbarui statistik untuk perencana kueri.

Gabungan luar kiri, kanan, dan penuh tidak didukung dalam klausa FROM dari pernyataan UPDATE; mereka mengembalikan kesalahan berikut:

```
ERROR: Target table must be part of an equijoin predicate
```

Jika Anda perlu menentukan gabungan luar, gunakan subquery dalam klausa WHERE dari pernyataan UPDATE.

Jika pernyataan UPDATE Anda memerlukan self-join ke tabel target, Anda perlu menentukan kondisi gabungan, serta kriteria klausa WHERE yang memenuhi syarat baris untuk operasi pembaruan. Secara umum, ketika tabel target bergabung dengan dirinya sendiri atau tabel lain, praktik terbaik adalah menggunakan subkueri yang secara jelas memisahkan kondisi gabungan dari kriteria yang memenuhi syarat baris untuk pembaruan.

Kueri UPDATE dengan beberapa kecocokan per baris menimbulkan kesalahan saat parameter konfigurasi `error_on_nondeterministic_update` disetel ke true. Untuk informasi selengkapnya, lihat error_pada_pembaruan_nondeterministik.

Anda dapat memperbarui kolom GENERATED BY DEFAULT AS IDENTITY. Kolom yang didefinisikan sebagai GENERATED BY DEFAULT AS IDENTITY dapat diperbarui dengan nilai yang Anda berikan. Untuk informasi selengkapnya, lihat GENERATED BY DEFAULT AS IDENTITY.

Contoh pernyataan UPDATE

Untuk informasi selengkapnya tentang tabel yang digunakan dalam contoh berikut, lihatDatabase sampel.

Tabel CATEGORY dalam database TICKIT berisi baris berikut:

```
+-------+----------+-----------+--------------------------------------------+
| catid | catgroup |  catname  |                  catdesc                   |
+-------+----------+-----------+--------------------------------------------+
| 5     | Sports   | MLS       | Major League Soccer                        |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts |
| 1     | Sports   | MLB       | Major League Baseball                      |
| 6     | Shows    | Musicals  | Musical theatre                            |
| 3     | Sports   | NFL       | National Football League                   |
| 8     | Shows    | Opera     | All opera and light opera                  |
| 2     | Sports   | NHL       | National Hockey League                     |
| 9     | Concerts | Pop       | All rock and pop music concerts            |
| 4     | Sports   | NBA       | National Basketball Association            |
| 7     | Shows    | Plays     | All non-musical theatre                    |
| 10    | Concerts | Jazz      | All jazz singers and bands                 |
+-------+----------+-----------+--------------------------------------------+
```

Memperbarui tabel berdasarkan rentang nilai

Perbarui kolom CATGROUP berdasarkan rentang nilai di kolom CATID.

```
UPDATE category
SET catgroup='Theatre'
WHERE catid BETWEEN 6 AND 8;

SELECT * FROM category
WHERE catid BETWEEN 6 AND 8;

+-------+----------+----------+---------------------------+
| catid | catgroup | catname  |          catdesc          |
+-------+----------+----------+---------------------------+
| 6     | Theatre  | Musicals | Musical theatre           |
| 7     | Theatre  | Plays    | All non-musical theatre   |
| 8     | Theatre  | Opera    | All opera and light opera |
+-------+----------+----------+---------------------------+
```

Memperbarui tabel berdasarkan nilai saat ini

Perbarui kolom CATNAME dan CATDESC berdasarkan nilai CATGROUP saat ini:

```
UPDATE category
SET catdesc=default, catname='Shows'
WHERE catgroup='Theatre';

SELECT * FROM category
WHERE catname='Shows';

+-------+----------+---------+---------+
| catid | catgroup | catname | catdesc |
+-------+----------+---------+---------+
| 6     | Theatre  | Shows   | NULL    |
| 7     | Theatre  | Shows   | NULL    |
| 8     | Theatre  | Shows   | NULL    |
+-------+----------+---------+---------+)
```

Dalam hal ini, kolom CATDESC disetel ke null karena tidak ada nilai default yang ditentukan saat tabel dibuat.

Jalankan perintah berikut untuk mengatur data tabel CATEGORY kembali ke nilai aslinya:

```
TRUNCATE category;
```

```
COPY category
FROM 's3://redshift-downloads/tickit/category_pipe.txt'
DELIMITER '|'
IGNOREHEADER 1
REGION 'us-east-1'
IAM_ROLE default;
```

Memperbarui tabel berdasarkan hasil subquery klausa WHERE

Perbarui tabel CATEGORY berdasarkan hasil subquery dalam klausa WHERE:

```
UPDATE category
SET catdesc='Broadway Musical'
WHERE category.catid IN
(SELECT category.catid FROM category
JOIN event ON category.catid = event.catid
JOIN venue ON venue.venueid = event.venueid
JOIN sales ON sales.eventid = event.eventid
WHERE venuecity='New York City' AND catname='Musicals');
```

Lihat tabel yang diperbarui:

```
SELECT * FROM category ORDER BY catid;

+-------+----------+-----------+--------------------------------------------+
| catid | catgroup |  catname  |                  catdesc                   |
+-------+----------+-----------+--------------------------------------------+
| 2     | Sports   | NHL       | National Hockey League                     |
| 3     | Sports   | NFL       | National Football League                   |
| 4     | Sports   | NBA       | National Basketball Association            |
| 5     | Sports   | MLS       | Major League Soccer                        |
| 6     | Shows    | Musicals  | Broadway Musical                           |
| 7     | Shows    | Plays     | All non-musical theatre                    |
| 8     | Shows    | Opera     | All opera and light opera                  |
| 9     | Concerts | Pop       | All rock and pop music concerts            |
| 10    | Concerts | Jazz      | All jazz singers and bands                 |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts |
+-------+----------+-----------+--------------------------------------------+
```

Memperbarui tabel berdasarkan hasil subquery klausa WITH

Untuk memperbarui tabel CATEGORY berdasarkan hasil subquery menggunakan klausa WITH, gunakan contoh berikut.

```
WITH u1 as (SELECT catid FROM event ORDER BY catid DESC LIMIT 1)
UPDATE category SET catid='200' FROM u1 WHERE u1.catid=category.catid;

SELECT * FROM category ORDER BY catid DESC LIMIT 1;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
| 200   | Concerts | Pop     | All rock and pop music concerts |
+-------+----------+---------+---------------------------------+
```

Memperbarui tabel berdasarkan hasil kondisi gabungan

Perbarui 11 baris asli dalam tabel KATEGORI berdasarkan baris CATID yang cocok di tabel EVENT:

```
UPDATE category SET catid=100
FROM event
WHERE event.catid=category.catid;

SELECT * FROM category ORDER BY catid;

+-------+----------+-----------+---------------------------------------------+
| catid | catgroup |  catname  |                  catdesc                    |
+-------+----------+-----------+---------------------------------------------+
| 2     | Sports   | NHL       | National Hockey League                      |
| 3     | Sports   | NFL       | National Football League                    |
| 4     | Sports   | NBA       | National Basketball Association             |
| 5     | Sports   | MLS       | Major League Soccer                         |
| 10    | Concerts | Jazz      | All jazz singers and bands                  |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts  |
| 100   | Concerts | Pop       | All rock and pop music concerts             |
| 100   | Shows    | Plays     | All non-musical theatre                     |
| 100   | Shows    | Opera     | All opera and light opera                   |
| 100   | Shows    | Musicals  | Broadway Musical                            |
+-------+----------+-----------+---------------------------------------------+
```

Perhatikan bahwa tabel EVENT tercantum dalam klausa FROM dan kondisi gabungan ke tabel target didefinisikan dalam klausa WHERE. Hanya empat baris yang memenuhi syarat untuk pembaruan. Keempat baris ini adalah baris yang nilai CATID-nya awalnya 6, 7, 8, dan 9; hanya empat kategori yang diwakili dalam tabel EVENT:

```
SELECT DISTINCT catid FROM event;
```

```
+-------+
| catid |
+-------+
| 6     |
| 7     |
| 8     |
| 9     |
+-------+
```

Perbarui 11 baris asli dalam tabel CATEGORY dengan memperluas contoh sebelumnya dan menambahkan kondisi lain ke klausa WHERE. Karena pembatasan pada kolom CATGROUP, hanya satu baris yang memenuhi syarat untuk pembaruan (meskipun empat baris memenuhi syarat untuk bergabung).

```
UPDATE category SET catid=100
FROM event
WHERE event.catid=category.catid
AND catgroup='Concerts';

SELECT * FROM category WHERE catid=100;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
| 100   | Concerts | Pop     | All rock and pop music concerts |
+-------+----------+---------+---------------------------------+
```

Cara alternatif untuk menulis contoh ini adalah sebagai berikut:

```
UPDATE category SET catid=100
FROM event JOIN category cat ON event.catid=cat.catid
WHERE cat.catgroup='Concerts';
```

Keuntungan dari pendekatan ini adalah bahwa kriteria gabungan dipisahkan dengan jelas dari kriteria lain yang memenuhi syarat baris untuk pembaruan. Perhatikan penggunaan alias CAT untuk tabel CATEGORY dalam klausa FROM.

Pembaruan dengan gabungan luar dalam klausa FROM

Contoh sebelumnya menunjukkan join dalam yang ditentukan dalam klausa FROM dari pernyataan UPDATE. Contoh berikut mengembalikan kesalahan karena klausa FROM tidak mendukung join luar ke tabel target:

```
UPDATE category SET catid=100
FROM event LEFT JOIN category cat ON event.catid=cat.catid
WHERE cat.catgroup='Concerts';
ERROR:  Target table must be part of an equijoin predicate
```

Jika gabungan luar diperlukan untuk pernyataan UPDATE, Anda dapat memindahkan sintaks join luar ke subkueri:

```
UPDATE category SET catid=100
FROM
(SELECT event.catid FROM event LEFT JOIN category cat ON event.catid=cat.catid)
 eventcat
WHERE category.catid=eventcat.catid
AND catgroup='Concerts';
```

Pembaruan dengan kolom dari tabel lain dalam klausa SET

Untuk memperbarui listing tabel dalam database sampel TICKIT dengan nilai-nilai dari sales tabel, gunakan contoh berikut.

```
SELECT listid, numtickets FROM listing WHERE sellerid = 1 ORDER BY 1 ASC LIMIT 5;

+--------+------------+
| listid | numtickets |
+--------+------------+
| 100423 | 4          |
| 108334 | 24         |
| 117150 | 4          |
| 135915 | 20         |
| 205927 | 6          |
+--------+------------+

UPDATE listing
SET numtickets = sales.sellerid
FROM sales
WHERE sales.sellerid = 1 AND listing.sellerid = sales.sellerid;
```

```
SELECT listid, numtickets FROM listing WHERE sellerid = 1 ORDER BY 1 ASC LIMIT 5;

+--------+------------+
| listid | numtickets |
+--------+------------+
| 100423 | 1          |
| 108334 | 1          |
| 117150 | 1          |
| 135915 | 1          |
| 205927 | 1          |
+--------+------------+
```

USE

Mengubah database tempat query dijalankan. SHOW USE menunjuk ke database yang terakhir digunakan dengan perintah USE. RESET USE mengatur ulang database yang digunakan. Ini berarti bahwa jika database tidak ditentukan dalam SQL, objek dicari di database saat ini.

Sintaksis

```
USE database
```

Contoh

Misalkan ada tiga database, `dev` danpdb, danpdb2. Biarkan ada dua tabel `t` dalam skema publik dari masing-masing database. Pertama, masukkan data ke dalam tabel di database yang berbeda:

```
dev=# insert into dev.public.t values (1);
INSERT 0 1
dev=# insert into pdb.public.t values (2);
INSERT 0 1
```

Tanpa menetapkan database secara eksplisit, sistem menggunakan database Anda yang terhubung. Periksa konteks database Anda saat ini:

```
dev=# show use;
Use Database

(1 row)
```

```
dev=> show search_path;
search_path
$user, public
(1 row)
```

Saat menanyakan tabel `t` tanpa menentukan database, sistem menggunakan tabel dalam database Anda saat ini:

```
dev=# select * from t;
c
----
1
(1 row)
```

Gunakan `use` perintah untuk beralih database tanpa mengubah koneksi Anda:

```
dev=# use pdb;
USE
dev=# show use;
 Use Database
--------------
 pdb
(1 row)
dev=# select * from t;
id
----
2
(1 row)
```

Anda juga dapat secara eksplisit menentukan skema:

```
dev=# select * from public.t;
id
----
2
(1 row)
```

Anda sekarang dapat membuat tabel dalam skema yang berbeda dalam database Anda saat ini:

```
dev=# create table s1.t(id int);
CREATE TABLE
```

```
dev=# insert into pdb.s1.t values (3);
INSERT 0 1
```

Jalur pencarian menentukan objek skema mana yang diakses saat Anda tidak menentukan skema:

```
dev=# set search_path to public, s1;
SET
dev=# select * from t;
 id
----
  2
(1 row)
```

Ubah urutan skema untuk mengakses tabel yang berbeda:

```
dev=# set search_path to s1, public;
SET
dev=# show search_path;
 search_path
-------------
 s1, public
(1 row)
dev=# select * from t;
 id
----
  3
(1 row)
```

Beralih ke database lain sambil mempertahankan koneksi asli Anda:

```
dev=# show use;
 Use Database
--------------
 pdb
(1 row)
dev=# use pdb2;
USE
dev=# show use;
 Use Database
--------------
 pdb2
(1 row)
```

Saat mengganti database, jalur pencarian diatur ulang ke default:

```
dev=# show search_path;
  search_path
---------------
 $user, public
(1 row)
```

Buat tabel dan masukkan data dalam database Anda saat ini:

```
dev=# create table pdb2.public.t(id int);
CREATE TABLE
dev=# insert into pdb2.public.t values (4);
INSERT 0 1
dev=# select * from t;
 id
----
  4
(1 row)
```

Dalam transaksi, Anda dapat menulis ke database saat ini dan membaca dari database apa pun menggunakan notasi tiga bagian. Ini juga termasuk database yang terhubung:

```
dev=# show use;
 Use Database
--------------
 pdb2
(1 row)

dev=# BEGIN;
BEGIN
dev=# select * from t;
 id
----
  4
(1 row)

dev=# insert into t values (5);
INSERT 0 1
dev=# select * from t;
 id
----
```

```
  4
  5
(2 rows)

dev=# select * from pdb.public.t;
 id
----
  2
(1 row)

dev=# select * from dev.public.t;
 id
----
  1
(1 row)
```

Setel ulang ke database Anda yang terhubung. Perhatikan bahwa ini tidak hanya kembali ke database yang digunakan sebelumnyapdb, tetapi mengatur ulang ke database yang terhubung. Jalur pencarian juga berubah ke jalur default:

```
dev=# RESET USE;
RESET
dev=# select * from t;
c
----
1
(1 row)
dev=# show use;
 Use Database
--------------

(1 row)

dev=# show search_path;
  search_path
---------------
 $user, public
(1 row)
```

Anda dapat mengubah database di awal transaksi, tetapi tidak setelah menjalankan kueri:

```
dev=# BEGIN;
BEGIN
```

```
dev=# use pdb;
USE
dev=# use pdb2;
USE
dev=# use pdb;
USE
dev=# select * from t;
 id
----
  2
(1 row)
dev=# use pdb2;
ERROR:  USEd Database cannot be set or reset inside a transaction after another
 command.
dev=# rollback;
ROLLBACK
(1 row)
```

Contoh Katalog Data

Pertama, buat tabel dalam skema dan katalog yang berbeda untuk mendemonstrasikan kueri lintas katalog. Mulailah dengan membuat tabel di database yang terhubung.

```
dev=# CREATE TABLE dev.public.t (col INT);
dev=# INSERT INTO dev.public.t VALUES (1);
dev=# CREATE SCHEMA write_schema;
dev=# CREATE TABLE dev.write_schema.t (state char (2));
dev=# INSERT INTO dev.write_schema.t VALUES ('WA');
```

Sekarang, buat tabel serupa di katalog yang berbeda. Ini menunjukkan cara bekerja dengan database lintas katalog.

```
dev=# CREATE TABLE my_db@my_catalog.public.t (col INT);
dev=# INSERT INTO my_db@my_catalog.public.t VALUES (100);
dev=# CREATE SCHEMA my_db@my_catalog.write_schema;
dev=# CREATE TABLE my_db@my_catalog.write_schema.t (state char (2));
dev=# INSERT INTO my_db@my_catalog.write_schema.t VALUES ('CA');
```

Periksa konteks database saat ini. Tanpa secara eksplisit mengatur database, sistem menggunakan database yang terhubung.

```
dev=# SHOW USE;
 Use Database
--------------

(1 row)

dev=# SHOW search_path;
  search_path
---------------
 $user, public
(1 row)

dev=# SELECT * FROM t;
 col
-----
   1
(1 row)
```

Atur database USed untuk menanyakan tabel dalam katalog yang berbeda.

```
dev=# USE my_db@my_catalog;

dev=# SHOW USE;
            Use Database
-------------------------------------
 my_db@my_catalog
(1 row)

dev=# SHOW search_path;
  search_path
---------------
 $user, public
(1 row)
```

Saat menanyakan tabel t, hasilnya berasal dari database cross-catalog.

```
dev=# SELECT * FROM t;
 col
-----
 100
(1 row)

dev=# SELECT * FROM public.t;
 col
-----
 100
(1 row)

dev=# SELECT * FROM my_db@my_catalog.public.t;
 col
-----
 100
(1 row)
```

Ubah jalur pencarian untuk mengakses tabel dalam skema yang berbeda dalam database USed.

```
dev=# SET search_path to write_schema;

dev=# SHOW search_path;
 search_path
--------------
 write_schema
(1 row)

dev=# SELECT * FROM t;
 state
-------
 CA
(1 row)

dev=# SELECT * FROM write_schema.t;
 state
-------
 CA
(1 row)
```

```
dev=# SELECT * FROM my_db@my_catalog.write_schema.t;
 state
-------
 CA
(1 row)
```

Meskipun USE diatur ke database lintas katalog, masih mungkin untuk secara eksplisit menanyakan database asli.

```
dev=# SELECT * FROM dev.write_schema.t;
 state
-------
 WA
(1 row)
```

Setel ulang database USed untuk merujuk lagi ke objek dalam database yang terhubung.

```
dev=# RESET USE;

dev=# SHOW USE;
 Use Database
--------------

(1 row)
```

Perhatikan bahwa search_path akan diatur ulang saat USE diatur ulang.

```
dev=# SHOW search_path;
  search_path
---------------
 $user, public
(1 row)
```

Setelah mengatur ulang, kueri sekarang merujuk ke database asli yang terhubung.

```
dev=# SELECT * FROM t;
 col
-----
   1
(1 row)

dev=# SELECT * FROM public.t;
 col
-----
   1
(1 row)

dev=# SELECT * FROM dev.public.t;
 col
-----
   1
(1 row)
```

Anda dapat memodifikasi jalur pencarian di database asli untuk mengakses skema yang berbeda.

```
dev=# SET search_path to write_schema;

dev=# SHOW search_path;
 search_path
--------------
 write_schema
(1 row)

dev=# SELECT * FROM t;
 state
-------
 WA
(1 row)

dev=# SELECT * FROM write_schema.t;
 state
-------
 WA
(1 row)
```

```
dev=# SELECT * FROM dev.write_schema.t;
 state
-------
 WA
(1 row)
```

VAKUM

Re-sorts baris dan mengklaim kembali ruang di tabel tertentu atau semua tabel dalam database saat ini.

> **Note**
>
> Hanya pengguna dengan izin tabel yang diperlukan yang dapat menyedot meja secara efektif. Jika VACUUM dijalankan tanpa izin tabel yang diperlukan, operasi selesai dengan sukses tetapi tidak berpengaruh. Untuk daftar izin tabel yang valid untuk menjalankan VACUUM secara efektif, lihat bagian Hak istimewa yang diperlukan berikut ini.

Amazon Redshift secara otomatis mengurutkan data dan menjalankan VACUUM DELETE di latar belakang. Ini mengurangi kebutuhan untuk menjalankan perintah VACUUM. Untuk informasi selengkapnya, lihat Meja penyedot debu.

Secara default, VACUUM melewatkan fase pengurutan untuk setiap tabel di mana lebih dari 95 persen baris tabel sudah diurutkan. Melewatkan fase pengurutan dapat secara signifikan meningkatkan kinerja VACUUM. Untuk mengubah ambang pengurutan atau hapus default untuk satu tabel, sertakan nama tabel dan parameter TO ambang batas PERSEN saat Anda menjalankan VACUUM.

Pengguna dapat mengakses tabel saat sedang disedot debu. Anda dapat melakukan kueri dan operasi penulisan saat tabel sedang disedot, tetapi ketika perintah bahasa manipulasi data (DML) dan vakum dijalankan secara bersamaan, keduanya mungkin membutuhkan waktu lebih lama. Jika Anda menjalankan pernyataan UPDATE dan DELETE selama vakum, kinerja sistem mungkin berkurang. VACUUM DELETE sementara memblokir operasi pembaruan dan penghapusan.

Amazon Redshift secara otomatis melakukan penyedot DELETE ONLY di latar belakang. Operasi vakum otomatis berhenti ketika pengguna menjalankan operasi bahasa definisi data (DDL), seperti ALTER TABLE.

> **Note**
>
> Sintaks dan perilaku perintah Amazon Redshift VACUUM sangat berbeda dari operasi VACUUM PostgreSQL. Misalnya, operasi VACUUM default di Amazon Redshift adalah VACUUM FULL, yang mengambil kembali ruang disk dan mengurutkan ulang semua baris. Sebaliknya, operasi VACUUM default di PostgreSQL hanya mengambil kembali ruang dan membuatnya tersedia untuk digunakan kembali.

Untuk informasi selengkapnya, lihat Meja penyedot debu.

Hak istimewa yang diperlukan

Berikut ini adalah hak istimewa yang diperlukan untuk VACUUM:

- Superuser
- Pengguna dengan hak istimewa VACUUM
- Pemilik meja
- Pemilik database kepada siapa tabel dibagikan

Sintaksis

```
VACUUM [ FULL | SORT ONLY | DELETE ONLY | REINDEX | RECLUSTER ]
[ [ table_name ] [ TO threshold PERCENT ] [ BOOST ] ]
```

Parameter

PENUH

Mengurutkan tabel yang ditentukan (atau semua tabel dalam database saat ini) dan mengklaim kembali ruang disk yang ditempati oleh baris yang ditandai untuk dihapus oleh operasi UPDATE dan DELETE sebelumnya. VACUUM FULL adalah default.

Vakum penuh tidak melakukan indeks ulang untuk tabel yang disisipkan. Untuk mengindeks ulang tabel yang disisipkan diikuti dengan vakum penuh, gunakan opsi. VACUUM REINDEX

Secara default, VACUUM FULL melewatkan fase pengurutan untuk setiap tabel yang sudah setidaknya 95 persen diurutkan. Jika VACUUM dapat melewati fase pengurutan, ia melakukan

DELETE ONLY dan mengklaim kembali ruang dalam fase hapus sehingga setidaknya 95 persen dari baris yang tersisa tidak ditandai untuk dihapus.

Jika ambang pengurutan tidak terpenuhi (misalnya, jika 90 persen baris diurutkan) dan VACUUM melakukan pengurutan penuh, maka VACUUM juga melakukan operasi penghapusan lengkap, memulihkan ruang dari 100 persen baris yang dihapus.

Anda dapat mengubah ambang vakum default hanya untuk satu tabel. Untuk mengubah ambang vakum default untuk satu tabel, sertakan nama tabel dan parameter TO ambang batas PERSEN.

HANYA MENYORTIR

Mengurutkan tabel yang ditentukan (atau semua tabel dalam database saat ini) tanpa mengambil kembali ruang yang dibebaskan oleh baris yang dihapus. Opsi ini berguna ketika mengambil kembali ruang disk tidak penting tetapi menyortir ulang baris baru itu penting. Vakum SORT ONLY mengurangi waktu yang telah berlalu untuk operasi vakum ketika wilayah yang tidak disortir tidak berisi sejumlah besar baris yang dihapus dan tidak menjangkau seluruh wilayah yang diurutkan. Aplikasi yang tidak memiliki kendala ruang disk tetapi bergantung pada pengoptimalan kueri yang terkait dengan menjaga baris tabel diurutkan dapat memperoleh manfaat dari vakum semacam ini.

Secara default, VACUUM SORT ONLY melewatkan tabel apa pun yang sudah setidaknya 95 persen diurutkan. Untuk mengubah ambang pengurutan default untuk satu tabel, sertakan nama tabel dan parameter TO ambang batas PERSEN saat Anda menjalankan VACUUM.

HAPUS SAJA

Amazon Redshift secara otomatis melakukan penyedot debu DELETE ONLY di latar belakang, jadi Anda jarang, jika pernah, perlu menjalankan vacuum DELETE ONLY.

VACUUM DELETE mengklaim kembali ruang disk yang ditempati oleh baris yang ditandai untuk dihapus oleh operasi UPDATE dan DELETE sebelumnya, dan memadatkan tabel untuk membebaskan ruang yang dikonsumsi. Operasi vakum DELETE ONLY tidak mengurutkan data tabel.

Opsi ini mengurangi waktu yang telah berlalu untuk operasi vakum ketika mengambil kembali ruang disk penting tetapi menyortir ulang baris baru tidak penting. Opsi ini juga dapat berguna ketika kinerja kueri Anda sudah optimal, dan menyortir ulang baris untuk mengoptimalkan kinerja kueri bukanlah suatu persyaratan.

Secara default, VACUUM DELETE ONLY mengklaim kembali ruang sehingga setidaknya 95 persen dari baris yang tersisa tidak ditandai untuk dihapus. Untuk mengubah ambang

penghapusan default untuk satu tabel, sertakan nama tabel dan parameter TO ambang batas PERSEN saat Anda menjalankan VACUUM.

Beberapa operasi, sepertiALTER TABLE APPEND, dapat menyebabkan tabel terfragmentasi. Saat Anda menggunakan DELETE ONLY klausa, operasi vakum mengambil kembali ruang dari tabel terfragmentasi. Nilai ambang yang sama sebesar 95 persen berlaku untuk operasi defragmentasi.

MENGINDEKS ULANG

Menganalisis distribusi nilai dalam kolom kunci pengurutan yang disisipkan, kemudian melakukan operasi VACUUM penuh. Jika REINDEX digunakan, nama tabel diperlukan.

VACUUM REINDEX membutuhkan waktu lebih lama secara signifikan daripada VACUUM FULL karena membuat izin tambahan untuk menganalisis kunci pengurutan yang disisipkan. Operasi pengurutan dan penggabungan dapat memakan waktu lebih lama untuk tabel yang disisipkan karena pengurutan yang disisipkan mungkin perlu mengatur ulang lebih banyak baris daripada pengurutan majemuk.

Jika operasi VACUUM REINDEX berakhir sebelum selesai, VACUUM berikutnya melanjutkan operasi pengindeksan ulang sebelum melakukan operasi vakum penuh.

VACUUM REINDEX tidak didukung dengan TO ambang PERSEN.

PENUTUPAN

Mengurutkan bagian-bagian tabel yang tidak disortir. Bagian tabel yang sudah diurutkan berdasarkan urutan tabel otomatis dibiarkan utuh. Perintah ini tidak menggabungkan data yang baru diurutkan dengan wilayah yang diurutkan. Itu juga tidak merebut kembali semua ruang yang ditandai untuk dihapus. Ketika perintah ini selesai, tabel mungkin tidak tampak sepenuhnya diurutkan, seperti yang ditunjukkan oleh unsorted bidang di SVV_TABLE_INFO.

Kami menyarankan Anda menggunakan VACUUM RECLUSTER untuk tabel besar dengan konsumsi yang sering dan kueri yang hanya mengakses data terbaru.

VACUUM RECLUSTER tidak didukung dengan TO AMBANG PERSEN. Jika RECLUSTER digunakan, nama tabel diperlukan.

VACUUM RECLUSTER tidak didukung pada tabel dengan kunci pengurutan dan tabel yang disisipkan dengan SEMUA gaya distribusi.

table_name

Nama meja untuk menyedot debu. Jika Anda tidak menentukan nama tabel, operasi vakum berlaku untuk semua tabel dalam database saat ini. Anda dapat menentukan tabel permanen atau sementara yang dibuat pengguna. Perintah ini tidak berarti untuk objek lain, seperti tampilan dan tabel sistem.

Jika Anda menyertakan parameter TO ambang batas PERSEN, nama tabel diperlukan.

UNTUK AM BANG PERSEN

Klausa yang menentukan ambang batas di mana VACUUM melewatkan fase pengurutan dan ambang target untuk merebut kembali ruang dalam fase hapus. Am bang peng urutan adalah persentase dari total baris yang sudah dalam urutan urutan untuk tabel yang ditentukan sebelum menyedot debu. Am bang penghapusan adalah persentase minimum dari total baris yang tidak ditandai untuk dihapus setelah menyedot debu.

Karena VACUUM mengurutkan ulang baris hanya jika persentase baris yang diurutkan dalam tabel kurang dari ambang pengurutan, Amazon Redshift sering dapat mengurangi waktu VACUUM secara signifikan. Demikian pula, ketika VACUUM tidak dibatasi untuk merebut kembali ruang dari 100 persen baris yang ditandai untuk dihapus, seringkali dapat melewati blok penulisan ulang yang hanya berisi beberapa baris yang dihapus.

Misalnya, jika Anda menentukan 75 untuk ambang batas, VACUUM melewatkan fase pengurutan jika 75 persen atau lebih dari baris tabel sudah dalam urutan urutan. Untuk fase hapus, VACUUMS menetapkan target untuk merebut kembali ruang disk sehingga setidaknya 75 persen baris tabel tidak ditandai untuk dihapus setelah vakum. Nilai ambang batas harus berupa bilangan bulat antara 0 dan 100. Defaultnya adalah 95. Jika Anda menentukan nilai 100, VACUUM selalu mengurutkan tabel kecuali sudah sepenuhnya diurutkan dan mengambil kembali ruang dari semua baris yang ditandai untuk dihapus. Jika Anda menentukan nilai 0, VACUUM tidak pernah mengurutkan tabel dan tidak pernah mengklaim kembali ruang.

Jika Anda menyertakan parameter TO ambang batas PERSEN, Anda juga harus menentukan nama tabel. Jika nama tabel dihilangkan, VACUUM gagal.

Anda tidak dapat menggunakan parameter TO ambang PERSEN dengan REINDEX.

MENINGKATKAN

Menjalankan perintah VACUUM dengan sumber daya tambahan, seperti memori dan ruang disk, saat tersedia. Dengan opsi BOOST, VACUUM beroperasi dalam satu jendela dan

memblokir penghapusan dan pembaruan secara bersamaan selama operasi VACUUM. Berjalan dengan opsi BOOST bersaing untuk sumber daya sistem, yang mungkin memengaruhi kinerja kueri. Jalankan VACUUM BOOST saat beban pada sistem ringan, seperti selama operasi pemeliharaan.

Pertimbangkan hal berikut saat menggunakan opsi BOOST:

- Ketika BOOST ditentukan, nilai table_name diperlukan.
- BOOST tidak didukung dengan REINDEX.
- BOOST diabaikan dengan DELETE ONLY.

Catatan penggunaan

Untuk sebagian besar aplikasi Amazon Redshift, penyedot debu penuh direkomendasikan. Untuk informasi selengkapnya, lihat Meja penyedot debu.

Sebelum menjalankan operasi vakum, perhatikan perilaku berikut:

- Anda tidak dapat menjalankan VACUUM dalam blok transaksi (MULAI... AKHIR). Untuk informasi lebih lanjut tentang transaksi, lihatTingkat isolasi di Amazon Redshift.
- Beberapa pertumbuhan meja mungkin terjadi ketika meja disedot debu. Perilaku ini diharapkan ketika tidak ada baris yang dihapus untuk mengklaim kembali atau urutan urutan baru tabel menghasilkan rasio kompresi data yang lebih rendah.
- Selama operasi vakum, beberapa tingkat penurunan kinerja kueri diharapkan. Kinerja normal dilanjutkan segera setelah operasi vakum selesai.
- Operasi penulisan bersamaan dilanjutkan selama operasi vakum, tetapi kami tidak menyarankan melakukan operasi penulisan saat menyedot debu. Lebih efisien untuk menyelesaikan operasi penulisan sebelum menjalankan vakum. Juga, data apa pun yang ditulis setelah operasi vakum dimulai tidak dapat disedot oleh operasi itu. Dalam hal ini, operasi vakum kedua diperlukan.
- Operasi vakum mungkin tidak dapat dimulai jika operasi pemuatan atau penyisipan sudah berlangsung. Operasi vakum sementara memerlukan akses eksklusif ke tabel untuk memulai. Akses eksklusif ini diperlukan sebentar, sehingga operasi vakum tidak memblokir beban dan sisipan bersamaan untuk jangka waktu yang signifikan.
- Operasi vakum dilewati ketika tidak ada pekerjaan yang harus dilakukan untuk tabel tertentu; Namun, ada beberapa overhead yang terkait dengan menemukan bahwa operasi dapat dilewati. Jika Anda tahu bahwa meja masih asli atau tidak memenuhi ambang vakum, jangan jalankan operasi vakum terhadapnya.

- Operasi vakum DELETE ONLY pada tabel kecil mungkin tidak mengurangi jumlah blok yang digunakan untuk menyimpan data, terutama ketika tabel memiliki sejumlah besar kolom atau cluster menggunakan sejumlah besar irisan per node. Operasi vakum ini menambahkan satu blok per kolom per irisan untuk memperhitungkan sisipan bersamaan ke dalam tabel, dan ada potensi overhead ini lebih besar daripada pengurangan jumlah blok dari ruang disk yang direklamasi. Misalnya, jika tabel 10 kolom pada cluster 8-node menempati 1000 blok sebelum vakum, vakum tidak mengurangi jumlah blok aktual kecuali lebih dari 80 blok ruang disk direklamasi karena baris yang dihapus. (Setiap blok data menggunakan 1 MB.)

Operasi vakum otomatis berhenti jika salah satu dari kondisi berikut terpenuhi:

- Pengguna menjalankan operasi bahasa definisi data (DDL), seperti ALTER TABLE, yang memerlukan kunci eksklusif pada meja yang sedang dikerjakan vakum otomatis.
- Periode beban cluster yang tinggi.

Dukungan untuk VACUUM bersamaan

Amazon Redshift mendukung menjalankan beberapa transaksi vakum secara bersamaan di sesi yang berbeda dalam cluster atau kelompok kerja. Ini berarti Anda dapat mengeluarkan contoh yang berbeda dan beberapa dari semua mode vakum sekaligus, dengan setiap transaksi vakum pada meja yang unik. Dua operasi vakum tidak dapat bekerja pada satu meja secara bersamaan.

Pedoman untuk menjalankan vakum bersamaan

- Saat menjalankan transaksi vakum bersamaan di sesi yang berbeda, Anda harus memantau sumber daya sistem, dan menghindari menjalankan terlalu banyak operasi vakum secara bersamaan.
- Tingkat konkurensi yang disarankan tergantung pada jumlah ruang yang akan direklamasi, baik jumlah dan lebar baris yang akan diurutkan, ukuran gudang, dan ukuran beban kerja Anda yang berjalan bersamaan dengan operasi VACUUM.
- Bergantung pada mode transaksi vakum, mulailah dengan dua operasi vakum bersamaan, dan tambahkan lebih banyak tergantung pada waktu berjalan dan beban sistemnya. Sama seperti kueri berat lainnya yang dikeluarkan oleh pengguna, operasi vakum dapat mulai mengantri jika Anda menjalankan terlalu banyak secara bersamaan ketika Amazon Redshift mencapai batas sumber daya sistem.

- Jalankan beberapa operasi Vacuum BOOST dengan hati-hati. Menjalankan Vacuum dengan opsi BOOST bersaing untuk sumber daya sistem, yang mungkin memengaruhi kinerja kueri. Jalankan VACUUM BOOST saat beban pada sistem ringan, seperti selama operasi pemeliharaan.
- Jika Anda tidak menentukan nama tabel, operasi vakum berlaku untuk semua tabel dalam database saat ini. Operasi vakum ini masih berjalan secara berurutan.

Contoh

Dapatkan kembali ruang dan database dan urutkan ulang baris di semua tabel berdasarkan ambang vakum default 95 persen.

```
vacuum;
```

Dapatkan kembali ruang dan urutkan ulang baris dalam tabel SALES berdasarkan ambang batas default 95 persen.

```
vacuum sales;
```

Selalu dapatkan kembali ruang dan urutkan ulang baris di tabel PENJUALAN.

```
vacuum sales to 100 percent;
```

Re-sort baris dalam tabel PENJUALAN hanya jika kurang dari 75 persen baris sudah diurutkan.

```
 vacuum sort only sales to 75 percent;
```

Dapatkan kembali ruang di tabel SALES sedemikian rupa sehingga setidaknya 75 persen dari baris yang tersisa tidak ditandai untuk dihapus setelah vakum.

```
vacuum delete only sales to 75 percent;
```

Mengindeks ulang dan kemudian menyedot tabel LISTING.

```
vacuum reindex listing;
```

Perintah berikut mengembalikan kesalahan.

```
vacuum reindex listing to 75 percent;
```

Rekluster dan kemudian vakum tabel LISTING.

```
vacuum recluster listing;
```

Rekluster dan kemudian vakum tabel LISTING dengan opsi BOOST.

```
vacuum recluster listing boost;
```

Referensi fungsi SQL

Topik

- Node—hanya fungsi utama
- Fungsi agregat
- Fungsi array
- Bit-wise fungsi agregat
- Ekspresi bersyarat
- Fungsi pemformatan tipe data
- Fungsi tanggal dan waktu
- Fungsi hash
- HyperLogLog fungsi
- Fungsi JSON
- Fungsi pembelajaran mesin
- Fungsi matematika
- Fungsi objek
- Fungsi spasial
- Fungsi string
- Fungsi informasi tipe SUPER
- Fungsi dan operator VARBYTE

- Fungsi jendela
- Fungsi administrasi sistem
- Fungsi informasi sistem

Amazon Redshift mendukung sejumlah fungsi yang merupakan ekstensi standar SQL, serta fungsi agregat standar, fungsi skalar, dan fungsi jendela.

> **Note**
>
> Amazon Redshift didasarkan pada PostgreSQL. Amazon Redshift dan PostgreSQL memiliki sejumlah perbedaan yang sangat penting yang harus Anda ketahui saat merancang dan mengembangkan aplikasi gudang data Anda. Untuk informasi selengkapnya tentang perbedaan Amazon Redshift SQL dari PostgreSQL, lihat. Amazon Redshift dan PostgreSQL

Node—hanya fungsi utama

Beberapa kueri Amazon Redshift didistribusikan dan dijalankan pada node komputasi; kueri lain berjalan secara eksklusif pada node pemimpin.

Node pemimpin mendistribusikan SQL ke node komputasi ketika kueri mereferensikan tabel atau tabel sistem yang dibuat pengguna (tabel dengan awalan STL atau STV dan tampilan sistem dengan awalan SVL atau SVV). Kueri yang hanya mereferensikan tabel katalog (tabel dengan awalan PG, seperti PG_TABLE_DEF) atau yang tidak mereferensikan tabel apa pun, berjalan secara eksklusif pada node pemimpin.

Beberapa fungsi Amazon Redshift SQL hanya didukung pada node pemimpin dan tidak didukung pada node komputasi. Kueri yang menggunakan fungsi leader-node harus berjalan secara eksklusif pada node pemimpin, bukan pada node komputasi, atau akan mengembalikan kesalahan.

Dokumentasi untuk setiap fungsi leader-node saja menyertakan catatan yang menyatakan bahwa fungsi akan mengembalikan kesalahan jika mereferensikan tabel yang ditentukan pengguna atau tabel sistem Amazon Redshift.

Untuk informasi selengkapnya, lihat Fungsi SQL didukung pada node pemimpin.

Fungsi SQL berikut hanya fungsi leader-node dan tidak didukung pada node komputasi:

Fungsi informasi sistem

- CURRENT_SCHEMA
- SKEMA SAAT INI
- MEMILIKI HAK ISTIMEWA BASIS DATA
- MEMILIKI HAK ISTIMEWA SKEMA
- MEMILIKI HAK ISTIMEWA TABEL_

Fungsi string

- SUBSTR

Fungsi matematika

- FAKTORIAL
- LOG

Fungsi hanya leader-node berikut tidak digunakan lagi dan tidak lagi didukung:

Fungsi tanggal

- USIA
- CURRENT_TIME
- CURRENT_TIMESTAMP
- LOCALTIME
- ISFINIT
- SEKARANG

Fungsi string

- GETBIT
- DAPAT_BYTE
- SET_BIT
- SET_BYTE

- TO_ASCII

Fungsi agregat

Topik

- Fungsi ANY_VALUE
- APPROXIMATE PERCENTILE_DISC fungsi
- Fungsi AVG
- Fungsi COUNT
- Fungsi LISTAGG
- Fungsi MAX
- Fungsi MEDIAN
- Fungsi MIN
- Fungsi PERCENTILE_CONT
- Fungsi STDDEV_SAMP dan STDDEV_POP
- Fungsi SUM
- Fungsi VAR_SAMP dan VAR_POP

Fungsi agregat menghitung nilai hasil tunggal dari satu set nilai input.

Pernyataan SELECT menggunakan fungsi agregat dapat mencakup dua klausa opsional: GROUP BY dan HAVING. Sintaks untuk klausa ini adalah sebagai berikut (menggunakan fungsi COUNT sebagai contoh):

```
SELECT count (*) expression FROM table_reference
WHERE condition [GROUP BY expression ] [ HAVING condition]
```

Klausa GROUP BY mengumpulkan dan mengelompokkan hasil berdasarkan nilai unik dalam kolom atau kolom tertentu. Klausa HAVING membatasi hasil yang dikembalikan ke baris di mana kondisi agregat tertentu benar, seperti count (*) > 1. Klausa HAVING digunakan dengan cara yang sama seperti WHERE untuk membatasi baris berdasarkan nilai kolom. Untuk contoh klausa tambahan ini, lihatCOUNT.

Fungsi agregat tidak menerima fungsi agregat bersarang atau fungsi jendela sebagai argumen.

Fungsi ANY_VALUE

Fungsi ANY_VALUE mengembalikan nilai apa pun dari nilai ekspresi masukan secara nondeterministik. Fungsi ini kembali NULL jika ekspresi input tidak menghasilkan baris apa pun yang dikembalikan. Fungsi juga dapat kembali NULL jika ada NULL nilai dalam ekspresi input. Jika input berisi NULL nilai yang dicampur dengan non- NULL nilai, NULL mungkin dikembalikan. Jika semua nilai adalahNULL, NULL dikembalikan. Jika tidak ada baris yang cocok dengan kondisi, NULL dikembalikan.

Sintaksis

```
ANY_VALUE( [ DISTINCT | ALL ] expression )
```

Argumen

BERBEDA | SEMUA

Tentukan DISTINCT atau ALL untuk mengembalikan nilai apa pun dari nilai ekspresi masukan. Argumen DISTINCT tidak berpengaruh dan diabaikan.

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ek spresi adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- PRESISI GANDA
- BOOLEAN
- CHAR
- VARCHAR
- DATE
- TIMESTAMP

- TIMESTAMPTZ
- TIME
- TIMETZ
- INTERVAL TAHUN KE BULAN
- INTERVAL HARI KE DETIK
- WARBYTE
- SUPER
- HLLSKETSA
- GEOMETRY
- GEOGRAPHY

Pengembalian

Mengembalikan tipe data yang sama dengan ekspresi.

Catatan penggunaan

Jika pernyataan yang menentukan fungsi ANY_VALUE untuk kolom juga menyertakan referensi kolom kedua, kolom kedua harus muncul dalam klausa GROUP BY atau dimasukkan dalam fungsi agregat.

Contoh

Contoh menggunakan tabel peristiwa yang dibuat di Langkah 4: Muat data sampel dari Amazon S3 di Panduan Memulai Amazon Redshift. Contoh berikut mengembalikan instance dari setiap dateid di mana eventname adalah Eagles.

```
select any_value(dateid) as dateid, eventname from event where eventname ='Eagles'
 group by eventname;
```

Berikut ini adalah hasilnya.

```
dateid | eventname
-------+---------------
 1878  | Eagles
```

Contoh berikut mengembalikan instance dari setiap dateid di mana eventname adalah Eagles atau Cold War Kids.

```
select any_value(dateid) as dateid, eventname from event where eventname in('Eagles',
 'Cold War Kids') group by eventname;
```

Berikut ini adalah hasilnya.

```
dateid | eventname
-------+---------------
 1922  | Cold War Kids
 1878  | Eagles
```

APPROXIMATE PERCENTILE_DISC fungsi

APPROXIMATE PERCENTILE_DISC adalah fungsi distribusi terbalik yang mengasumsikan model distribusi diskrit. Dibutuhkan nilai persentil dan spesifikasi pengurutan dan mengembalikan elemen dari himpunan yang diberikan. Pendekatan memungkinkan fungsi berjalan lebih cepat, dengan kesalahan relatif rendah sekitar 0,5 persen.

Untuk nilai persentil tertentu, APPROXIMATE PERCENTILE_DISC menggunakan algoritma ringkasan kuantil untuk mendekati persentil diskrit ekspresi dalam klausa ORDER BY. APPROXIMATE PERCENTILE_DISC mengembalikan nilai dengan nilai distribusi kumulatif terkecil (sehubungan dengan spesifikasi pengurutan yang sama) yang lebih besar dari atau sama dengan persentil.

Sintaksis

```
APPROXIMATE  PERCENTILE_DISC ( percentile )
WITHIN GROUP (ORDER BY expr)
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Klausa yang menentukan numerik atau date/time nilai untuk mengurutkan dan menghitung persentil.

Pengembalian

Jenis data yang sama dengan ekspresi ORDER BY dalam klausa WITHIN GROUP.

Catatan penggunaan

Jika pernyataan APPROXIMATE PERCENTILE_DISC menyertakan klausa GROUP BY, kumpulan hasil terbatas. Batas bervariasi berdasarkan jenis node dan jumlah node. Jika batas terlampaui, fungsi gagal dan mengembalikan kesalahan berikut.

```
GROUP BY limit for approximate percentile_disc exceeded.
```

Jika Anda perlu mengevaluasi lebih banyak kelompok daripada batas yang diizinkan, pertimbangkan untuk menggunakannyaFungsi PERCENTILE_CONT.

Contoh

Contoh berikut mengembalikan jumlah penjualan, total penjualan, dan nilai persentil kelima puluh untuk 10 tanggal teratas.

```
select top 10 date.caldate,
count(totalprice), sum(totalprice),
approximate percentile_disc(0.5)
within group (order by totalprice)
from listing
join date on listing.dateid = date.dateid
group by date.caldate
order by 3 desc;

caldate    | count | sum        | percentile_disc
-----------+-------+------------+----------------
2008-01-07 |   658 | 2081400.00 |         2020.00
2008-01-02 |   614 | 2064840.00 |         2178.00
2008-07-22 |   593 | 1994256.00 |         2214.00
2008-01-26 |   595 | 1993188.00 |         2272.00
2008-02-24 |   655 | 1975345.00 |         2070.00
2008-02-04 |   616 | 1972491.00 |         1995.00
2008-02-14 |   628 | 1971759.00 |         2184.00
2008-09-01 |   600 | 1944976.00 |         2100.00
2008-07-29 |   597 | 1944488.00 |         2106.00
2008-07-23 |   592 | 1943265.00 |         1974.00
```

Fungsi AVG

Fungsi AVG mengembalikan rata-rata (rata-rata aritmatika) dari nilai ekspresi input. Fungsi AVG bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaksis

```
AVG ( [ DISTINCT | ALL ] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ek spresi adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- NUMERIC
- DECIMAL
- REAL
- PRESISI GANDA
- SUPER

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung rata-rata. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung rata-rata. ALL adalah default.

Jenis Data

Jenis argumen yang didukung oleh fungsi AVG adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, DOUBLE PRECISION, dan SUPER.

Jenis pengembalian yang didukung oleh fungsi AVG adalah:

- BIGINT untuk setiap argumen tipe integer
- PRESISI GANDA untuk argumen float point

- Mengembalikan tipe data yang sama dengan ekspresi untuk jenis argumen lainnya.

Presisi default untuk hasil fungsi AVG dengan argumen NUMERIC atau DECIMAL adalah 38. Skala hasilnya sama dengan skala argumen. Misalnya, AVG kolom DEC (5,2) mengembalikan tipe data DEC (38,2).

Contoh

Temukan jumlah rata-rata yang dijual per transaksi dari tabel PENJUALAN:

```
select avg(qtysold)from sales;

avg
-----
2
(1 row)
```

Temukan harga total rata-rata yang tercantum untuk semua listing:

```
select avg(numtickets*priceperticket) as avg_total_price from listing;

avg_total_price
-----------------
3034.41
(1 row)
```

Temukan harga rata-rata yang dibayarkan, dikelompokkan berdasarkan bulan dalam urutan menurun:

```
select avg(pricepaid) as avg_price, month
from sales, date
where sales.dateid = date.dateid
group by month
order by avg_price desc;

avg_price | month
-----------+-------
659.34 | MAR
655.06 | APR
645.82 | JAN
643.10 | MAY
```

```
642.72 | JUN
642.37 | SEP
640.72 | OCT
640.57 | DEC
635.34 | JUL
635.24 | FEB
634.24 | NOV
632.78 | AUG
(12 rows)
```

Fungsi COUNT

Fungsi COUNT menghitung baris yang ditentukan oleh ekspresi.

Fungsi COUNT memiliki variasi berikut.

- COUNT (*) menghitung semua baris dalam tabel target apakah mereka menyertakan nol atau tidak.
- COUNT (ekspresi) menghitung jumlah baris dengan nilai non-null dalam kolom atau ekspresi tertentu.
- COUNT (ekspresi DISTINCT) menghitung jumlah nilai non-null yang berbeda dalam kolom atau ekspresi.
- APPROXIMATE COUNT DISTINCT memperkirakan jumlah nilai non-NULL yang berbeda dalam kolom atau ekspresi.

Sintaksis

```
COUNT( * | expression )
```

```
COUNT ( [ DISTINCT | ALL ] expression )
```

```
APPROXIMATE COUNT ( DISTINCT expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Fungsi COUNT mendukung semua tipe data argumen.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum melakukan penghitungan. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung. ALL adalah default.

PERKIRAAN

Ketika digunakan dengan APPROXIMATE, fungsi COUNT DISTINCT menggunakan HyperLogLog algoritma untuk memperkirakan jumlah nilai non-NULL yang berbeda dalam kolom atau ekspresi. Kueri yang menggunakan kata kunci APPROXIMATE berjalan jauh lebih cepat, dengan kesalahan relatif rendah sekitar 2%. Pendekatan dijamin untuk kueri yang mengembalikan sejumlah besar nilai berbeda, dalam jutaan atau lebih per kueri, atau per grup, jika ada grup demi klausa. Untuk set nilai berbeda yang lebih kecil, dalam ribuan, perkiraan mungkin lebih lambat daripada hitungan yang tepat. APPROXIMATE hanya dapat digunakan dengan COUNT DISTINCT.

Jenis pengembalian

Fungsi COUNT mengembalikan BIGINT.

Contoh

Hitung semua pengguna dari negara bagian Florida:

```
select count(*) from users where state='FL';

count
-------
510
```

Hitung semua nama acara dari tabel EVENT:

```
select count(eventname) from event;

count
-------
8798
```

Hitung semua nama acara dari tabel EVENT:

```
select count(all eventname) from event;

count
-------
8798
```

Hitung semua ID venue unik dari tabel EVENT:

```
select count(distinct venueid) as venues from event;

venues
--------
204
```

Hitung berapa kali setiap penjual mendaftarkan batch lebih dari empat tiket untuk dijual. Kelompokkan hasil berdasarkan ID penjual:

```
select count(*), sellerid from listing
where numtickets > 4
group by sellerid
order by 1 desc, 2;

count | sellerid
------+----------
12    |    6386
11    |    17304
11    |    20123
11    |    25428
...
```

Contoh berikut membandingkan nilai pengembalian dan waktu eksekusi untuk COUNT dan APPROXIMATE COUNT.

```
select  count(distinct pricepaid) from sales;

count
-------
  4528


Time: 48.048 ms
```

```
select approximate count(distinct pricepaid) from sales;

count
-------
  4553


Time: 21.728 ms
```

Fungsi LISTAGG

Untuk setiap grup dalam kueri, fungsi agregat LISTAGG mengurutkan baris untuk grup tersebut sesuai dengan ekspresi ORDER BY, kemudian menggabungkan nilai-nilai menjadi satu string.

Sintaksis

```
LISTAGG( [DISTINCT] aggregate_expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]
```

Argumen

DISTINCT

Klausa yang menghilangkan nilai duplikat dari ekspresi yang ditentukan sebelum menggabungkan. Spasi akhir diabaikan. Misalnya, string `'a'` dan `'a '` diperlakukan sebagai duplikat. LISTAGG menggunakan nilai pertama yang ditemui. Untuk informasi selengkapnya, lihat Signifikansi blanko yang tertinggal.

ekspresi agregat

Setiap ekspresi yang valid, seperti nama kolom, yang menyediakan nilai untuk digabungkan. Nilai NULL dan string kosong diabaikan.

pembatas

Konstanta string untuk memisahkan nilai gabungan. Default-nya adalah NULL.

DALAM GRUP (ORDER BY order_list)

Klausa yang menentukan urutan dari nilai agregat.

Pengembalian

VARCHAR (MAKS). Jika set hasil lebih besar dari ukuran VARCHAR maksimum, LISTAGG mengembalikan kesalahan berikut:

```
Invalid operation: Result size exceeds LISTAGG limit
```

Catatan penggunaan

- Jika pernyataan menyertakan beberapa fungsi LISTAGG yang menggunakan klausa WITHIN GROUP, setiap klausa WITHIN GROUP harus menggunakan nilai ORDER BY yang sama.

  Misalnya, pernyataan berikut mengembalikan kesalahan.

  ```
  SELECT LISTAGG(sellerid)
  WITHIN GROUP (ORDER BY dateid) AS sellers,
  LISTAGG(dateid)
  WITHIN GROUP (ORDER BY sellerid) AS dates
  FROM sales;
  ```

  Pernyataan berikut berjalan dengan sukses.

  ```
  SELECT LISTAGG(sellerid)
  WITHIN GROUP (ORDER BY dateid) AS sellers,
  LISTAGG(dateid)
  WITHIN GROUP (ORDER BY dateid) AS dates
  FROM sales;

  SELECT LISTAGG(sellerid)
  WITHIN GROUP (ORDER BY dateid) AS sellers,
  LISTAGG(dateid) AS dates
  FROM sales;
  ```

- Anda tidak dapat menggunakan fungsi agregat LISTAGG, PERCENTILE_CONT, dan MEDIAN dengan fungsi agregat berbeda lainnya.

Contoh

Contoh berikut menggabungkan ID penjual, diurutkan berdasarkan ID penjual.

```
SELECT LISTAGG(sellerid, ', ')
```

```
WITHIN GROUP (ORDER BY sellerid)
FROM sales
WHERE eventid = 4337;

listagg

-------------------------------------------------------------------------------------------------
380, 380, 1178, 1178, 1178, 2731, 8117, 12905, 32043, 32043, 32043, 32432, 32432,
 38669, 38750, 41498, 45676, 46324, 47188, 47188, 48294
```

Contoh berikut menggunakan DISTINCT untuk mengembalikan daftar ID penjual unik.

```
SELECT LISTAGG(DISTINCT sellerid, ', ')
WITHIN GROUP (ORDER BY sellerid)
FROM sales
WHERE eventid = 4337;

listagg

-------------------------------------------------------------------------------------------------
380, 1178, 2731, 8117, 12905, 32043, 32432, 38669, 38750, 41498, 45676, 46324, 47188,
 48294
```

Contoh berikut menggabungkan ID penjual dalam urutan tanggal.

```
SELECT LISTAGG(sellerid, ', ')
WITHIN GROUP (ORDER BY dateid)
FROM sales
WHERE eventid = 4337;

   listagg
-------------------------------------------------------------------------------------------------
 41498, 47188, 47188, 1178, 1178, 1178, 380, 45676, 46324, 48294, 32043, 32043, 32432,
 12905, 8117, 38750, 2731, 32432, 32043, 380, 38669
```

Contoh berikut mengembalikan daftar tanggal penjualan yang dipisahkan pipa untuk pembeli dengan ID 660.

```
SELECT LISTAGG(
    (SELECT caldate FROM date WHERE date.dateid=sales.dateid), ' | '
)
WITHIN GROUP (ORDER BY sellerid DESC, salesid ASC)
```

```
FROM sales
WHERE buyerid = 660;

             listagg
-------------------------------------------------
2008-07-16 | 2008-07-09 | 2008-01-01 | 2008-10-26
```

Contoh berikut mengembalikan daftar ID penjualan yang dipisahkan koma untuk ID pembeli 660, 661, dan 662.

```
SELECT buyerid,
LISTAGG(salesid,', ')
WITHIN GROUP (ORDER BY salesid) AS sales_id
FROM sales
WHERE buyerid BETWEEN 660 AND 662
GROUP BY buyerid
ORDER BY buyerid;

buyerid |                sales_id
--------+-------------------------------------------------------
660     | 32872, 33095, 33514, 34548
661     | 19951, 20517, 21695, 21931
662     | 3318, 3823, 4215, 51980, 53202, 55908, 57832, 171603
```

Fungsi MAX

Fungsi MAX mengembalikan nilai maksimum dalam satu set baris. DISTINCT atau ALL dapat digunakan tetapi tidak mempengaruhi hasilnya.

Sintaksis

```
MAX ( [ DISTINCT | ALL ] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ek spresi adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER

- BIGINT
- DECIMAL
- REAL
- PRESISI GANDA
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- WARBYTE
- SUPER

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung maksimum. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung maksimum. ALL adalah default.

Jenis Data

Mengembalikan tipe data yang sama dengan ekspresi. Setara Boolean dari fungsi MIN adalahFungsi BOOL_AND, dan setara Boolean dari MAX adalah. Fungsi BOOL_OR

Contoh

Temukan harga tertinggi yang dibayarkan dari semua penjualan:

```
select max(pricepaid) from sales;

max
----------
12624.00
(1 row)
```

Temukan harga tertinggi yang dibayarkan per tiket dari semua penjualan:

```
select max(pricepaid/qtysold) as max_ticket_price
from sales;

max_ticket_price
-----------------
2500.00000000
(1 row)
```

Fungsi MEDIAN

Menghitung nilai median untuk rentang nilai. NULLnilai dalam rentang diabaikan.

MEDIAN adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu.

MEDIAN adalah kasus khususPERSENTILE_CONT.

Sintaksis

```
MEDIAN(median_expression)
```

Argumen

median_ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

Jenis Data

Jenis pengembalian ditentukan oleh tipe data median_expression. Tabel berikut menunjukkan tipe pengembalian untuk setiap tipe data median_expression.

Jenis masukan	Jenis pengembalian
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP

Jenis masukan	Jenis pengembalian
TIMESTAMPTZ	TIMESTAMPTZ

Catatan penggunaan

Jika argumen median_expression adalah tipe DECIMAL data yang ditentukan dengan presisi maksimum 38 digit, ada kemungkinan MEDIAN akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi MEDIAN melebihi 38 digit, hasilnya dipotong agar pas, yang menyebabkan hilangnya presisi. Jika, selama interpolasi, hasil perantara melebihi presisi maksimum, terjadi luapan numerik dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi yang lebih rendah atau melemparkan argumen median_expression ke presisi yang lebih rendah.

Jika pernyataan menyertakan beberapa panggilan ke fungsi agregat berbasis pengurutan (LISTAGG, PERCENTILE_CONT, atau MEDIAN), semuanya harus menggunakan nilai ORDER BY yang sama. Perhatikan bahwa MEDIAN menerapkan urutan implisit berdasarkan nilai ekspresi.

Misalnya, pernyataan berikut mengembalikan kesalahan.

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

An error occurred when executing the SQL command:
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

Pernyataan berikut berjalan dengan sukses.

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(salesid)
```

```
FROM sales
GROUP BY salesid, pricepaid;
```

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Contoh berikut menunjukkan bahwa MEDIAN menghasilkan hasil yang sama dengan PERCENTILE_CONT (0,5).

```
SELECT TOP 10 DISTINCT sellerid, qtysold,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold),
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid, qtysold;

+----------+---------+-----------------+--------+
| sellerid | qtysold | percentile_cont | median |
+----------+---------+-----------------+--------+
|        2 |       2 |               2 |      2 |
|       26 |       1 |               1 |      1 |
|       33 |       1 |               1 |      1 |
|       38 |       1 |               1 |      1 |
|       43 |       1 |               1 |      1 |
|       48 |       2 |               2 |      2 |
|       48 |       3 |               3 |      3 |
|       77 |       4 |               4 |      4 |
|       85 |       4 |               4 |      4 |
|       95 |       2 |               2 |      2 |
+----------+---------+-----------------+--------+
```

Contoh berikut menemukan kuantitas median yang dijual untuk setiap sellerid.

```
SELECT sellerid,
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid
ORDER BY sellerid
LIMIT 10;

+----------+--------+
| sellerid | median |
```

```
+----------+--------+
|        1 |    1.5 |
|        2 |      2 |
|        3 |      2 |
|        4 |      2 |
|        5 |      1 |
|        6 |      1 |
|        7 |    1.5 |
|        8 |      1 |
|        9 |      4 |
|       12 |      2 |
+----------+--------+
```

Untuk memverifikasi hasil kueri sebelumnya untuk sellerid pertama, gunakan contoh berikut.

```
SELECT qtysold
FROM sales
WHERE sellerid=1;

+---------+
| qtysold |
+---------+
|       2 |
|       1 |
+---------+
```

Fungsi MIN

Fungsi MIN mengembalikan nilai minimum dalam satu set baris. DISTINCT atau ALL dapat digunakan tetapi tidak mempengaruhi hasilnya.

Sintaksis

```
MIN ( [ DISTINCT | ALL ] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ek spresi adalah salah satu tipe data berikut:

- SMALLINT

- INTEGER
- BIGINT
- DECIMAL
- REAL
- PRESISI GANDA
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- WARBYTE
- SUPER

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung minimum. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung minimum. ALL adalah default.

Jenis Data

Mengembalikan tipe data yang sama dengan ekspresi. Setara Boolean dari fungsi MIN adalahFungsi BOOL_AND, dan setara Boolean dari MAX adalah. Fungsi BOOL_OR

Contoh

Temukan harga terendah yang dibayarkan dari semua penjualan:

```
select min(pricepaid) from sales;

min
-------
20.00
(1 row)
```

Temukan harga terendah yang dibayarkan per tiket dari semua penjualan:

```
select min(pricepaid/qtysold)as min_ticket_price
from sales;

min_ticket_price
------------------
20.00000000
(1 row)
```

Fungsi PERCENTILE_CONT

PERCENTILE_CONT adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu. Dibutuhkan nilai persentil dan spesifikasi pengurutan, dan mengembalikan nilai interpolasi yang akan jatuh ke dalam nilai persentil yang diberikan sehubungan dengan spesifikasi pengurutan.

PERCENTILE_CONT menghitung interpolasi linier antara nilai-nilai setelah mengurutkannya. Menggunakan nilai persentil `(P)` dan jumlah baris bukan nol `(N)` dalam grup agregasi, fungsi menghitung nomor baris setelah mengurutkan baris sesuai dengan spesifikasi pengurutan. Nomor baris ini `(RN)` dihitung sesuai dengan rumus`RN = (1+ (P*(N-1))`. Hasil akhir dari fungsi agregat dihitung dengan interpolasi linier antara nilai-nilai dari baris pada nomor `CRN = CEILING(RN)` baris dan. `FRN = FLOOR(RN)`

Hasil akhirnya adalah sebagai berikut.

`(CRN = FRN = RN)`Jika hasilnya adalah `(value of expression from row at RN)`

Jika tidak, hasilnya adalah sebagai berikut:

`(CRN - RN) * (value of expression for row at FRN) + (RN - FRN) * (value of expression for row at CRN)`.

Sintaksis

```
PERCENTILE_CONT(percentile)
WITHIN GROUP(ORDER BY expr)
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. `NULL`nilai-nilai diabaikan dalam perhitungan.

expres

Menentukan numerik atau date/time nilai untuk mengurutkan dan menghitung persentil.

Pengembalian

Jenis pengembalian ditentukan oleh tipe data ekspresi ORDER BY dalam klausa WITHIN GROUP. Tabel berikut menunjukkan tipe pengembalian untuk setiap tipe data ekspresi ORDER BY.

Jenis masukan	Jenis pengembalian
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

Catatan penggunaan

Jika ekspresi ORDER BY adalah tipe data DECIMAL yang ditentukan dengan presisi maksimum 38 digit, ada kemungkinan bahwa PERCENTILE_CONT akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi PERCENTILE_CONT melebihi 38 digit, hasilnya dipotong agar pas, yang menyebabkan hilangnya presisi.. Jika, selama interpolasi, hasil perantara melebihi presisi maksimum, terjadi luapan numerik dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi yang lebih rendah atau merapikan ekspresi ORDER BY ke presisi yang lebih rendah.

Jika pernyataan menyertakan beberapa panggilan ke fungsi agregat berbasis pengurutan (LISTAGG, PERCENTILE_CONT, atau MEDIAN), semuanya harus menggunakan nilai ORDER BY yang sama. Perhatikan bahwa MEDIAN menerapkan urutan implisit berdasarkan nilai ekspresi.

Misalnya, pernyataan berikut mengembalikan kesalahan.

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
```

```
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

An error occurred when executing the SQL command:
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

Pernyataan berikut berjalan dengan sukses.

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(salesid)
FROM sales
GROUP BY salesid, pricepaid;
```

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Contoh berikut menunjukkan bahwa PERCENTILE_CONT (0.5) menghasilkan hasil yang sama dengan MEDIAN.

```
SELECT TOP 10 DISTINCT sellerid, qtysold,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold),
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid, qtysold;

+----------+---------+-----------------+--------+
| sellerid | qtysold | percentile_cont | median |
+----------+---------+-----------------+--------+
|        2 |       2 |               2 |      2 |
|       26 |       1 |               1 |      1 |
|       33 |       1 |               1 |      1 |
|       38 |       1 |               1 |      1 |
```

```
|       43 |       1 |                1 |      1 |
|       48 |       2 |                2 |      2 |
|       48 |       3 |                3 |      3 |
|       77 |       4 |                4 |      4 |
|       85 |       4 |                4 |      4 |
|       95 |       2 |                2 |      2 |
+----------+---------+------------------+--------+
```

Contoh berikut menemukan PERCENTILE_CONT (0,5) dan PERCENTILE_CONT (0,75) untuk jumlah yang dijual untuk setiap sellerid dalam tabel SALES.

```
SELECT sellerid,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold) as pct_50,
PERCENTILE_CONT(0.75) WITHIN GROUP(ORDER BY qtysold) as pct_75
FROM sales
GROUP BY sellerid
ORDER BY sellerid
LIMIT 10;

+----------+--------+---------+
| sellerid | pct_50 | pct_75 |
+----------+--------+---------+
|        1 |    1.5 |    1.75 |
|        2 |      2 |    2.25 |
|        3 |      2 |       3 |
|        4 |      2 |       2 |
|        5 |      1 |     1.5 |
|        6 |      1 |       1 |
|        7 |    1.5 |    1.75 |
|        8 |      1 |       1 |
|        9 |      4 |       4 |
|       12 |      2 |    3.25 |
+----------+--------+---------+
```

Untuk memverifikasi hasil kueri sebelumnya untuk sellerid pertama, gunakan contoh berikut.

```
SELECT qtysold
FROM sales
WHERE sellerid=1;

+---------+
| qtysold |
+---------+
```

```
|       2 |
|       1 |
+---------+
```

Fungsi STDDEV_SAMP dan STDDEV_POP

Fungsi STDDEV_SAMP dan STDDEV_POP mengembalikan sampel dan standar deviasi populasi dari satu set nilai numerik (integer, desimal, atau floating-point). Hasil fungsi STDDEV_SAMP setara dengan akar kuadrat dari varians sampel dari kumpulan nilai yang sama.

STDDEV_SAMP dan STDDEV adalah sinonim untuk fungsi yang sama.

Sintaksis

```
STDDEV_SAMP | STDDEV ( [ DISTINCT | ALL ] expression)
STDDEV_POP ( [ DISTINCT | ALL ] expression)
```

Ekspresi harus memiliki tipe data integer, desimal, atau float point. Terlepas dari jenis data ekspresi, tipe pengembalian fungsi ini adalah angka presisi ganda.

> **Note**
>
> Standar deviasi dihitung menggunakan aritmatika titik mengambang, yang mungkin mengakibatkan sedikit ketidaktepatan.

Catatan penggunaan

Ketika standar deviasi sampel (STDDEV atau STDDEV_SAMP) dihitung untuk ekspresi yang terdiri dari nilai tunggal, hasil fungsinya adalah NULL bukan 0.

Contoh

Kueri berikut mengembalikan rata-rata nilai di kolom VENUESEATS dari tabel VENUE, diikuti oleh standar deviasi sampel dan standar deviasi populasi dari kumpulan nilai yang sama. VENUESEATS adalah kolom INTEGER. Skala hasilnya dikurangi menjadi 2 digit.

```
select avg(venueseats),
cast(stddev_samp(venueseats) as dec(14,2)) stddevsamp,
```

```
cast(stddev_pop(venueseats) as dec(14,2)) stddevpop
from venue;

avg  | stddevsamp | stddevpop
-------+------------+-----------
17503 |   27847.76 |  27773.20
(1 row)
```

Kueri berikut mengembalikan standar deviasi sampel untuk kolom COMMISSION di tabel PENJUALAN. KOMISI adalah kolom DESIMAL. Skala hasilnya dikurangi menjadi 10 digit.

```
select cast(stddev(commission) as dec(18,10))
from sales;

stddev
----------------
130.3912659086
(1 row)
```

Kueri berikut menampilkan standar deviasi sampel untuk kolom COMMISSION sebagai bilangan bulat.

```
select cast(stddev(commission) as integer)
from sales;

stddev
--------
130
(1 row)
```

Kueri berikut mengembalikan standar deviasi sampel dan akar kuadrat dari varians sampel untuk kolom COMMISSION. Hasil perhitungan ini sama.

```
select
cast(stddev_samp(commission) as dec(18,10)) stddevsamp,
cast(sqrt(var_samp(commission)) as dec(18,10)) sqrtvarsamp
from sales;

stddevsamp   |  sqrtvarsamp
----------------+----------------
130.3912659086 | 130.3912659086
```

```
(1 row)
```

Fungsi SUM

Fungsi SUM mengembalikan jumlah kolom input atau nilai ekspresi. Fungsi SUM bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaksis

```
SUM ( [ DISTINCT | ALL ] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ek spresi adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- NUMERIC
- DECIMAL
- REAL
- PRESISI GANDA
- SUPER

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung jumlah. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung jumlah. ALL adalah default.

Jenis Data

Jenis argumen yang didukung oleh fungsi SUM adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, DOUBLE PRECISION, dan SUPER.

Jenis pengembalian yang didukung oleh fungsi SUM adalah

- BIGINT untuk argumen BIGINT, SMALLINT, dan INTEGER
- NUMERIK untuk argumen NUMERIC
- PRESISI GANDA untuk argumen float point
- Mengembalikan tipe data yang sama dengan ekspresi untuk jenis argumen lainnya.

Presisi default untuk hasil fungsi SUM dengan argumen NUMERIC atau DECIMAL adalah 38. Skala hasilnya sama dengan skala argumen. Misalnya, SUM kolom DEC (5,2) mengembalikan tipe data DEC (38,2).

Contoh

Temukan jumlah semua komisi yang dibayarkan dari tabel PENJUALAN:

```
select sum(commission) from sales;

sum
-------------
16614814.65
(1 row)
```

Temukan jumlah kursi di semua tempat di negara bagian Florida:

```
select sum(venueseats) from venue
where venuestate = 'FL';

sum
--------
250411
(1 row)
```

Temukan jumlah kursi yang terjual di bulan Mei:

```
select sum(qtysold) from sales, date
where sales.dateid = date.dateid and date.month = 'MAY';

sum
-------
32291
(1 row)
```

Fungsi VAR_SAMP dan VAR_POP

Fungsi VAR_SAMP dan VAR_POP mengembalikan sampel dan varians populasi dari sekumpulan nilai numerik (integer, desimal, atau floating-point). Hasil fungsi VAR_SAMP setara dengan standar deviasi sampel kuadrat dari set nilai yang sama.

VAR_SAMP dan VARIANCE adalah sinonim untuk fungsi yang sama.

Sintaksis

```
VAR_SAMP | VARIANCE ( [ DISTINCT | ALL ] expression)
VAR_POP ( [ DISTINCT | ALL ] expression)
```

Ekspresi harus memiliki tipe data integer, desimal, atau floating-point. Terlepas dari jenis data ekspresi, tipe pengembalian fungsi ini adalah angka presisi ganda.

> **Note**
>
> Hasil dari fungsi-fungsi ini mungkin bervariasi di seluruh cluster gudang data, tergantung pada konfigurasi cluster dalam setiap kasus.

Catatan penggunaan

Ketika varian sampel (VARIANCE atau VAR_SAMP) dihitung untuk ekspresi yang terdiri dari nilai tunggal, hasil fungsi tersebut adalah NULL bukan 0.

Contoh

Kueri berikut mengembalikan sampel yang dibulatkan dan varians populasi dari kolom NUMTICKETS dalam tabel LISTING.

```
select avg(numtickets),
round(var_samp(numtickets)) varsamp,
round(var_pop(numtickets)) varpop
from listing;

avg | varsamp | varpop
-----+---------+--------
10 |      54 |     54
```

```
(1 row)
```

Kueri berikut menjalankan perhitungan yang sama tetapi memberikan hasilnya ke nilai desimal.

```
select avg(numtickets),
cast(var_samp(numtickets) as dec(10,4)) varsamp,
cast(var_pop(numtickets) as dec(10,4)) varpop
from listing;

avg | varsamp | varpop
-----+---------+---------
10 | 53.6291 | 53.6288
(1 row)
```

Fungsi array

Berikut ini, Anda dapat menemukan deskripsi fungsi array untuk SQL yang didukung Amazon Redshift untuk mengakses dan memanipulasi array.

Topik

- Fungsi ARRAY
- Fungsi ARRAY_CONCAT
- Fungsi ARRAY_CONTAINS
- Fungsi ARRAY_DISTINCT
- Fungsi ARRAY_EXCEPT
- Fungsi ARRAY_FLATTEN
- Fungsi ARRAY_INTERSECTION
- Fungsi ARRAY_POSITION
- Fungsi ARRAY_POSITIONS
- Fungsi ARRAY_SORT
- Fungsi ARRAY_UNION
- Fungsi ARRAYS_OVERLAP
- Fungsi GET_ARRAY_LENGTH
- Fungsi SPLIT_TO_ARRAY
- Fungsi SUBARRAY

Fungsi ARRAY

Membuat array tipe data SUPER.

Sintaksis

```
ARRAY( [ expr1 ] [, expr2 [, ... ]] )
```

Pendapat

expr1, expr2

Ekspresi jenis data Amazon Redshift apa pun kecuali tipe tanggal dan waktu, karena Amazon Redshift tidak mentransmisikan tipe tanggal dan waktu ke tipe data SUPER. Argumen tidak harus dari tipe data yang sama.

Jenis pengembalian

Fungsi ARRAY mengembalikan tipe data SUPER.

Contoh

Contoh berikut menunjukkan array nilai numerik dan array tipe data yang berbeda.

```
--an array of numeric values
select ARRAY(1,50,null,100);
      array
------------------
 [1,50,null,100]
(1 row)

--an array of different data types
select ARRAY(1,'abc',true,3.14);
        array
-----------------------
 [1,"abc",true,3.14]
(1 row)
```

Lihat juga

- Fungsi ARRAY_CONCAT

- Fungsi SPLIT_TO_ARRAY
- Fungsi ARRAY_FLATTEN

Fungsi ARRAY_CONCAT

Gabungkan dua array untuk membuat array yang berisi semua elemen dalam array pertama diikuti oleh semua elemen dalam array kedua. Kedua argumen harus berupa array SUPER yang valid.

Sintaksis

```
ARRAY_CONCAT( array1, array2 )
```

Argumen

susunan1

Nilai yang menentukan yang pertama dari dua array untuk digabungkan.

susunan2

Nilai yang menentukan yang kedua dari dua array untuk digabungkan.

Jenis pengembalian

Fungsi ARRAY_CONCAT mengembalikan nilai data SUPER.

Contoh

Contoh berikut menunjukkan penggabungan dua array dari jenis yang sama dan penggabungan dua array dari jenis yang berbeda.

```
-- concatenating two arrays
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY(10003,10004));
              array_concat
-----------------------------------
 [10001,10002,10003,10004]
(1 row)

-- concatenating two arrays of different types
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY('ab','cd'));
          array_concat
```

```
-----------------------------
 [10001,10002,"ab","cd"]
(1 row)
```

Lihat juga

- Fungsi ARRAY_UNION
- Fungsi ARRAY_FLATTEN
- Fungsi SPLIT_TO_ARRAY
- Fungsi ARRAY_DISTINCT

Fungsi ARRAY_CONTAINS

Memeriksa apakah array berisi nilai yang diberikan dan mengembalikan TRUE jika ditemukan.

Sintaksis

```
ARRAY_CONTAINS( array, value [, null_match] )
```

Argumen

susunan

Ekspresi SUPER yang menentukan array untuk mencari.

nilai

Nilai yang menentukan elemen yang akan dicari.

null_match

Nilai boolean yang menentukan bagaimana nilai NULL ditangani:

- null_match = FALSE: Mencari NULL mengembalikan NULL. Jika array berisi nilai NULL dan tidak ada kecocokan yang ditemukan untuk nilai pencarian non-NULL, mengembalikan NULL.
- null_match = TRUE: NULL diperlakukan sebagai elemen yang valid dan dapat dicari. Jika array berisi nilai NULL dan tidak ada kecocokan yang ditemukan untuk nilai pencarian non-NULL, ia mengembalikan FALSE.

Defaultnya adalah BETUL.

Penanganan NULL default juga dapat ditentukan oleh opsi konfigurasi:

```
-- same as null_match = TRUE
SET default_array_search_null_handling to TRUE;
```

Jenis pengembalian

Fungsi ARRAY_CONTAINS mengembalikan tipe BOOLEAN.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_CONTAINS.

```
SELECT ARRAY_CONTAINS(ARRAY('red', 'green'), 'red');
array_contains
----------------
 t
(1 row)
```

Contoh berikut menunjukkan perilaku fungsi dengan null_match disetel ke TRUE.

```
SET default_array_search_null_handling to TRUE;

-- NULL search is enabled
SELECT ARRAY_CONTAINS(ARRAY('red', NULL, 'green'), NULL);
array_contains
----------------
 t
(1 row)

-- The array can contain NULLs
SELECT ARRAY_CONTAINS(ARRAY('red', NULL, 'green'), 'blue', TRUE);
array_contains
----------------
 f
(1 row)
```

Contoh berikut menunjukkan perilaku fungsi dengan null_match disetel ke FALSE. Perhatikan bahwa menentukan perilaku null_match dalam fungsi akan menggantikan pengaturan konfigurasi default.

```
-- same as null_match = TRUE
SET default_array_search_null_handling to TRUE;
```

```
-- NULL search is disabled. The default behavior is overridden
SELECT ARRAY_CONTAINS(ARRAY('red', 'green'), NULL, FALSE);
array_contains
---------------

(1 row)

-- same as null_match = FALSE
SET default_array_search_null_handling to FALSE;

-- The array contains NULL and a match is found
SELECT ARRAY_CONTAINS(ARRAY('red', NULL, 'green'), 'green');
array_contains
---------------
 t
(1 row)

-- The array contains NULL but no match is found
SELECT ARRAY_CONTAINS(ARRAY('red', NULL, 'green'), 'blue');
array_contains
---------------

(1 row)
```

Lihat juga

- Fungsi ARRAY_POSITION
- Fungsi ARRAY_POSITIONS
- Fungsi ARRAYS_OVERLAP

Fungsi ARRAY_DISTINCT

Membuat array baru yang hanya berisi elemen unik dari array input, menghapus semua duplikat. Urutan elemen dalam array keluaran tidak dijamin cocok dengan urutan input. Nilai NULL diperlakukan sebagai elemen yang valid; jika beberapa NULL ada dalam array input, hanya satu NULL muncul di output.

Sintaksis

```
ARRAY_DISTINCT( array )
```

Argumen

susunan

Ekspresi SUPER yang menentukan array.

Jenis pengembalian

Fungsi ARRAY_DISTINCT mengembalikan tipe SUPER.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_DISTINCT.

```
SELECT ARRAY_DISTINCT(ARRAY(1, 1, 'a', 'a', NULL, NULL));
 array_distinct
----------------
 [1,"a",null]
(1 row)

SELECT ARRAY_DISTINCT(ARRAY_CONCAT(ARRAY(1,2,3,3),ARRAY(2,3,4,4)));
 array_distinct
----------------
 [1,2,3,4]
(1 row)
```

Lihat juga

- Fungsi ARRAY_UNION
- Fungsi ARRAY_SORT
- Fungsi ARRAY_EXCEPT
- Fungsi ARRAY_INTERSECTION

Fungsi ARRAY_EXCEPT

Mengembalikan perbedaan antara dua array dengan menjaga elemen dari array pertama yang tidak ada di array kedua. Fungsinya adalah NULL-safe, artinya memperlakukan NULL diperlakukan sebagai objek yang dikenal.

Sintaksis

```
ARRAY_EXCEPT( array1, array2 [, distinct] )
```

Argumen

susunan1

Ekspresi SUPER yang menentukan array pertama.

susunan2

Ekspresi SUPER yang menentukan array kedua.

berbeda

Nilai boolean yang menentukan apakah akan mengembalikan elemen yang berbeda saja:

- distinct = SALAH: Multi-set semantik berlaku. Setiap kejadian elemen dalam array pertama dicocokkan dengan kejadian di array kedua. Jika array pertama memiliki lebih banyak kemunculan elemen daripada array kedua, kejadian ekstra dipertahankan dalam hasilnya.
- distinct = TRUE: Setel semantik berlaku. Kedua array diperlakukan sebagai set, mengabaikan elemen duplikat. Elemen dari array pertama dihapus jika ada di mana saja di array kedua, terlepas dari jumlah kejadian.

Defaultnya adalah FALSE.

Jenis pengembalian

Fungsi ARRAY_EXCEPT mengembalikan tipe SUPER.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_EXCEPT.

```
SELECT ARRAY_EXCEPT(ARRAY('a','b','c'), ARRAY('b','c','d'));
 array_except
--------------
 ["a"]
(1 row)
```

Multi-set semantik:

```
SELECT ARRAY_EXCEPT(ARRAY('b','b','b','b'), ARRAY('b','b'));
 array_except
--------------
 ["b","b"]
(1 row)
```

Tetapkan semantik:

```
SELECT ARRAY_EXCEPT(ARRAY('a','b','b'), ARRAY('b'), TRUE);
 array_except
--------------
 ["a"]
(1 row)
```

NULL diperlakukan sebagai objek yang dikenal.

```
SELECT ARRAY_EXCEPT(ARRAY('a',NULL), ARRAY(NULL));
 array_except
--------------
 ["a"]
(1 row)
```

Lihat juga

- Fungsi ARRAY_INTERSECTION
- Fungsi ARRAY_UNION
- Fungsi ARRAY_DISTINCT
- Fungsi ARRAYS_OVERLAP

Fungsi ARRAY_FLATTEN

Menggabungkan beberapa array ke dalam satu array tipe SUPER. Elemen dari array dalam pertama muncul pertama, diikuti oleh elemen dari array dalam berikutnya. NULL diperlakukan sebagai objek yang dikenal.

Sintaksis

```
ARRAY_FLATTEN( array )
```

Argumen

susunan

Ekspresi SUPER dari bentuk array.

Jenis pengembalian

Fungsi ARRAY_FLATTEN mengembalikan tipe SUPER.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_FLATTEN.

```
SELECT ARRAY_FLATTEN(ARRAY(ARRAY(1,2,3,4),ARRAY(5,6,7,8),ARRAY(9,10)));
     array_flatten
-----------------------
 [1,2,3,4,5,6,7,8,9,10]
(1 row)
```

Lihat juga

- Fungsi ARRAY_CONCAT
- Fungsi SUBARRAY
- Fungsi ARRAY_DISTINCT

Fungsi ARRAY_INTERSECTION

Mengembalikan array baru yang hanya berisi elemen yang ada di kedua array input. Fungsinya adalah NULL-safe, artinya memperlakukan NULL diperlakukan sebagai objek yang dikenal. Urutan elemen dalam hasil tidak dijamin.

Sintaksis

```
ARRAY_INTERSECTION( array1, array2 [, distinct] )
```

Argumen

susunan1

Ekspresi SUPER yang menentukan array.

susunan2

Ekspresi SUPER yang menentukan array.

berbeda

Nilai boolean yang menentukan apakah akan mengembalikan elemen yang berbeda saja:

- distinct = SALAH: Multi-set semantik berlaku. Elemen duplikat dipertahankan, dan frekuensi setiap elemen dalam hasilnya sama dengan frekuensi minimum dalam dua array input.
- distinct = TRUE: Setel semantik berlaku. Hanya elemen unik yang umum untuk kedua array dikembalikan, tanpa duplikat.

Defaultnya adalah FALSE.

Jenis pengembalian

Fungsi ARRAY_INTERSECTION mengembalikan tipe SUPER.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_INTERSECTION.

```
SELECT ARRAY_INTERSECTION(ARRAY('a','b','c'), ARRAY('b','c','d'));
 array_intersection
--------------------
 ["b","c"]
(1 row)
```

Multi-set semantik:

```
SELECT ARRAY_INTERSECTION(ARRAY('a','b','b'), ARRAY('b','b','b'));
 array_intersection
--------------------
 ["b","b"]
(1 row)
```

Tetapkan semantik:

```
SELECT ARRAY_INTERSECTION(ARRAY('a','b','b'), ARRAY('b','b','b'), TRUE);
```

```
 array_intersection
--------------------
 ["b"]
(1 row)
```

NULL diperlakukan sebagai objek yang dikenal.

```
SELECT ARRAY_INTERSECTION(ARRAY('a',NULL), ARRAY('b',NULL));
 array_intersection
--------------------
 [null]
(1 row)
```

Lihat juga

- Fungsi ARRAY_EXCEPT
- Fungsi ARRAYS_OVERLAP
- Fungsi ARRAY_UNION
- Fungsi ARRAY_DISTINCT

Fungsi ARRAY_POSITION

Mengembalikan posisi (indeks) dari kejadian pertama elemen tertentu dalam array. Indeks berbasis 0, di mana 0 menunjukkan elemen pertama, 1 menunjukkan elemen kedua, dan seterusnya. Mengembalikan -1 jika elemen tidak ditemukan dalam array.

Fungsi hanya mengembalikan posisi kejadian pertama. Untuk menemukan semua kejadian, pertimbangkan untuk menggunakan Fungsi ARRAY_POSITIONS fungsi.

Sintaksis

```
ARRAY_POSITION( array, value [, null_match] )
```

Argumen

susunan

Ekspresi SUPER yang menentukan array untuk mencari.

nilai

Nilai yang menentukan elemen yang akan dicari.

null_match

Nilai boolean yang menentukan bagaimana nilai NULL ditangani:

- null_match = FALSE: Mencari NULL mengembalikan NULL. Jika array berisi nilai NULL dan tidak ada kecocokan yang ditemukan untuk nilai pencarian non-NULL, mengembalikan NULL.
- null_match = TRUE: NULL diperlakukan sebagai elemen yang valid dan dapat dicari. Jika array berisi nilai NULL dan tidak ada kecocokan yang ditemukan untuk nilai pencarian non-NULL, ia mengembalikan -1.

Defaultnya adalah BETUL.

Penanganan NULL default juga dapat ditentukan oleh opsi konfigurasi:

```
-- same as null_match = TRUE
SET default_array_search_null_handling to TRUE;
```

Jenis pengembalian

Fungsi ARRAY_POSITION mengembalikan tipe INT.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_POSITION.

```
SELECT ARRAY_POSITION(ARRAY('red', 'green'), 'red');
 array_position
----------------
              0
(1 row)

SELECT ARRAY_POSITION(ARRAY(1, 2, 3), 4);
 array_position
----------------
             -1
(1 row)

-- only the position of the first occurrence is returned
SELECT ARRAY_POSITION(ARRAY('red', 'green', 'red'), 'red');
```

```
 array_position
----------------
              0
(1 row)
```

Contoh berikut menunjukkan perilaku fungsi dengan null_match disetel ke TRUE.

```
SET default_array_search_null_handling to TRUE;

-- NULL search is enabled
SELECT ARRAY_POSITION(ARRAY('red', NULL, 'green'), NULL);
 array_position
----------------
              1
(1 row)

-- The array can contain NULLs
SELECT ARRAY_POSITION(ARRAY('red', NULL, 'green'), 'blue', TRUE);
 array_position
----------------
             -1
(1 row)
```

Contoh berikut menunjukkan perilaku fungsi dengan null_match disetel ke FALSE. Perhatikan bahwa menentukan perilaku null_match dalam fungsi akan menggantikan pengaturan konfigurasi default.

```
-- same as null_match = TRUE
SET default_array_search_null_handling to TRUE;

-- NULL search is disabled. The default behavior is overridden
SELECT ARRAY_POSITION(ARRAY('red', 'green'), NULL, FALSE);
 array_position
----------------

(1 row)

-- same as null_match = FALSE
SET default_array_search_null_handling to FALSE;

-- The array contains NULL and a match is found
SELECT ARRAY_POSITION(ARRAY('red', NULL, 'green'), 'green');
 array_position
----------------
```

```
              2
(1 row)

-- The array contains NULL but no match is found
SELECT ARRAY_POSITION(ARRAY('red', NULL, 'green'), 'blue');
 array_position
----------------

(1 row)
```

Lihat juga

- Fungsi ARRAY_POSITIONS
- Fungsi ARRAY_CONTAINS
- Fungsi SUBARRAY

Fungsi ARRAY_POSITIONS

Mengembalikan array posisi (indeks) di mana elemen yang ditentukan muncul dalam array input. Indeks berbasis 0, di mana 0 menunjukkan elemen pertama, 1 menunjukkan elemen kedua, dan seterusnya. Mengembalikan array kosong jika elemen tidak ditemukan.

Sintaksis

```
ARRAY_POSITIONS( array, value [, null_match] )
```

Argumen

susunan

Ekspresi SUPER yang menentukan array untuk mencari.

nilai

Nilai yang menentukan elemen yang akan dicari.

null_match

Nilai boolean yang menentukan bagaimana nilai NULL ditangani:

- null_match = FALSE: Mencari NULL mengembalikan NULL. Jika array berisi nilai NULL dan tidak ada kecocokan yang ditemukan untuk nilai pencarian non-NULL, mengembalikan NULL.

- null_match = TRUE: NULL diperlakukan sebagai elemen yang valid dan dapat dicari. Jika array berisi nilai NULL dan tidak ada kecocokan yang ditemukan untuk nilai pencarian non-NULL, ia mengembalikan array kosong.

Defaultnya adalah BETUL.

Penanganan NULL default juga dapat ditentukan oleh opsi konfigurasi:

```
-- same as null_match = TRUE
SET default_array_search_null_handling to TRUE;
```

Jenis pengembalian

Fungsi ARRAY_POSITIONS mengembalikan tipe SUPER.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_POSITIONS.

```
SELECT ARRAY_POSITIONS(ARRAY('red', 'green', 'red'), 'red');
 array_positions
-----------------
 [0,2]
(1 row)

SELECT ARRAY_POSITIONS(ARRAY(1, 2, 3), 4);
 array_positions
-----------------
 []
(1 row)
```

Contoh berikut menunjukkan perilaku fungsi dengan null_match disetel ke TRUE.

```
SET default_array_search_null_handling to TRUE;

-- NULL search is enabled
SELECT ARRAY_POSITIONS(ARRAY('red', NULL, 'green', NULL), NULL);
 array_positions
-----------------
 [1,3]
(1 row)
```

```
-- The array can contain NULLs
SELECT ARRAY_POSITIONS(ARRAY('red', NULL, 'green'), 'blue', TRUE);
 array_positions
-----------------
 []
(1 row)
```

Contoh berikut menunjukkan perilaku fungsi dengan null_match disetel ke FALSE. Perhatikan bahwa menentukan perilaku null_match dalam fungsi akan menggantikan pengaturan konfigurasi default.

```
-- same as null_match = TRUE
SET default_array_search_null_handling to TRUE;

-- NULL search is disabled. The default behavior is overridden
SELECT ARRAY_POSITIONS(ARRAY('red', 'green'), NULL, FALSE);
 array_positions
-----------------

(1 row)

-- same as null_match = FALSE
SET default_array_search_null_handling to FALSE;

-- The array contains NULL and a match is found
SELECT ARRAY_POSITIONS(ARRAY('red', NULL, 'green'), 'green');
 array_positions
-----------------
 [2]
(1 row)

-- The array contains NULL but no match is found
SELECT ARRAY_POSITIONS(ARRAY('red', NULL, 'green'), 'blue');
 array_positions
-----------------

(1 row)
```

Lihat juga

- Fungsi ARRAY_POSITION
- Fungsi ARRAY_CONTAINS

- Fungsi SUBARRAY

Fungsi ARRAY_SORT

Membuat versi array input yang diurutkan dalam urutan naik atau turun. Anda dapat menentukan di mana nilai NULL akan muncul dalam hasil. Fungsinya adalah NULL-safe, artinya memperlakukan NULL diperlakukan sebagai objek yang dikenal.

Sintaksis

```
ARRAY_SORT( array [, sort_ascending [, nulls_first]] )
```

Argumen

susunan

Ekspresi SUPER yang menentukan array yang akan diurutkan.

urutkan_naik

Nilai boolean yang menentukan apakah akan mengurutkan array dalam urutan naik atau turun:

- Tentukan TRUE untuk mengurutkan elemen dalam urutan menaik.
- Tentukan FALSE untuk mengurutkan elemen dalam urutan menurun.

Defaultnya adalah BETUL.

nulls_first

Nilai boolean yang menentukan posisi NULL:

- Tentukan TRUE untuk menempatkan NULL di awal array yang diurutkan.
- Tentukan FALSE untuk menempatkan NULL di akhir array yang diurutkan.

Jenis pengembalian

Fungsi ARRAY_SORT mengembalikan tipe SUPER.

Catatan

Saat menyortir array yang berisi tipe data campuran, elemen diurutkan sesuai dengan prioritas tipe berikut:

- Nilai Boolean
- Nilai numerik
- Nilai string
- Array
- Objects/Dictionaries

Dalam setiap kategori tipe, elemen diurutkan menurut urutan alaminya (misalnya, angka diurutkan secara numerik, string menurut abjad).

Contoh

Contoh berikut menunjukkan fungsi ARRAY_SORT.

```
-- Ascending order (default)
SELECT ARRAY_SORT(ARRAY('b', 'a', 0, NULL, 1, false));
        array_sort
--------------------------
 [false,0,1,"a","b",null]
(1 row)

-- Descending order
SELECT ARRAY_SORT(ARRAY('b', 'a', 0, NULL, 1, false), False);
        array_sort
--------------------------
 [null,"b","a",1,0,false]
(1 row)

-- Descending order with NULLs at the end of the sorted array
SELECT ARRAY_SORT(ARRAY('b', 'a', 0, NULL, 1, false), False, False);
        array_sort
--------------------------
 ["b","a",1,0,false,null]
(1 row)
```

Lihat juga

- Fungsi ARRAY_DISTINCT
- Fungsi ARRAY_FLATTEN
- Fungsi SUBARRAY

Fungsi ARRAY_UNION

Menggabungkan dua array dan mengembalikan array tunggal yang berisi semua nilai unik, menghapus duplikat apa pun. Fungsinya adalah NULL-safe, artinya memperlakukan NULL diperlakukan sebagai objek yang dikenal. Urutan elemen dalam hasil tidak dijamin.

Sintaksis

```
ARRAY_UNION( array1, array2 )
```

Argumen

susunan1

Ekspresi SUPER yang menentukan array pertama.

susunan2

Ekspresi SUPER yang menentukan array kedua.

Jenis pengembalian

Fungsi ARRAY_UNION mengembalikan tipe SUPER.

Contoh

Contoh berikut menunjukkan fungsi ARRAY_UNION.

```
SELECT ARRAY_UNION(ARRAY('a','b','b'), ARRAY('b','c','c'));
  array_union
---------------
 ["a","b","c"]
(1 row)
```

Urutan elemen tidak dijamin:

```
SELECT ARRAY_UNION(ARRAY('b','a','b'), ARRAY(NULL,'b',NULL));
  array_union
----------------
 ["b","a",null]
(1 row)
```

Lihat juga

- Fungsi ARRAY_CONCAT
- Fungsi ARRAY_DISTINCT
- Fungsi ARRAY_INTERSECTION
- Fungsi ARRAY_EXCEPT

Fungsi ARRAYS_OVERLAP

Memeriksa apakah dua array memiliki elemen yang sama. Mengembalikan TRUE jika array berbagi setidaknya satu elemen, atau FALSE jika tidak ada elemen umum yang ada. Fungsinya adalah NULL-safe, artinya memperlakukan NULL diperlakukan sebagai objek yang dikenal.

Sintaksis

```
ARRAYS_OVERLAP( array1, array2 )
```

Argumen

susunan1

Ekspresi SUPER yang menentukan array.

susunan2

Ekspresi SUPER yang menentukan array.

Jenis pengembalian

Fungsi ARRAYS_OVERLAP mengembalikan tipe Boolean.

Contoh

Contoh berikut menunjukkan fungsi ARRAYS_OVERLAP.

```
SELECT ARRAYS_OVERLAP(ARRAY('blue', 'green'), ARRAY('red', 'green'));
 arrays_overlap
----------------
 t
```

```
(1 row)
```

Contoh berikut menunjukkan bahwa NULL diperlakukan sebagai elemen yang valid.

```
SELECT ARRAYS_OVERLAP(ARRAY('red', NULL, 'blue'), ARRAY('green', NULL));
 arrays_overlap
----------------
 t
(1 row)

SELECT ARRAYS_OVERLAP(ARRAY('red', NULL, 'blue'), ARRAY('green'));
 arrays_overlap
----------------
 f
(1 row)

SELECT ARRAYS_OVERLAP(JSON_PARSE('[null]'), ARRAY(NULL));
 arrays_overlap
----------------
 t
(1 row)
```

Lihat juga

- Fungsi ARRAY_INTERSECTION
- Fungsi ARRAY_CONTAINS
- Fungsi ARRAY_EXCEPT

Fungsi GET_ARRAY_LENGTH

Mengembalikan panjang array SUPER yang diberikan objek atau jalur array.

Sintaksis

```
GET_ARRAY_LENGTH( super_expr )
```

Argumen

super_expr

Ekspresi SUPER yang valid dari bentuk array.

Jenis pengembalian

Fungsi GET_ARRAY_LENGTH mengembalikan sebuah INT.

Contoh

Contoh berikut menunjukkan fungsi GET_ARRAY_LENGTH.

```
SELECT GET_ARRAY_LENGTH(ARRAY(1,2,3,4,5,6,7,8,9,10));
 get_array_length
----------------------
            10
(1 row)
```

Fungsi SPLIT_TO_ARRAY

Menggunakan pembatas sebagai parameter opsional. Jika tidak ada pembatas, maka defaultnya adalah koma.

Sintaksis

```
SPLIT_TO_ARRAY( string, delimiter )
```

Argumen

tali

String input yang akan dipecah.

pembatas

Nilai opsional di mana string input akan dibagi. Default adalah koma.

Jenis pengembalian

Fungsi SPLIT_TO_ARRAY mengembalikan nilai data SUPER.

Contoh

Contoh berikut menunjukkan fungsi SPLIT_TO_ARRAY.

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');
     split_to_array
```

```
------------------------
 ["12","345","6789"]
(1 row)
```

Lihat juga

- Fungsi ARRAY
- Fungsi ARRAY_CONCAT
- Fungsi SUBARRAY
- Fungsi ARRAY_FLATTEN

Fungsi SUBARRAY

Mengekstrak sebagian dari array mulai dari posisi tertentu. Mengembalikan array baru yang berisi jumlah elemen tertentu dari array input.

Sintaksis

```
SUBARRAY( super_expr, start_position, length )
```

Argumen

super_expr

Ekspresi SUPER yang valid dalam bentuk array.

posisi awal

Bilangan bulat yang menentukan posisi awal untuk ekstraksi. Indeks berbasis 0, di mana 0 menunjukkan elemen pertama. Jika start_position berada di luar panjang array, array kosong dikembalikan.

panjang

Bilangan bulat opsional yang menentukan jumlah elemen yang akan diekstrak. Jika dihilangkan, semua elemen dari posisi awal hingga akhir array dikembalikan.

Jenis pengembalian

Fungsi SUBARRAY mengembalikan nilai data SUPER.

Contoh

Berikut ini adalah contoh fungsi SUBARRAY.

```
 SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);
   subarray
---------------
 ["c","d","e"]
(1 row)
```

Lihat juga

- Fungsi ARRAY_POSITION
- Fungsi ARRAY_POSITIONS
- Fungsi ARRAY_FLATTEN
- Fungsi ARRAY_CONCAT

Bit-wise fungsi agregat

Bit-wise fungsi agregat menghitung operasi bit untuk melakukan agregasi kolom integer dan kolom yang dapat dikonversi atau dibulatkan ke nilai integer.

Topik

- Menggunakan NULL dalam agregasi bit-wise
- Dukungan DISTINCT untuk agregasi bit-wise
- Contoh ikhtisar untuk fungsi bit-wise
- Fungsi BIT_AND
- Fungsi BIT_OR
- Fungsi BOOL_AND
- Fungsi BOOL_OR

Menggunakan NULL dalam agregasi bit-wise

Saat Anda menerapkan fungsi bit-wise ke kolom yang dapat dibatalkan, nilai NULL apa pun dihilangkan sebelum hasil fungsi dihitung. Jika tidak ada baris yang memenuhi syarat untuk agregasi,

fungsi bit-wise mengembalikan NULL. Perilaku yang sama berlaku untuk fungsi agregat reguler. Berikut adalah contohnya.

```
select sum(venueseats), bit_and(venueseats) from venue
where venueseats is null;

sum  | bit_and
------+---------
null |    null
(1 row)
```

Dukungan DISTINCT untuk agregasi bit-wise

Seperti fungsi agregat lainnya, fungsi bit-wise mendukung kata kunci DISTINCT.

Namun, menggunakan DISTINCT dengan fungsi-fungsi ini tidak berdampak pada hasil. Contoh pertama dari suatu nilai cukup untuk memenuhi operasi AND atau OR bit-wise. Tidak ada bedanya jika nilai duplikat hadir dalam ekspresi yang sedang dievaluasi.

Karena pemrosesan DISTINCT cenderung menimbulkan beberapa overhead eksekusi kueri, kami sarankan Anda tidak menggunakan DISTINCT dengan fungsi bit-wise.

Contoh ikhtisar untuk fungsi bit-wise

Berikut ini, Anda dapat menemukan beberapa contoh ikhtisar yang menunjukkan cara bekerja dengan fungsi bit-wise. Anda juga dapat menemukan contoh kode spesifik dengan setiap deskripsi fungsi.

Contoh untuk fungsi bit-wise didasarkan pada database sampel TICKIT. Tabel USERS dalam database sampel TICKIT berisi beberapa kolom Boolean yang menunjukkan apakah setiap pengguna diketahui menyukai berbagai jenis acara, seperti olahraga, teater, opera, dan sebagainya. Berikut contohnya.

```
select userid, username, lastname, city, state,
likesports, liketheatre
from users limit 10;

userid | username | lastname  |     city     | state | likesports | liketheatre
-------+----------+-----------+--------------+-------+------------+-------------
1 | JSG99FHE | Taylor    | Kent         | WA    | t          | t
9 | MSD36KVR | Watkins   | Port Orford  | MD    | t          | f
```

Asumsikan bahwa versi baru dari tabel USERS dibangun dengan cara yang berbeda. Dalam versi baru ini, kolom integer tunggal yang mendefinisikan (dalam bentuk biner) delapan jenis peristiwa yang disukai atau tidak disukai setiap pengguna. Dalam desain ini, setiap posisi bit mewakili jenis peristiwa. Pengguna yang menyukai semua delapan jenis memiliki semua delapan bit diatur ke 1 (seperti pada baris pertama tabel berikut). Pengguna yang tidak menyukai salah satu dari peristiwa ini memiliki semua delapan bit disetel ke 0 (lihat baris kedua). Pengguna yang hanya menyukai olahraga dan jazz diwakili di baris ketiga berikut.

USER	OLAHRA(	TEATER	JAZZ	OPERA	BATU	VEGAS	BROADW	KLASIK
Pengguna 1	1	1	1	1	1	1	1	1
Pengguna 2	0	0	0	0	0	0	0	0
Pengguna 3	1	0	1	0	0	0	0	0

Dalam tabel database, nilai-nilai biner ini dapat disimpan dalam kolom LIKES tunggal sebagai bilangan bulat, seperti yang ditunjukkan berikut.

Pengguna	Nilai biner	Nilai yang disimpan (integer)
Pengguna 1	111111111	255
Pengguna 2	00000000	0
Pengguna 3	10100000	160

Fungsi BIT_AND

Fungsi BIT_AND menjalankan operasi AND bit-wise pada semua nilai dalam kolom atau ekspresi integer tunggal. Fungsi ini menggabungkan setiap bit dari setiap nilai biner yang sesuai dengan setiap nilai integer dalam ekspresi.

Fungsi BIT_AND mengembalikan hasil `0` jika tidak ada bit yang diatur ke 1 di semua nilai. Jika satu atau lebih bit diatur ke 1 di semua nilai, fungsi mengembalikan nilai integer. Bilangan bulat ini adalah angka yang sesuai dengan nilai biner untuk bit tersebut.

Misalnya, tabel berisi empat nilai integer dalam kolom: 3, 7, 10, dan 22. Bilangan bulat ini direpresentasikan dalam bentuk biner sebagai berikut:

Bilangan Bulat	Nilai biner
3	11
7	111
10	1010
22	10110

Operasi BIT_AND pada dataset ini menemukan bahwa semua bit disetel ke posisi kedua 1 hingga terakhir saja. Hasilnya adalah nilai biner`00000010`, yang mewakili nilai integer2. Oleh karena itu, fungsi BIT_AND kembali. 2

Sintaksis

```
BIT_AND ( [DISTINCT | ALL] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi ini harus memiliki tipe data INT, INT2, atau INT8. Fungsi mengembalikan tipe data INT, INT2, atau INT8 yang setara.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat untuk ekspresi yang ditentukan sebelum menghitung hasilnya. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat. ALL adalah default. Untuk informasi selengkapnya, lihat Dukungan DISTINCT untuk agregasi bit-wise.

Contoh

Mengingat bahwa informasi bisnis yang berarti disimpan dalam kolom integer, Anda dapat menggunakan fungsi bit-wise untuk mengekstrak dan mengumpulkan informasi itu. Kueri berikut menerapkan fungsi BIT_AND ke kolom LIKES dalam tabel yang disebut USERLIKES dan mengelompokkan hasil berdasarkan kolom CITY.

```
select city, bit_and(likes) from userlikes group by city
order by city;
city          | bit_and
--------------+---------
Los Angeles   |       0
Sacramento    |       0
San Francisco |       0
San Jose      |      64
Santa Barbara |     192
(5 rows)
```

Anda dapat menafsirkan hasil ini sebagai berikut:

- Nilai integer `192` untuk Santa Barbara diterjemahkan ke nilai biner`11000000`. Dengan kata lain, semua pengguna di kota ini menyukai olahraga dan teater, tetapi tidak semua pengguna menyukai jenis acara lainnya.
- Bilangan bulat `64` diterjemahkan menjadi`01000000`. Jadi, bagi pengguna di San Jose, satu-satunya jenis acara yang mereka sukai adalah teater.
- Nilai `0` untuk tiga kota lainnya menunjukkan bahwa tidak ada “suka” yang dibagikan oleh semua pengguna di kota-kota tersebut.

Fungsi BIT_OR

Fungsi BIT_OR menjalankan operasi OR bit-wise pada semua nilai dalam kolom atau ekspresi integer tunggal. Fungsi ini menggabungkan setiap bit dari setiap nilai biner yang sesuai dengan setiap nilai integer dalam ekspresi.

Misalnya, misalkan tabel Anda berisi empat nilai integer dalam kolom: 3, 7, 10, dan 22. Bilangan bulat ini direpresentasikan dalam bentuk biner sebagai berikut.

Bilangan Bulat	Nilai biner
3	11
7	111
10	1010
22	10110

Jika Anda menerapkan fungsi BIT_OR ke himpunan nilai integer, operasi mencari nilai apa pun di mana a 1 ditemukan di setiap posisi. Dalam hal ini, 1 a ada di lima posisi terakhir untuk setidaknya satu nilai, menghasilkan hasil biner dari`00011111`; oleh karena itu, fungsi mengembalikan `31` (atau`16 + 8 + 4 + 2 + 1`).

Sintaksis

```
BIT_OR ( [DISTINCT | ALL] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi ini harus memiliki tipe data INT, INT2, atau INT8. Fungsi mengembalikan tipe data INT, INT2, atau INT8 yang setara.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat untuk ekspresi yang ditentukan sebelum menghitung hasilnya. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat. ALL adalah default. Untuk informasi selengkapnya, lihat Dukungan DISTINCT untuk agregasi bit-wise.

Contoh

Kueri berikut menerapkan fungsi BIT_OR ke kolom LIKES dalam tabel yang disebut USERLIKES dan mengelompokkan hasil berdasarkan kolom CITY.

```
select city, bit_or(likes) from userlikes group by city
```

```
order by city;
city          | bit_or
--------------+--------
Los Angeles   |    127
Sacramento    |    255
San Francisco |    255
San Jose      |    255
Santa Barbara |    255
(5 rows)
```

Untuk empat kota yang terdaftar, semua jenis acara disukai oleh setidaknya satu pengguna (`255=11111111`). Untuk Los Angeles, semua jenis acara kecuali olahraga disukai oleh setidaknya satu pengguna (`127=01111111`).

Fungsi BOOL_AND

Fungsi BOOL_AND beroperasi pada kolom atau ekspresi Boolean atau integer tunggal. Fungsi ini menerapkan logika serupa untuk fungsi BIT_AND dan BIT_OR. Untuk fungsi ini, tipe pengembalian adalah nilai Boolean (`true`atau`false`).

Jika semua nilai dalam satu set benar, fungsi BOOL_AND mengembalikan `true` (). `t` Jika ada nilai false, fungsi mengembalikan `false` (`f`).

Sintaksis

```
BOOL_AND ( [DISTINCT | ALL] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi ini harus memiliki tipe data BOOLEAN atau integer. Jenis pengembalian fungsi adalah BOOLEAN.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat untuk ekspresi yang ditentukan sebelum menghitung hasilnya. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat. ALL adalah default. Untuk informasi selengkapnya, lihat Dukungan DISTINCT untuk agregasi bit-wise.

Contoh

Anda dapat menggunakan fungsi Boolean terhadap ekspresi Boolean atau ekspresi integer. Misalnya, hasil query berikut dari tabel USERS standar dalam database TICKIT, yang memiliki beberapa kolom Boolean.

Fungsi BOOL_AND mengembalikan `false` untuk semua lima baris. Tidak semua pengguna di masing-masing negara bagian menyukai olahraga.

```
select state, bool_and(likesports) from users
group by state order by state limit 5;

state | bool_and
------+---------
AB    | f
AK    | f
AL    | f
AZ    | f
BC    | f
(5 rows)
```

Fungsi BOOL_OR

Fungsi BOOL_OR beroperasi pada kolom atau ekspresi Boolean atau integer tunggal. Fungsi ini menerapkan logika serupa untuk fungsi BIT_AND dan BIT_OR. Untuk fungsi ini, tipe pengembalian adalah nilai Boolean (`true`,`false`, atau`NULL`).

Jika satu atau lebih nilai dalam satu set adalah`true`, fungsi BOOL_OR mengembalikan `true` (). `t` Jika semua nilai dalam satu himpunan adalah`false`, fungsi mengembalikan `false` (`f`). NULL dapat dikembalikan jika nilainya tidak diketahui.

Sintaksis

```
BOOL_OR ( [DISTINCT | ALL] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi ini harus memiliki tipe data BOOLEAN atau integer. Jenis pengembalian fungsi adalah BOOLEAN.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat untuk ekspresi yang ditentukan sebelum menghitung hasilnya. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat. ALL adalah default. Lihat Dukungan DISTINCT untuk agregasi bit-wise.

Contoh

Anda dapat menggunakan fungsi Boolean dengan ekspresi Boolean atau ekspresi integer. Misalnya, hasil query berikut dari tabel USERS standar dalam database TICKIT, yang memiliki beberapa kolom Boolean.

Fungsi BOOL_OR mengembalikan `true` untuk semua lima baris. Setidaknya satu pengguna di masing-masing negara bagian menyukai olahraga.

```
select state, bool_or(likesports) from users
group by state order by state limit 5;

state | bool_or
------+--------
AB    | t
AK    | t
AL    | t
AZ    | t
BC    | t
(5 rows)
```

Contoh berikut mengembalikan NULL.

```
SELECT BOOL_OR(NULL = '123')
               bool_or
------
NULL
```

Ekspresi bersyarat

Topik

- Ekspresi bersyarat CASE
- Fungsi DECODE
- Fungsi TERBESAR dan TERKECIL

- Fungsi NVL dan COALESCE
- Fungsi NVL2
- Fungsi NULLIF

Amazon Redshift mendukung beberapa ekspresi kondisional yang merupakan ekstensi standar SQL.

Ekspresi bersyarat CASE

Ekspresi CASE adalah ekspresi kondisional, mirip dengan pernyataan if/then /else yang ditemukan dalam bahasa lain. CASE digunakan untuk menentukan hasil jika terdapat beberapa kondisi. Gunakan CASE di mana ekspresi SQL valid, seperti dalam perintah SELECT.

Ada dua jenis ekspresi CASE: sederhana dan dicari.

- Dalam ekspresi CASE sederhana, ekspresi dibandingkan dengan nilai. Ketika kecocokan ditemukan, tindakan yang ditentukan dalam klausul THEN diterapkan. Jika tidak ada kecocokan ditemukan, tindakan dalam klausul ELSE diterapkan.
- Dalam ekspresi CASE yang dicari, setiap CASE dievaluasi berdasarkan ekspresi Boolean, dan pernyataan CASE mengembalikan CASE yang cocok pertama. Jika tidak ada kecocokan yang ditemukan di antara klausa WHEN, tindakan dalam klausa ELSE dikembalikan.

Sintaksis

Pernyataan CASE sederhana yang digunakan untuk menyesuaikan kondisi:

```
CASE expression
  WHEN value THEN result
  [WHEN...]
  [ELSE result]
END
```

Pernyataan CASE yang dicari digunakan untuk mengevaluasi setiap kondisi:

```
CASE
  WHEN condition THEN result
  [WHEN ...]
  [ELSE result]
END
```

Argumen

ekspresi

Nama kolom atau ekspresi yang valid.

nilai

Nilai yang dibandingkan dengan ekspresi, seperti konstanta numerik atau string karakter.

hasil

Nilai target atau ekspresi yang dikembalikan ketika ekspresi atau kondisi Boolean dievaluasi. Tipe data dari semua ekspresi hasil harus dapat dikonversi ke tipe keluaran tunggal.

ketentuan

Ekspresi Boolean yang mengevaluasi menjadi benar atau salah. Jika kondisi benar, nilai ekspresi CASE adalah hasil yang mengikuti kondisi, dan sisa ekspresi CASE tidak diproses. Jika kondisi salah, klausa WHEN berikutnya dievaluasi. Jika tidak ada hasil kondisi WHEN yang benar, nilai ekspresi CASE adalah hasil dari klausa ELSE. Jika klausa ELSE dihilangkan dan tidak ada kondisi yang benar, hasilnya adalah null.

Contoh

Contoh berikut menggunakan tabel VENUE dan tabel SALES dari sampel data TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Gunakan ekspresi CASE sederhana untuk diganti `New York City` dengan `Big Apple` dalam kueri terhadap tabel VENUE. Ganti semua nama kota lainnya dengan `other`.

```
select venuecity,
  case venuecity
    when 'New York City'
    then 'Big Apple' else 'other'
  end
from venue
order by venueid desc;

venuecity       |   case
----------------+-----------
Los Angeles     | other
New York City   | Big Apple
```

```
San Francisco    | other
Baltimore        | other
...
```

Gunakan ekspresi CASE yang dicari untuk menetapkan nomor grup berdasarkan nilai PRICEPAID untuk penjualan tiket individu:

```
select pricepaid,
  case when pricepaid <10000 then 'group 1'
    when pricepaid >10000 then 'group 2'
    else 'group 3'
  end
from sales
order by 1 desc;

pricepaid |  case
----------+---------
12624     | group 2
10000     | group 3
10000     | group 3
9996      | group 1
9988      | group 1
...
```

Fungsi DECODE

Ekspresi DECODE menggantikan nilai tertentu dengan nilai spesifik lain atau nilai default, tergantung pada hasil dari kondisi kesetaraan. Operasi ini setara dengan operasi ekspresi CASE sederhana atau IF-THEN-ELSE pernyataan.

Sintaksis

```
DECODE ( expression, search, result [, search, result ]... [ ,default ] )
```

Jenis ekspresi ini berguna untuk mengganti singkatan atau kode yang disimpan dalam tabel dengan nilai bisnis bermakna yang diperlukan untuk laporan.

Parameter

ekspresi

Sumber nilai yang ingin Anda bandingkan, seperti kolom dalam tabel.

pencarian

Nilai target yang dibandingkan dengan ekspresi sumber, seperti nilai numerik atau string karakter. Ekspresi pencarian harus mengevaluasi ke nilai tetap tunggal. Anda tidak dapat menentukan ekspresi yang mengevaluasi rentang nilai, sepertiage between 20 and 29; Anda perlu menentukan search/result pasangan terpisah untuk setiap nilai yang ingin Anda ganti.

Tipe data dari semua instance ekspresi pencarian harus sama atau kompatibel. Param eter ekspresi dan pencarian juga harus kompatibel.

hasil

Nilai pengganti yang dikembalikan kueri saat ekspresi cocok dengan nilai pencarian. Anda harus menyertakan setidaknya satu search/result pasangan dalam ekspresi DECODE.

Tipe data dari semua instance ekspresi hasil harus sama atau kompatibel. Param eter hasil dan default juga harus kompatibel.

default

Nilai default opsional yang digunakan untuk kasus ketika kondisi pencarian gagal. Jika Anda tidak menentukan nilai default, ekspresi DECODE mengembalikan NULL.

Catatan penggunaan

Jika nilai ekspresi dan nilai pencarian keduanya NULL, hasil DECODE adalah nilai hasil yang sesuai. Untuk ilustrasi penggunaan fungsi ini, lihat bagian Contoh.

Ketika digunakan dengan cara ini, DECODE mirip denganFungsi NVL2, tetapi ada beberapa perbedaan. Untuk deskripsi perbedaan ini, lihat catatan penggunaan NVL2.

Contoh

Ketika nilai 2008-06-01 ada di kolom caldate dari datetable, contoh berikut menggantikannya dengan. June 1st, 2008 Contoh menggantikan semua nilai caldate lainnya dengan NULL.

```
select decode(caldate, '2008-06-01', 'June 1st, 2008')
from datetable where month='JUN' order by caldate;

case
----------------
```

```
June 1st, 2008

...
(30 rows)
```

Contoh berikut menggunakan ekspresi DECODE untuk mengubah lima kolom CATNAME yang disingkat dalam tabel CATEGORY menjadi nama lengkap dan mengonversi nilai lain di kolom menjadi. Unknown

```
select catid, decode(catname,
'NHL', 'National Hockey League',
'MLB', 'Major League Baseball',
'MLS', 'Major League Soccer',
'NFL', 'National Football League',
'NBA', 'National Basketball Association',
'Unknown')
from category
order by catid;

catid  | case
-------+---------------------------------
1      | Major League Baseball
2      | National Hockey League
3      | National Football League
4      | National Basketball Association
5      | Major League Soccer
6      | Unknown
7      | Unknown
8      | Unknown
9      | Unknown
10     | Unknown
11     | Unknown
(11 rows)
```

Gunakan ekspresi DECODE untuk menemukan tempat di Colorado dan Nevada dengan NULL di kolom VENUESEATS; ubah NULL menjadi nol. Jika kolom VENUESEATS bukan NULL, kembalikan 1 sebagai hasilnya.

```
select venuename, venuestate, decode(venueseats,null,0,1)
from venue
where venuestate in('NV','CO')
order by 2,3,1;
```

```
venuename                | venuestate     | case
-----------------------------+----------------+-----------
Coors Field                  | CO         |   1
Dick's Sporting Goods Park   | CO         |   1
Ellie Caulkins Opera House   | CO         |   1
INVESCO Field        | CO         |   1
Pepsi Center         | CO         |   1
Ballys Hotel         | NV         |   0
Bellagio Hotel               | NV         |   0
Caesars Palace               | NV         |   0
Harrahs Hotel                | NV         |   0
Hilton Hotel                 | NV         |   0
...
(20 rows)
```

Fungsi TERBESAR dan TERKECIL

Mengembalikan nilai terbesar atau terkecil dari daftar sejumlah ekspresi.

Sintaksis

```
GREATEST (value [, ...])
LEAST (value [, ...])
```

Parameter

daftar ekspresi

Daftar ekspresi yang dipisahkan koma, seperti nama kolom. Semua ekspresi harus dapat dikonversi ke tipe data umum. Nilai NULL dalam daftar diabaikan. Jika semua ekspresi dievaluasi ke NULL, hasilnya adalah NULL.

Pengembalian

Mengembalikan nilai terbesar (untuk GREATEST) atau terkecil (untuk PALING SEDIKIT) dari daftar ekspresi yang disediakan.

Contoh

Contoh berikut mengembalikan nilai tertinggi menurut abjad untuk `firstname` atau`lastname`.

```
select firstname, lastname, greatest(firstname,lastname) from users
where userid < 10
order by 3;

 firstname | lastname  | greatest
-----------+-----------+-----------
 Lars      | Ratliff   | Ratliff
 Reagan    | Hodge     | Reagan
 Colton    | Roy       | Roy
 Barry     | Roy       | Roy
 Tamekah   | Juarez    | Tamekah
 Rafael    | Taylor    | Taylor
 Victor    | Hernandez | Victor
 Vladimir  | Humphrey  | Vladimir
 Mufutau   | Watkins   | Watkins
(9 rows)
```

Fungsi NVL dan COALESCE

Mengembalikan nilai ekspresi pertama yang tidak nol dalam serangkaian ekspresi. Ketika nilai non-null ditemukan, ekspresi yang tersisa dalam daftar tidak dievaluasi.

NVL identik dengan COALESCE. Mereka adalah sinonim. Topik ini menjelaskan sintaks dan berisi contoh untuk keduanya.

Sintaksis

```
NVL( expression, expression, ... )
```

Sintaks untuk COALESCE adalah sama:

```
COALESCE( expression, expression, ... )
```

Jika semua ekspresi null, hasilnya adalah null.

Fungsi-fungsi ini berguna ketika Anda ingin mengembalikan nilai sekunder ketika nilai primer hilang atau nol. Misalnya, kueri mungkin mengembalikan nomor telepon pertama dari tiga nomor telepon yang tersedia: seluler, rumah, atau kantor. Urutan ekspresi dalam fungsi menentukan urutan evaluasi.

Argumen

ekspresi

Sebuah ekspresi, seperti nama kolom, untuk dievaluasi untuk status null.

Jenis pengembalian

Amazon Redshift menentukan tipe data dari nilai yang dikembalikan berdasarkan ekspresi masukan. Jika tipe data dari ekspresi input tidak memiliki tipe umum, maka kesalahan dikembalikan.

Contoh

Jika daftar berisi ekspresi integer, fungsi mengembalikan bilangan bulat.

```
SELECT COALESCE(NULL, 12, NULL);

coalesce
--------------
12
```

Contoh ini, yang sama dengan contoh sebelumnya, kecuali bahwa ia menggunakan NVL, mengembalikan hasil yang sama.

```
SELECT NVL(NULL, 12, NULL);

coalesce
--------------
12
```

Contoh berikut mengembalikan tipe string.

```
SELECT COALESCE(NULL, 'Amazon Redshift', NULL);

coalesce
--------------
Amazon Redshift
```

Contoh berikut menghasilkan kesalahan karena tipe data bervariasi dalam daftar ekspresi. Dalam hal ini, ada tipe string dan tipe angka dalam daftar.

```
SELECT COALESCE(NULL, 'Amazon Redshift', 12);
ERROR: invalid input syntax for integer: "Amazon Redshift"
```

Untuk contoh ini, Anda membuat tabel dengan kolom START_DATE dan END_DATE, menyisipkan baris yang menyertakan nilai null, lalu menerapkan ekspresi NVL ke dua kolom.

```
create table datetable (start_date date, end_date date);
insert into datetable values ('2008-06-01','2008-12-31');
insert into datetable values (null,'2008-12-31');
insert into datetable values ('2008-12-31',null);
```

```
select nvl(start_date, end_date)
from datetable
order by 1;

coalesce
------------
2008-06-01
2008-12-31
2008-12-31
```

Nama kolom default untuk ekspresi NVL adalah COALESCE. Kueri berikut mengembalikan hasil yang sama:

```
select coalesce(start_date, end_date)
from datetable
order by 1;
```

Untuk contoh kueri berikut, Anda membuat tabel dengan contoh informasi pemesanan hotel dan memasukkan beberapa baris. Beberapa catatan berisi nilai null.

```
create table booking_info (booking_id int, booking_code character(8), check_in date,
 check_out date, funds_collected numeric(12,2));
```

Masukkan data sampel berikut. Beberapa catatan tidak memiliki `check_out` tanggal atau `funds_collected` jumlah.

```
insert into booking_info values (1, 'OCEAN_WV', '2023-02-01','2023-02-03',100.00);
insert into booking_info values (2, 'OCEAN_WV', '2023-04-22','2023-04-26',120.00);
insert into booking_info values (3, 'DSRT_SUN', '2023-03-13','2023-03-16',125.00);
```

```
insert into booking_info values (4, 'DSRT_SUN', '2023-06-01','2023-06-03',140.00);
insert into booking_info values (5, 'DSRT_SUN', '2023-07-10',null,null);
insert into booking_info values (6, 'OCEAN_WV', '2023-08-15',null,null);
```

Kueri berikut mengembalikan daftar tanggal. Jika `check_out` tanggal tidak tersedia, itu mencantumkan `check_in` tanggal.

```
select coalesce(check_out, check_in)
from booking_info
order by booking_id;
```

Hasilnya adalah sebagai berikut. Perhatikan bahwa dua catatan terakhir menunjukkan `check_in` tanggal.

```
coalesce
------------
2023-02-03
2023-04-26
2023-03-16
2023-06-03
2023-07-10
2023-08-15
```

Jika Anda mengharapkan kueri mengembalikan nilai null untuk fungsi atau kolom tertentu, Anda dapat menggunakan ekspresi NVL untuk mengganti nol dengan beberapa nilai lainnya. Misalnya, fungsi agregat, seperti SUM, mengembalikan nilai nol alih-alih nol ketika mereka tidak memiliki baris untuk dievaluasi. Anda dapat menggunakan ekspresi NVL untuk mengganti nilai null ini dengan`700.0`. Alih-alih`485`, hasil penjumlahan `funds_collected` adalah `1885` karena dua baris yang memiliki nol diganti dengan`700`.

```
select sum(nvl(funds_collected, 700.0)) as sumresult from booking_info;

sumresult
------
 1885
```

Fungsi NVL2

Mengembalikan salah satu dari dua nilai berdasarkan apakah ekspresi tertentu mengevaluasi ke NULL atau NOT NULL.

Sintaksis

```
NVL2 ( expression, not_null_return_value, null_return_value )
```

Argumen

ekspresi

Sebuah ekspresi, seperti nama kolom, untuk dievaluasi untuk status null.

tidak_null_nilai_pengembalian

Nilai dikembalikan jika ekspresi di evaluasi menjadi NOT NULL. Nilai not_null_return_ value harus memiliki tipe data yang sama dengan ekspresi atau secara implisit dapat dikonversi ke tipe data tersebut.

null_return_value

Nilai dikembalikan jika ekspresi di evaluasi ke NULL. Nilai null_return_ value harus memiliki tipe data yang sama dengan ekspresi atau secara implisit dapat dikonversi ke tipe data tersebut.

Jenis pengembalian

Jenis pengembalian NVL2 ditentukan sebagai berikut:

- Jika not_null_return_value atau null_return_value adalah null, tipe data ekspresi not-null dikem balikan.

Jika not_null_return_value dan null_return_value tidak null:

- Jika not_null_return_value dan null_return_value memiliki tipe data yang sama, tipe data tersebut dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data numerik yang berbeda, tipe data numerik terkecil yang kompatibel dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data datetime yang berbeda, tipe data stempel waktu dikem balikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data karakter yang berbeda, tipe data not _null_return_value dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data num erik dan non-numerik campuran, tipe data not_null_return_value dikembalikan.

> **Important**
>
> Dalam dua kasus terakhir di mana tipe data not_null_return_value dikembalikan, null_return_value secara implisit ditransfer ke tipe data tersebut. Jika tipe data tidak kompatibel, fungsi gagal.

Catatan penggunaan

Fungsi DECODEdapat digunakan dengan cara yang mirip dengan NVL2 ketika parameter ekspresi dan pencarian keduanya nol. Perbedaannya adalah bahwa untuk DECODE, pengembalian akan memiliki nilai dan tipe data dari parameter hasil. Sebaliknya, untuk NVL2, pengembalian akan memiliki nilai parameter not_null_return_value atau null_return_value, mana saja yang dipilih oleh fungsi, tetapi akan memiliki tipe data not _null_return_value.

Misalnya, dengan asumsi column1 adalah NULL, kueri berikut akan mengembalikan nilai yang sama. Namun, tipe data nilai pengembalian DECODE akan menjadi INTEGER dan tipe data nilai pengembalian NVL2 adalah VARCHAR.

```
select decode(column1, null, 1234, '2345');
select nvl2(column1, '2345', 1234);
```

Contoh

Contoh berikut memodifikasi beberapa data sampel, kemudian mengevaluasi dua bidang untuk memberikan informasi kontak yang sesuai bagi pengguna:

```
update users set email = null where firstname = 'Aphrodite' and lastname = 'Acevedo';

select (firstname + ' ' + lastname) as name,
nvl2(email, email, phone) AS contact_info
from users
where state = 'WA'
and lastname  like 'A%'
order by lastname, firstname;

name        contact_info
--------------------+--------------------------------------------
Aphrodite Acevedo (906) 632-4407
Caldwell Acevedo  Nunc.sollicitudin@Duisac.ca
```

```
Quinn Adams    vel@adipiscingligulaAenean.com
Kamal Aguilar  quis@vulputaterisusa.com
Samson Alexander  hendrerit.neque@indolorFusce.ca
Hall Alford    ac.mattis@vitaediamProin.edu
Lane Allen     et.netus@risusDonec.org
Xander Allison    ac.facilisis.facilisis@Infaucibus.com
Amaya Alvarado    dui.nec.tempus@eudui.edu
Vera Alvarez   at.arcu.Vestibulum@pellentesque.edu
Yetta Anthony  enim.sit@risus.org
Violet Arnold  ad.litora@at.com
August Ashley  consectetuer.euismod@Phasellus.com
Karyn Austin   ipsum.primis.in@Maurisblanditenim.org
Lucas Ayers    at@elitpretiumet.com
```

Fungsi NULLIF

Sintaksis

Ekspresi NULLIF membandingkan dua argumen dan mengembalikan null jika argumennya sama. Jika tidak sama, argumen pertama dikembalikan. Ekspresi ini adalah kebalikan dari ekspresi NVL atau COALESCE.

```
NULLIF ( expression1, expression2 )
```

Argumen

ekspresi1, ekspresi2

Kolom target atau ekspresi yang dibandingkan. Jenis pengembalian sama dengan tipe ekspresi pertama. Nama kolom default dari hasil NULLIF adalah nama kolom dari ekspresi pertama.

Contoh

Dalam contoh berikut, query mengembalikan string `first` karena argumennya tidak sama.

```
SELECT NULLIF('first', 'second');

case
-------
first
```

Dalam contoh berikut, query kembali NULL karena argumen literal string sama.

```
SELECT NULLIF('first', 'first');

case
-------
NULL
```

Dalam contoh berikut, query kembali 1 karena argumen integer tidak sama.

```
SELECT NULLIF(1, 2);

case
-------
1
```

Dalam contoh berikut, query kembali NULL karena argumen integer sama.

```
SELECT NULLIF(1, 1);

case
-------
NULL
```

Dalam contoh berikut, query mengembalikan null ketika nilai LISTID dan SALESID cocok:

```
select nullif(listid,salesid), salesid
from sales where salesid<10 order by 1, 2 desc;

listid  | salesid
--------+---------
     4  |       2
     5  |       4
     5  |       3
     6  |       5
     10 |       9
     10 |       8
     10 |       7
     10 |       6
        |       1
(9 rows)
```

Anda dapat menggunakan NULLIF untuk memastikan bahwa string kosong selalu dikembalikan sebagai nol. Dalam contoh di bawah ini, ekspresi NULLIF mengembalikan nilai null atau string yang berisi setidaknya satu karakter.

```
insert into category
values(0,'','Special','Special');

select nullif(catgroup,'') from category
where catdesc='Special';

catgroup
----------
null
(1 row)
```

NULLIF mengabaikan bagian akhir yang kosong. Jika string tidak kosong tetapi berisi kosong, NULLIF masih mengembalikan null:

```
create table nulliftest(c1 char(2), c2 char(2));

insert into nulliftest values ('a','a ');

insert into nulliftest values ('b','b');


select nullif(c1,c2) from nulliftest;
c1
------
null
null
(2 rows)
```

Fungsi pemformatan tipe data

Topik

- Fungsi CAST
- Fungsi CONVERT
- TEKS_KE_INT_ALT
- TEKS_KE_NUMERIK_ALT

- TO_CHAR
- Fungsi TO_DATE
- TO_NOMOR
- Fungsi TRY_CAST
- String format tanggal-waktu
- String format numerik
- Teradata-style memformat karakter untuk data numerik

Fungsi pemformatan tipe data menyediakan cara mudah untuk mengonversi nilai dari satu tipe data ke tipe data lainnya. Untuk masing-masing fungsi ini, argumen pertama selalu nilai yang akan diformat dan argumen kedua berisi template untuk format baru. Amazon Redshift mendukung beberapa fungsi pemformatan tipe data.

Fungsi CAST

Fungsi CAST mengonversi satu tipe data ke tipe data lain yang kompatibel. Misalnya, Anda dapat mengonversi string ke tanggal, atau tipe numerik ke string. CAST melakukan konversi runtime, yang berarti konversi tidak mengubah tipe data nilai dalam tabel sumber. Itu hanya diubah dalam konteks kueri.

Fungsi CAST sangat mirip denganthe section called “MENGUBAH”, karena keduanya mengubah satu tipe data ke yang lain, tetapi mereka disebut secara berbeda.

Tipe data tertentu memerlukan konversi eksplisit ke tipe data lain menggunakan fungsi CAST atau CONVERT. Jenis data lain dapat dikonversi secara implisit, sebagai bagian dari perintah lain, tanpa menggunakan CAST atau CONVERT. Lihat Kompatibilitas dan konversi tipe.

Sintaksis

Gunakan salah satu dari dua bentuk sintaks yang setara ini untuk mentransmisikan ekspresi dari satu tipe data ke tipe data lainnya.

```
CAST ( expression AS type )
expression :: type
```

Argumen

ekspresi

Ekspresi yang mengevaluasi satu atau lebih nilai, seperti nama kolom atau literal. Mengkonversi nilai null mengembalikan nol. Ekspresi tidak dapat berisi string kosong atau kosong.

jenis

Salah satu yang didukungJenis Data.

Jenis pengembalian

CAST mengembalikan tipe data yang ditentukan oleh argum en tipe.

> **Note**
>
> Amazon Redshift mengembalikan kesalahan jika Anda mencoba melakukan konversi yang bermasalah, seperti konversi DECIMAL yang kehilangan presisi, seperti berikut ini:
>
> ```
> select 123.456::decimal(2,1);
> ```
>
> atau konversi INTEGER yang menyebabkan luapan:
>
> ```
> select 12345678::smallint;
> ```

Contoh

Beberapa contoh menggunakan contoh database https://docs.aws.amazon.com/redshift/latest/dg/c_sampledb.html TICKIT. Untuk informasi selengkapnya tentang menyiapkan data sampel, lihat Mem uat data.

Dua query berikut ini setara. Mereka berdua memberikan nilai desimal ke bilangan bulat:

```
select cast(pricepaid as integer)
from sales where salesid=100;

pricepaid
-----------
162
```

```
(1 row)
```

```
select pricepaid::integer
from sales where salesid=100;

pricepaid
-----------
162
(1 row)
```

Berikut ini menghasilkan hasil yang serupa. Tidak memerlukan data sampel untuk menjalankan:

```
select cast(162.00 as integer) as pricepaid;

pricepaid
-----------
162
(1 row)
```

Dalam contoh ini, nilai dalam kolom stempel waktu ditampilkan sebagai tanggal, yang menghasilkan penghapusan waktu dari setiap hasil:

```
select cast(saletime as date), salesid
from sales order by salesid limit 10;

 saletime  | salesid
-----------+---------
2008-02-18 |       1
2008-06-06 |       2
2008-06-06 |       3
2008-06-09 |       4
2008-08-31 |       5
2008-07-16 |       6
2008-06-26 |       7
2008-07-10 |       8
2008-07-22 |       9
2008-08-06 |      10
(10 rows)
```

Jika Anda tidak menggunakan CAST seperti yang diilustrasikan dalam sampel sebelumnya, hasilnya akan mencakup waktu: 2008-02-18 02:36:48.

Kueri berikut mengirimkan data karakter variabel ke tanggal. Itu tidak memerlukan data sampel untuk dijalankan.

```
select cast('2008-02-18 02:36:48' as date) as mysaletime;

mysaletime
--------------------
2008-02-18
(1 row)
```

Dalam contoh ini, nilai-nilai dalam kolom tanggal diberikan sebagai stempel waktu:

```
select cast(caldate as timestamp), dateid
from date order by dateid limit 10;

      caldate       | dateid
--------------------+--------
2008-01-01 00:00:00 |   1827
2008-01-02 00:00:00 |   1828
2008-01-03 00:00:00 |   1829
2008-01-04 00:00:00 |   1830
2008-01-05 00:00:00 |   1831
2008-01-06 00:00:00 |   1832
2008-01-07 00:00:00 |   1833
2008-01-08 00:00:00 |   1834
2008-01-09 00:00:00 |   1835
2008-01-10 00:00:00 |   1836
(10 rows)
```

Dalam kasus seperti sampel sebelumnya, Anda dapat memperoleh kontrol tambahan atas pemformatan output dengan menggunakan TO_ CHAR.

Dalam contoh ini, bilangan bulat digunakan sebagai string karakter:

```
select cast(2008 as char(4));

bpchar
--------
2008
```

Dalam contoh ini, nilai DECIMAL (6,3) diberikan sebagai nilai DECIMAL (4,1):

```
select cast(109.652 as decimal(4,1));

numeric
---------
109.7
```

Contoh ini menunjukkan ekspresi yang lebih kompleks. Kolom PRICEPAID (kolom DECIMAL (8,2)) dalam tabel PENJUALAN dikonversi ke kolom DECIMAL (38,2) dan nilainya dikalikan dengan 100000000000000000000:

```
select salesid, pricepaid::decimal(38,2)*100000000000000000000
as value from sales where salesid<10 order by salesid;


 salesid |           value
---------+----------------------------
       1 | 72800000000000000000000.00
       2 |  76000000000000000000000.00
       3 | 35000000000000000000000.00
       4 | 17500000000000000000000.00
       5 | 15400000000000000000000.00
       6 | 39400000000000000000000.00
       7 | 78800000000000000000000.00
       8 | 19700000000000000000000.00
       9 | 59100000000000000000000.00
(9 rows)
```

> **Note**
>
> Anda tidak dapat melakukan operasi CAST atau CONVERT pada tipe `GEOMETRY` data untuk mengubahnya ke tipe data lain. Namun, Anda dapat memberikan representasi heksadesimal dari string literal dalam format biner terkenal diperluas (EWKB) sebagai input ke fungsi yang menerima `GEOMETRY` argumen. Misalnya, `ST_AsText` fungsi berikut mengharapkan tipe `GEOMETRY` data.
>
> ```
> SELECT ST_AsText('010100000000000000000001C400000000000002040');
> ```
>
> ```
> st_astext
> ------------
> ```

```
 POINT(7 8)
```

Anda juga dapat secara eksplisit menentukan tipe GEOMETRY data.

```
SELECT ST_AsText('0101000000000000000000144000000000000001840'::geometry);
```

```
st_astext
------------
 POINT(5 6)
```

Fungsi CONVERT

Seperti fungsi CAST, fungsi CONVERT mengubah satu tipe data ke tipe data lain yang kompatibel. Misalnya, Anda dapat mengonversi string ke tanggal, atau tipe numerik ke string. CONVERT melakukan konversi runtime, yang berarti konversi tidak mengubah tipe data nilai dalam tabel sumber. Itu hanya diubah dalam konteks kueri.

Tipe data tertentu memerlukan konversi eksplisit ke tipe data lain menggunakan fungsi CONVERT. Jenis data lain dapat dikonversi secara implisit, sebagai bagian dari perintah lain, tanpa menggunakan CAST atau CONVERT. Lihat Kompatibilitas dan konversi tipe.

Sintaksis

```
CONVERT ( type, expression )
```

Argumen

jenis

Salah satu yang didukungJenis Data.

ekspresi

Ekspresi yang mengevaluasi satu atau lebih nilai, seperti nama kolom atau literal. Mengkonversi nilai null mengembalikan nol. Ekspresi tidak dapat berisi string kosong atau kosong.

Jenis pengembalian

CONVERT mengembalikan tipe data yang ditentukan oleh argum en tipe.

> **Note**
>
> Amazon Redshift mengembalikan kesalahan jika Anda mencoba melakukan konversi yang bermasalah, seperti konversi DECIMAL yang kehilangan presisi, seperti berikut ini:
>
> ```
> SELECT CONVERT(decimal(2,1), 123.456);
> ```
>
> atau konversi INTEGER yang menyebabkan luapan:
>
> ```
> SELECT CONVERT(smallint, 12345678);
> ```

Contoh

Beberapa contoh menggunakan contoh database https://docs.aws.amazon.com/redshift/latest/dg/c_sampledb.html TICKIT. Untuk informasi selengkapnya tentang menyiapkan data sampel, lihat Mem uat data.

Kueri berikut menggunakan fungsi CONVERT untuk mengubah kolom desimal menjadi bilangan bulat

```
SELECT CONVERT(integer, pricepaid)
FROM sales WHERE salesid=100;
```

Contoh ini mengubah integer menjadi string karakter.

```
SELECT CONVERT(char(4), 2008);
```

Dalam contoh ini, tanggal dan waktu saat ini dikonversi ke tipe data karakter variabel:

```
SELECT CONVERT(VARCHAR(30), GETDATE());

getdate
---------
2023-02-02 04:31:16
```

Contoh ini mengubah kolom waktu penjualan menjadi hanya waktu, menghapus tanggal dari setiap baris.

```
SELECT CONVERT(time, saletime), salesid
```

```
FROM sales order by salesid limit 10;
```

Untuk informasi tentang mengubah stempel waktu dari satu zona waktu ke zona waktu lainnya, lihat. Fungsi CONVERT_TIMEZONE Untuk fungsi tanggal dan waktu tambahan, lihatFungsi tanggal dan waktu.

Contoh berikut mengubah data karakter variabel menjadi objek datetime.

```
SELECT CONVERT(datetime, '2008-02-18 02:36:48') as mysaletime;
```

> **Note**
>
> Anda tidak dapat melakukan operasi CAST atau CONVERT pada tipe GEOMETRY data untuk mengubahnya ke tipe data lain. Namun, Anda dapat memberikan representasi heksadesimal dari string literal dalam format biner terkenal diperluas (EWKB) sebagai input ke fungsi yang menerima GEOMETRY argumen. Misalnya, ST_AsText fungsi berikut mengharapkan tipe GEOMETRY data.
>
> ```
> SELECT ST_AsText('010100000000000000000001C40000000000002040');
> ```
>
> ```
> st_astext
> ------------
>  POINT(7 8)
> ```
>
> Anda juga dapat secara eksplisit menentukan tipe GEOMETRY data.
>
> ```
> SELECT ST_AsText('010100000000000000000014400000000000001840'::geometry);
> ```
>
> ```
> st_astext
> ------------
>  POINT(5 6)
> ```

TEKS_KE_INT_ALT

TEXT_TO_INT_ALT mengonversi string karakter ke integer menggunakan pemformatan. Teradata-style Digit pecahan dalam hasil dipotong.

Sintaksis

```
TEXT_TO_INT_ALT (expression [ , 'format'])
```

Argumen

ekspresi

Ekspresi yang menghasilkan satu atau lebih nilai CHAR atau VARCHAR, seperti nama kolom atau string literal. Mengkonversi nilai null mengembalikan nol. Fungsi mengubah string kosong atau kosong menjadi 0.

format

Sebuah string literal yang mendefinisikan format ekspresi input. Untuk informasi selengkapnya tentang karakter pemformatan yang dapat Anda tentukan, lihatTeradata-style memformat karakter untuk data numerik.

Jenis pengembalian

TEXT_TO_INT_ALT mengembalikan nilai INTEGER.

Bagian fraksional dari hasil cast terpotong.

Amazon Redshift mengembalikan kesalahan jika konversi ke frasa format yang Anda tentukan tidak berhasil.

Contoh

Contoh berikut mengubah string ekspresi input '123-' ke integer -123.

```
select text_to_int_alt('123-');
```

```
text_to_int_alt
----------
      -123
```

Contoh berikut mengubah string ekspresi input '2147483647+' ke integer 2147483647.

```
select text_to_int_alt('2147483647+');
```

```
text_to_int_alt
----------
2147483647
```

Contoh berikut mengubah string ekspresi input eksponensial '-123 E-2 'menjadi integer -1.

```
select text_to_int_alt('-123E-2');
```

```
text_to_int_alt
----------
        -1
```

Contoh berikut mengubah string ekspresi input '2147483647+' ke integer 2147483647.

```
select text_to_int_alt('2147483647+');
```

```
text_to_int_alt
----------
2147483647
```

Contoh berikut mengubah string ekspresi input '123 {'dengan frasa format' 999S 'ke integer 1230. Karakter S menunjukkan Desimal Zoning yang Ditandatangani. Untuk informasi selengkapnya, lihat Teradata-style memformat karakter untuk data numerik.

```
select text_to_int_alt('123{', '999S');
```

```
text_to_int_alt
----------
      1230
```

Contoh berikut mengubah string ekspresi input 'USD123' dengan frasa format 'C9 (I) 'ke integer 123. Lihat Teradata-style memformat karakter untuk data numerik.

```
select text_to_int_alt('USD123', 'C9(I)');
```

```
text_to_int_alt
----------
```

```
       123
```

Contoh berikut menentukan kolom tabel sebagai ekspresi masukan.

```
select text_to_int_alt(a), text_to_int_alt(b) from t_text2int order by 1;
```

```
 text_to_int_alt | text_to_int_alt
-----------------+-----------------
            -123 |            -123
            -123 |            -123
             123 |             123
             123 |             123
```

Berikut ini adalah definisi tabel dan pernyataan insert untuk contoh ini.

```
create table t_text2int (a varchar(200), b char(200));
```

```
insert into t_text2int VALUES('123', '123'),('123.123', '123.123'), ('-123', '-123'),
 ('123-', '123-');
```

TEKS_KE_NUMERIK_ALT

TEXT_TO_NUMERIC_ALT melakukan operasi Teradata-style cast untuk mengonversi string karakter ke format data numerik.

Sintaksis

```
TEXT_TO_NUMERIC_ALT (expression [, 'format'] [, precision, scale])
```

Argumen

ekspresi

Ekspresi yang mengevaluasi satu atau lebih nilai CHAR atau VARCHAR, seperti nama kolom atau literal. Mengkonversi nilai null mengembalikan nol. String kosong atau kosong diubah menjadi 0.

format

Sebuah string literal yang mendefinisikan format ekspresi input. Untuk informasi selengkapnya, lihat Teradata-style memformat karakter untuk data numerik.

presisi

Jumlah digit dalam hasil numerik. Defaultnya adalah 38.

skala

Jumlah digit di sebelah kanan titik desimal dalam hasil numerik. Default-nya adalah 0.

Jenis pengembalian

TEXT_TO_NUMERIC_ALT mengembalikan angka DECIMAL.

Amazon Redshift mengembalikan kesalahan jika konversi ke frasa format yang Anda tentukan tidak berhasil.

Amazon Redshift mentransmisikan string ekspresi masukan ke tipe numerik dengan presisi tertinggi yang Anda tentukan untuk jenis tersebut dalam opsi presisi. Jika panjang nilai numerik melebihi nilai yang Anda tentukan untuk presisi, Amazon Redshift membulatkan nilai numerik sesuai dengan aturan berikut:

- Jika panjang hasil cast melebihi panjang yang Anda tentukan dalam frase format, Amazon Redshift mengembalikan kesalahan.
- Jika hasilnya dialihkan ke nilai numerik, hasilnya dibulatkan ke nilai terdekat. Jika bagian pecahan tepat di tengah antara hasil cetak atas dan bawah, hasilnya dibulatkan ke nilai genap terdekat.

Contoh

Contoh berikut mengubah string ekspresi input '1.5' ke nilai numerik '2'. Karena pernyataan tidak menentukan skala, skala default ke 0 dan hasil cast tidak menyertakan hasil pecahan. Karena 0,5 berada di tengah antara 1 dan 2, hasil cast dibulatkan ke nilai genap 2.

```
select text_to_numeric_alt('1.5');
```

```
 text_to_numeric_alt
---------------------
                   2
```

Contoh berikut mengubah string ekspresi input '2.51' ke nilai numerik 3. Karena pernyataan tidak menentukan nilai skala, skala default ke 0 dan hasil cast tidak menyertakan hasil pecahan. Karena 0,51 lebih dekat ke 3 dari 2, hasil cast dibulatkan menjadi nilai 3.

```
select text_to_numeric_alt('2.51');
```

```
 text_to_numeric_alt
---------------------
                   3
```

Contoh berikut mengubah string ekspresi input 123.52501 dengan presisi 10 dan skala 2 ke nilai numerik 123.53.

```
select text_to_numeric_alt('123.52501', 10, 2);
```

```
 text_to_numeric_alt
---------------------
              123.53
```

Contoh berikut mengubah string ekspresi input '123 {'dengan frasa format' 999S 'ke angka 1230. Karakter S menunjukkan Desimal Zoning yang Ditandatangani. Untuk informasi selengkapnya, lihat Teradata-style memformat karakter untuk data numerik.

```
select text_to_numeric_alt('123{', '999S');
```

```
text_to_int_alt
----------
      1230
```

Contoh berikut mengubah string ekspresi input 'USD123' dengan frasa format 'C9 (I) 'menjadi angka 124. Lihat Teradata-style memformat karakter untuk data numerik.

```
select text_to_numeric_alt('USD123.9', 'C9(I)');
```

```
text_to_numeric_alt
----------
       124
```

Contoh berikut menentukan kolom tabel sebagai ekspresi masukan.

```
select text_to_numeric_alt(a), text_to_numeric_alt(b) from t_text2numeric order by 1;
```

```
          text_to_numeric_alt           |          text_to_numeric_alt
----------------------------------------+----------------------------------------
 -99999999999999999999999999999999999999 | -99999999999999999999999999999999999999
                                -12300 |                                 -12300
                                   123 |                                    123
                                   123 |                                    123
  99999999999999999999999999999999999999 |  99999999999999999999999999999999999999
```

Berikut ini adalah definisi tabel dan pernyataan insert untuk contoh ini.

```
create table  t_text2numeric (a varchar(200), b char(200));
```

```
insert into  t_text2numeric values
('123', '123'),
('+123.456', '+123.456'),
('-' || repeat('9', 38), '-' || repeat('9', 38)),
(repeat('9', 38) || '+', repeat('9', 38) || '+'),
('-123E2', '-123E2');
```

TO_CHAR

TO_CHAR mengonversi stempel waktu atau ekspresi numerik ke format data string karakter.

Sintaksis

```
TO_CHAR (timestamp_expression | numeric_expression , 'format')
```

Argumen

ekspresi stempel waktu

Ekspresi yang menghasilkan nilai tipe TIMESTAMP atau TIMESTAMPTZ atau nilai yang secara implisit dapat dipaksa ke stempel waktu.

Ekspresi numerik

Ekspresi yang menghasilkan nilai tipe data numerik atau nilai yang secara implisit dapat dipaksa ke tipe numerik. Untuk informasi selengkapnya, lihat Jenis numerik. TO_CHAR menyisipkan spasi di sebelah kiri string angka.

> **Note**
>
> TO_CHAR tidak mendukung nilai DECIMAL 128-bit.

format

Format untuk nilai baru. Untuk format yang valid, lihat String format tanggal-waktu danString format numerik.

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut mengubah stempel waktu menjadi nilai dengan tanggal dan waktu dalam format dengan nama bulan dilapisi menjadi sembilan karakter, nama hari dalam seminggu, dan nomor hari bulan.

```
select to_char(timestamp '2009-12-31 23:15:59', 'MONTH-DY-DD-YYYY HH12:MIPM');

to_char
-------------------------
DECEMBER -THU-31-2009 11:15PM
```

Contoh berikut mengubah stempel waktu menjadi nilai dengan nomor hari tahun.

```
select to_char(timestamp '2009-12-31 23:15:59', 'DDD');

to_char
-------------------------
365
```

Contoh berikut mengubah stempel waktu ke nomor hari ISO dalam seminggu.

```
select to_char(timestamp '2022-05-16 23:15:59', 'ID');

to_char
```

```
-------------------------
1
```

Contoh berikut mengekstrak nama bulan dari tanggal.

```
select to_char(date '2009-12-31', 'MONTH');

to_char
-------------------------
DECEMBER
```

Contoh berikut mengubah setiap nilai STARTTIME dalam tabel EVENT ke string yang terdiri dari jam, menit, dan detik.

```
select to_char(starttime, 'HH12:MI:SS')
from event where eventid between 1 and 5
order by eventid;

to_char
----------
02:30:00
08:00:00
02:30:00
02:30:00
07:00:00
```

Contoh berikut mengubah seluruh nilai stempel waktu ke dalam format yang berbeda.

```
select starttime, to_char(starttime, 'MON-DD-YYYY HH12:MIPM')
from event where eventid=1;

      starttime      |       to_char
---------------------+---------------------
 2008-01-25 14:30:00 | JAN-25-2008 02:30PM
```

Contoh berikut mengubah timestamp literal menjadi string karakter.

```
select to_char(timestamp '2009-12-31 23:15:59','HH24:MI:SS');

to_char
----------
```

```
23:15:59
```

Contoh berikut mengubah angka desimal menjadi string karakter.

```
select to_char(125.8, '999.99');

to_char
---------
125.80
```

Contoh berikut mengubah angka desimal menjadi string karakter.

```
select to_char(125.8, '999D99');

to_char
---------
125.80
```

Contoh berikut mengubah angka menjadi string karakter dengan nol di depan.

```
select to_char(125.8, '0999D99');

to_char
---------
0125.80
```

Contoh berikut mengubah angka menjadi string karakter dengan tanda negatif di akhir.

```
select to_char(-125.8, '999D99S');

to_char
---------
125.80-
```

Contoh berikut mengubah angka menjadi string karakter dengan tanda positif atau negatif pada posisi yang ditentukan.

```
select to_char(125.8, '999D99SG');

to_char
---------
```

```
125.80+
```

Contoh berikut mengubah angka menjadi string karakter dengan tanda positif pada posisi yang ditentukan.

```
select to_char(125.8, 'PL999D99');

to_char
---------
+ 125.80
```

Contoh berikut mengubah angka menjadi string karakter dengan simbol mata uang.

```
select to_char(-125.88, '$S999D99');

to_char
---------
$-125.88
```

Contoh berikut mengubah angka menjadi string karakter dengan simbol mata uang di posisi yang ditentukan.

```
select to_char(-125.88, 'S999D99L');

to_char
---------
-125.88$
```

Contoh berikut mengubah angka menjadi string karakter menggunakan pemisah ribuan (koma).

```
select to_char(1125.8, '9,999.99');

to_char
---------
1,125.80
```

Contoh berikut mengubah angka menjadi string karakter menggunakan tanda kurung sudut untuk angka negatif.

```
select to_char(-125.88, '$999D99PR');
```

```
to_char
---------
$<125.88>
```

Contoh berikut mengubah angka menjadi string angka Romawi.

```
select to_char(125, 'RN');

to_char
---------
   CXXV
```

Contoh berikut mengubah tanggal menjadi kode abad.

```
select to_char(date '2020-12-31', 'CC');

to_char
---------
21
```

Contoh berikut menampilkan hari dalam seminggu.

```
SELECT to_char(current_timestamp, 'FMDay, FMDD HH12:MI:SS');

to_char
-----------------------
Wednesday, 31 09:34:26
```

Contoh berikut menampilkan akhiran nomor ordinal untuk angka.

```
SELECT to_char(482, '999th');

to_char
-----------------------
 482nd
```

Contoh berikut mengurangi komisi dari harga yang dibayarkan dalam tabel penjualan. Perbedaannya kemudian dibulatkan dan diubah menjadi angka romawi, ditunjukkan pada kolom to_char:

```
select salesid, pricepaid, commission, (pricepaid - commission)
```

```
as difference, to_char(pricepaid - commission, 'rn') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;

 salesid | pricepaid | commission | difference |     to_char
---------+-----------+------------+------------+-----------------
       1 |    728.00 |     109.20 |     618.80 |           dcxix
       2 |     76.00 |      11.40 |      64.60 |             lxv
       3 |    350.00 |      52.50 |     297.50 |        ccxcviii
       4 |    175.00 |      26.25 |     148.75 |           cxlix
       5 |    154.00 |      23.10 |     130.90 |           cxxxi
       6 |    394.00 |      59.10 |     334.90 |         cccxxxv
       7 |    788.00 |     118.20 |     669.80 |           dclxx
       8 |    197.00 |      29.55 |     167.45 |          clxvii
       9 |    591.00 |      88.65 |     502.35 |             dii
      10 |     65.00 |       9.75 |      55.25 |              lv
```

Contoh berikut menambahkan simbol mata uang ke nilai perbedaan yang ditunjukkan pada kolom to_char:

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'l99999D99') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;

salesid | pricepaid | commission | difference |  to_char
--------+-----------+------------+------------+------------
      1 |    728.00 |     109.20 |     618.80 | $   618.80
      2 |     76.00 |      11.40 |      64.60 | $    64.60
      3 |    350.00 |      52.50 |     297.50 | $   297.50
      4 |    175.00 |      26.25 |     148.75 | $   148.75
      5 |    154.00 |      23.10 |     130.90 | $   130.90
      6 |    394.00 |      59.10 |     334.90 | $   334.90
      7 |    788.00 |     118.20 |     669.80 | $   669.80
      8 |    197.00 |      29.55 |     167.45 | $   167.45
      9 |    591.00 |      88.65 |     502.35 | $   502.35
     10 |     65.00 |       9.75 |      55.25 | $    55.25
```

Contoh berikut mencantumkan abad di mana setiap penjualan dilakukan.

```
select salesid, saletime, to_char(saletime, 'cc') from sales
order by salesid limit 10;
```

```
 salesid |      saletime       | to_char
---------+---------------------+---------
       1 | 2008-02-18 02:36:48 | 21
       2 | 2008-06-06 05:00:16 | 21
       3 | 2008-06-06 08:26:17 | 21
       4 | 2008-06-09 08:38:52 | 21
       5 | 2008-08-31 09:17:02 | 21
       6 | 2008-07-16 11:59:24 | 21
       7 | 2008-06-26 12:56:06 | 21
       8 | 2008-07-10 02:12:36 | 21
       9 | 2008-07-22 02:23:17 | 21
      10 | 2008-08-06 02:51:55 | 21
```

Contoh berikut mengubah setiap nilai STARTTIME dalam tabel EVENT ke string yang terdiri dari jam, menit, detik, dan zona waktu.

```
select to_char(starttime, 'HH12:MI:SS TZ')
from event where eventid between 1 and 5
order by eventid;

to_char
----------
02:30:00 UTC
08:00:00 UTC
02:30:00 UTC
02:30:00 UTC
07:00:00 UTC
```

Contoh berikut menunjukkan pemformatan untuk detik, milidetik, dan mikrodetik.

```
select sysdate,
to_char(sysdate, 'HH24:MI:SS') as seconds,
to_char(sysdate, 'HH24:MI:SS.MS') as milliseconds,
to_char(sysdate, 'HH24:MI:SS:US') as microseconds;

timestamp           | seconds  | milliseconds | microseconds
--------------------+----------+--------------+----------------
2015-04-10 18:45:09 | 18:45:09 | 18:45:09.325 | 18:45:09:325143
```

Fungsi TO_DATE

TO_DATE mengonversi tanggal yang diwakili oleh string karakter ke tipe data DATE.

> **Note**
>
> TO_DATE tidak mendukung string format dengan Q (Quarter number).

Sintaksis

```
TO_DATE(string, format)
```

```
TO_DATE(string, format, is_strict)
```

Argumen

tali

Sebuah string yang akan dikonversi.

format

Sebuah string literal yang mendefinisikan format string input, dalam hal bagian tanggalnya. Untuk daftar format hari, bulan, dan tahun yang valid, lihatString format tanggal-waktu.

is_ketat

Nilai Boolean opsional yang menentukan apakah kesalahan dikembalikan jika nilai tanggal input berada di luar rentang. Ketika is_strict disetel keTRUE, kesalahan dikembalikan jika ada nilai di luar jangkauan. Ketika is_strict disetel keFALSE, yang merupakan default, maka nilai overflow diterima.

Jenis pengembalian

TO_DATE mengembalikan DATE, tergantung pada nilai format.

Jika konversi ke format gagal, maka kesalahan dikembalikan.

Contoh

Pernyataan SQL berikut mengubah tanggal `02 Oct 2001` menjadi tipe data tanggal.

```
select to_date('02 Oct 2001', 'DD Mon YYYY');
```

```
to_date
------------
2001-10-02
(1 row)
```

Pernyataan SQL berikut mengubah string 20010631 ke tanggal.

```
select to_date('20010631', 'YYYYMMDD', FALSE);
```

Hasilnya adalah 1 Juli 2001, karena hanya ada 30 hari di bulan Juni.

```
to_date
------------
2001-07-01
```

Pernyataan SQL berikut mengubah string 20010631 ke tanggal:

```
to_date('20010631', 'YYYYMMDD', TRUE);
```

Hasilnya adalah kesalahan karena hanya ada 30 hari di bulan Juni.

```
ERROR:  date/time field date value out of range: 2001-6-31
```

TO_NOMOR

TO_NUMBER mengonversi string ke nilai numerik (desimal).

> **Note**
>
> Sebaiknya gunakan FM dalam format string untuk menekan padding blanko dan nol. Untuk daftar format yang valid, lihatString format numerik.

Sintaksis

```
to_number(string, format)
```

Argumen

tali

String yang akan dikonversi. Format harus berupa nilai literal.

format

Argumen kedua adalah string format yang menunjukkan bagaimana string karakter harus diurai untuk membuat nilai numerik. Misalnya, format `'FM99D999'` menentukan bahwa string yang akan dikonversi terdiri dari lima digit dengan titik desimal di posisi ketiga. Misalnya, `to_number('12.345','FM99D999')` mengembalikan `12.345` sebagai nilai numerik. Untuk daftar format yang valid, lihatString format numerik.

Jenis pengembalian

TO_NUMBER mengembalikan angka DECIMAL.

Jika konversi ke format gagal, maka kesalahan dikembalikan.

Contoh

Contoh berikut mengubah string `12,454.8-` menjadi angka:

```
select to_number('12,454.8-', 'FM99G999D9S');

to_number
-----------
-12454.8
```

Contoh berikut mengubah string `$ 12,454.88` menjadi angka:

```
select to_number('$ 12,454.88', 'FML99G999D99');

to_number
-----------
12454.88
```

Contoh berikut mengubah string `$ 2,012,454.88` menjadi angka:

```
select to_number('$ 2,012,454.88', 'FML9,999,999.99');
```

```
to_number
-----------
2012454.88
```

Fungsi TRY_CAST

Dibandingkan dengan fungsi CAST, TRY_CAST pertama-tama mencoba untuk mentransmisikan ekspresi ke tipe yang ditentukan. Jika casting gagal karena kesalahan konversi, operasi mengembalikan null. Jika konversi tidak diizinkan secara eksplisit, operasi mengembalikan kesalahan. Anda dapat menemukan daftar kemungkinan konversi di catatan penggunaan di bawah ini. Misalnya, mencoba mengonversi boolean ke stempel waktu tidak diizinkan.

Sintaksis

```
TRY_CAST(expression AS type)
```

Argumen

ekspresi

Ekspresi yang mengevaluasi satu atau lebih nilai, seperti nama kolom atau literal. Mengkonversi nilai null mengembalikan nol. Ekspresi tidak dapat berisi string kosong atau kosong.

jenis

Salah satu tipe data yang didukung. Untuk daftar lengkap tipe data, lihatJenis Data. Untuk daftar jenis data sumber yang didukung dan pasangan tipe data target, lihat catatan penggunaan di bawah ini.

Jenis pengembalian

TRY_CAST mengembalikan nilai tipe data yang ditentukan oleh argumen tipe. Jika konversi gagal, operasi mengembalikan null.

Catatan penggunaan

Berikut ini adalah daftar tipe data sumber dan pasangan tipe data target yang didukung Amazon Redshift untuk TRY_CAST.

BOOL

SMALLINT, INT, BIGINT, SUPER

SMALLINT

BOOL, INT, BIGINT, DESIMAL, REAL, FLOAT, BPCHAR, TEKS, VARCHAR, SUPER

INT

BOOL, SMALLINT, BIGINT, DESIMAL, REAL, FLOAT, BPCHAR, TEKS, VARCHAR, SUPER

BIGINT

BOOL, SMALLINT, INT, DESIMAL, REAL, FLOAT, BPCHAR, TEKS, VARCHAR, SUPER

DESIMAL

SMALLINT, INT, BIGINT, REAL, FLOAT, BPCHAR, TEKS, VARCHAR, SUPER

NYATA

SMALLINT, INT, BIGINT, DESIMAL, FLOAT, BPCHAR, TEKS, VARCHAR, SUPER

MENGAPUNG

SMALLINT, INT, BIGINT, DESIMAL, REAL, BPCHAR, TEKS, VARCHAR, SUPER

BPCHAR

SMALLINT, INT, BIGINT, DESIMAL, REAL, FLOAT, TEKS, VARCHAR, STEMPEL WAKTU, TIMESTAMPTZ, TANGGAL, WAKTU, TIMEZ, SUPER

TEKS

SMALLINT, INT, BIGINT, DESIMAL, REAL, FLOAT, BPCHAR, VARCHAR, STEMPEL WAKTU, TIMESTAMPTZ, TANGGAL, WAKTU, TIMEZ, SUPER

VARCHAR

SMALLINT, INT, BIGINT, DESIMAL, REAL, FLOAT, BPCHAR, TEKS, STEMPEL WAKTU, TIMESTAMPTZ, TANGGAL, WAKTU, TIMEZ, SUPER

STEMPEL WAKTU

BPCHAR, TEKS, VARCHAR, TIMESTAMPTZ, TANGGAL, WAKTU

STEMPEL WAKTU

BPCHAR, TEKS, VARCHAR, STEMPEL WAKTU, TANGGAL, WAKTU, TIMEZ

TANGGAL

BPCHAR, TEKS, VARCHAR, STEMPEL WAKTU, STEMPEL WAKTU

WAKTU

BPCHAR, TEKS, VARCHAR

TIMETZ

BPCHAR, TEKS, VARCHAR

SUPER

SUPER dapat dikonversi ke jenis data lainnya, dengan pengecualian DATE, TIME, TIMETZ, TIMESTAMP, dan TIMESTAMPTZ.

Contoh

Contoh berikut melemparkan STRING menjadi INTEGER.

```
SELECT TRY_CAST('123' AS INT);

int
----
123
```

Contoh berikut mengembalikan null. Mengkonversi STRING ke INTEGER diizinkan sehingga TRY_CAST tidak mengembalikan kesalahan, tetapi 'foo' bukan bilangan bulat sehingga fungsinya mengembalikan null.

```
SELECT TRY_CAST('foo' AS INT)
```

Contoh berikut mengembalikan kesalahan, karena mengonversi BOOLEAN ke TIMESTAMP tidak diizinkan.

```
SELECT TRY_CAST(true as timestamp);
```

Karena TRY_CAST mengembalikan null alih-alih segera mengembalikan kesalahan jika konversi gagal, Anda dapat menggunakan TRY_CAST untuk menyaring data yang tidak valid. Pertimbangkan contoh berikut, di mana baris yang tidak valid disaring karena kegagalan konversi di kolom usia untuk Akua Mansa.

```
CREATE TABLE IF NOT EXISTS student_data(
name VARCHAR(100) NOT NULL,
```

```
age VARCHAR(3) NOT NULL,
enrollment_date DATE NOT NULL);

INSERT INTO student_data (name, age, enrollment_date)
VALUES
('Alejandro Rosalez', '10', '01/01/2000'),
('Akua Mansa', 'Ten', '01/01/2000');

SELECT * FROM student_data WHERE TRY_CAST(age AS INT) IS NOT NULL;

--Akua is not returned.
 name              | age | enrollment_date
-------------------+-----+-----------------
 Alejandro Rosalez | 10  | 01/01/2000
```

String format tanggal-waktu

Anda dapat menemukan referensi untuk string format datetime berikut.

String format berikut berlaku untuk fungsi seperti TO_CHAR. String ini dapat berisi pemisah datetime (seperti '-', '/', atau ':') dan “dateparts” dan “timeparts” berikut.

Part tanggal atau bagian waktu	Arti
BC atau B.C., AD atau A.D., b.c. atau bc, ad atau a.d.	Indikator era huruf besar dan kecil
CC	Two-digit nomor abad
YYYY, YYY, YY, Y	Nomor tahun 4 digit, 3 digit, 2 digit, 1 digit
Y, YYY	Nomor tahun 4 digit dengan koma
IYYY, IYY, IY, I	4 digit, 3 digit, 2 digit, 1 digit nomor tahun Organisasi Internasional untuk Standardisasi (ISO)
Q	Nomor kuartal (1 hingga 4)
BULAN, Bulan, bulan	Nama bulan (huruf besar, huruf campuran, huruf kecil, kosong hingga 9 karakter)

Part tanggal atau bagian waktu	Arti
Senin, Senin, Senin	Nama bulan yang disingkat (huruf besar, huruf campuran, huruf kecil, diisi kosong hingga 3 karakter)
MM	Nomor bulan (01-12)
Rm, RM	Nomor bulan dalam angka Romawi (I—XII, dengan I adalah Januari, huruf besar atau kecil)
W	Minggu dalam sebulan (1—5; minggu pertama dimulai pada hari pertama bulan itu.)
WW	Nomor minggu tahun (1—53; minggu pertama dimulai pada hari pertama tahun itu.)
IW	Nomor minggu ISO tahun (Kamis pertama tahun baru adalah di minggu 1.)
HARI, Hari, hari	Nama hari (huruf besar, huruf campuran, huruf kecil, kosong hingga 9 karakter)
NY, Dy, Dy	Nama hari yang disingkat (huruf besar, huruf campuran, huruf kecil, diisi kosong hingga 3 karakter)
DDD	Hari dalam setahun (001—366)
IDD	Hari tahun penomoran minggu ISO 8601 (001-371; hari 1 tahun adalah Senin minggu ISO pertama)
DD	Hari dalam sebulan sebagai angka (01-31)

Part tanggal atau bagian waktu	Arti
D	Hari dalam seminggu (1—7; Minggu adalah 1) **Note** Bagian tanggal D berperilaku berbeda dari datepart day of week (DOW) yang digunakan untuk fungsi datetime DATE_PART dan EXTRACT. DOW didasarkan pada bilangan bulat 0-6, di mana hari Minggu adalah 0. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.
ID	ISO 8601 hari dalam seminggu, Senin (1) hingga Minggu (7)
J	Hari Julian (hari-hari sejak 1 Januari 4712 SM)
HH24	Jam (jam 24 jam, 00—23)
HH atau HH12	Jam (jam 12 jam, 01—12)
MI	Menit (00—59)
SS	Detik (00—59)
MS	Milidetik (.000)
US	Mikrodetik (.000000)
AM atau PM, A.M. atau P.M., pagi atau sore, pagi atau sore	Indikator meridian huruf besar dan kecil (untuk jam 12 jam)
TZ, tz	Singkatan zona waktu huruf besar dan kecil; hanya berlaku untuk TIMESTAMPTZ

Part tanggal atau bagian waktu	Arti
DARI	Offset dari UTC; hanya berlaku untuk TIMESTAMPTZ

> **Note**
>
> Anda harus mengelilingi pemisah datetime (seperti '-', '/' atau ':') dengan tanda kutip tunggal, tetapi Anda harus mengelilingi “dateparts” dan “timeparts” yang tercantum dalam tabel sebelumnya dengan tanda kutip ganda.

Contoh

Untuk contoh memformat tanggal sebagai string, lihatTO_CHAR.

String format numerik

Berikut ini, Anda dapat menemukan referensi untuk string format numerik.

String format berikut berlaku untuk fungsi seperti TO_NUMBER dan TO_CHAR.

- Untuk contoh memformat string sebagai angka, lihatTO_NOMOR.
- Untuk contoh memformat angka sebagai string, lihatTO_CHAR.

Format	Deskripsi
9	Nilai numerik dengan jumlah digit yang ditentukan.
0	Nilai numerik dengan nol terdepan.
. (periode), D	Titik desimal.
, (koma)	Ribuan pemisah.
CC	Kode abad. Misalnya, abad ke-21 dimulai pada 2001-01-01 (didukung hanya untuk TO_CHAR).

Format	Deskripsi
FM	Mode isi. Menekan padding kosong dan nol.
PR	Nilai negatif dalam tanda kurung sudut.
D	Tanda ditambatkan ke nomor.
L	Simbol mata uang di posisi yang ditentukan.
G	Pemisah grup.
MI	Tanda minus pada posisi yang ditentukan untuk angka yang kurang dari 0.
PL	Tanda plus di posisi yang ditentukan untuk angka yang lebih besar dari 0.
SG	Tanda plus atau minus di posisi yang ditentuka n.
RN	Angka Romawi antara 1 dan 3999 (hanya didukung untuk TO_CHAR).
TH atau th	Sufiks nomor ordinal. Tidak mengubah bilangan pecahan atau nilai yang kurang dari nol.

Teradata-style memformat karakter untuk data numerik

Berikut ini, Anda dapat menemukan bagaimana fungsi TEXT_TO_INT_ALT dan TEXT_TO_NUMERIC_ALT menafsirkan karakter dalam string ekspresi input. Anda juga dapat menemukan daftar karakter yang dapat Anda tentukan dalam frasa format. Selain itu, Anda dapat menemukan deskripsi perbedaan antara Teradata-style pemformatan dan Amazon Redshift untuk opsi format.

Format	Deskripsi
G	Tidak didukung sebagai pemisah grup dalam string ekspresi input. Anda tidak dapat menentukan karakter ini dalam frasa format.
D	Simbol Radix. Anda dapat menentukan karakter ini dalam frasa format. Karakter ini setara dengan. (periode). Simbol Radix tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • . (periode) • S (huruf besar 's') • V (huruf besar 'v')
/ , : %	Karakter penyisipan/(garis miring maju), koma (,),: (titik dua), dan% (tanda persen). Anda tidak dapat memasukkan karakter ini dalam frasa format. Amazon Redshift mengabaikan karakter ini dalam string ekspresi masukan.
.	Periode sebagai karakter radix, yaitu titik desimal. Karakter ini tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • D (huruf besar 'd') • S (huruf besar 's') • V (huruf besar 'v')
B	Anda tidak dapat menyertakan karakter spasi kosong (B) dalam frasa format. Dalam string ekspresi input, spasi depan dan belakang

Format	Deskripsi
	diabaikan dan spasi di antara digit tidak diperbolehkan.
+ -	Anda tidak dapat menyertakan tanda plus (+) atau tanda minus (-) dalam frasa format. Namun, tanda plus (+) dan tanda minus (-) diurai secara implisit sebagai bagian dari nilai numerik jika muncul dalam string ekspresi input.
V	Indikator posisi titik desimal. Karakter ini tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • D (huruf besar 'd') • . (periode)
Z	Zero-suppressed digit desimal. Amazon Redshift memangkas nol di depan. Karakter Z tidak bisa mengikuti karakter 9. Karakter Z harus berada di sebelah kiri karakter radix jika bagian pecahan berisi karakter 9.
9	Digit desimal.

Format	Deskripsi
CHAR (n)	Untuk format ini, Anda dapat menentukan yang berikut: • CHAR terdiri dari Z atau 9 karakter. Amazon Redshift tidak mendukung + (plus) atau - (minus) dalam nilai CHAR. • n adalah konstanta integer, I, atau F. Untuk I, ini adalah jumlah karakter yang diperluka n untuk menampilkan bagian integer dari data numerik atau integer. Untuk F, ini adalah jumlah karakter yang diperlukan untuk menampilkan bagian pecahan data numerik.
-	Karakter tanda hubung (-). Anda tidak dapat memasukkan karakter ini dalam frasa format. Amazon Redshift mengabaikan karakter ini dalam string ekspresi masukan.

Format	Deskripsi
D	Desimal yang Ditandatangani Zonasi. Karakter S harus mengikuti digit desimal terakhir dalam frasa format. Karakter terakhir dari string ekspresi input dan konversi numerik yang sesuai tercantum di Karakter pemformatan data untuk Signed Zone Decimal, pemformatan data numerik bergaya Teradata. Karakter S tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • + (tanda tambah) • . (periode) • D (huruf besar 'd') • Z (huruf besar 'z') • F (huruf besar 'f') • E (huruf besar 'e')
E	Notasi eksponensial. String ekspresi input dapat menyertakan karakter eksponen. Anda tidak dapat menentukan E sebagai karakter eksponen dalam frasa format.
FN9	Tidak didukung di Amazon Redshift.
FNE	Tidak didukung di Amazon Redshift.

Format	Deskripsi
$, USD, Dolar Amerika Serikat	Tanda Dolar ($), simbol mata uang ISO (USD), dan nama mata uang Dolar AS. Simbol mata uang ISO USD dan nama mata uang Dolar AS peka huruf besar/kecil. Amazon Redshift hanya mendukung mata uang USD. String ekspresi input dapat menyertakan spasi antara simbol mata uang USD dan nilai numerik, misalnya '$ 123E2' atau '123E2 $'.
L	Simbol mata uang. Karakter simbol mata uang ini hanya dapat muncul sekali dalam frasa format. Anda tidak dapat menentukan karakter simbol mata uang berulang.
C	Simbol mata uang ISO. Karakter simbol mata uang ini hanya dapat muncul sekali dalam frasa format. Anda tidak dapat menentukan karakter simbol mata uang berulang.
T	Nama mata uang lengkap. Karakter simbol mata uang ini hanya dapat muncul sekali dalam frasa format. Anda tidak dapat menentukan karakter simbol mata uang berulang.
O	Simbol mata uang ganda. Anda tidak dapat menentukan karakter ini dalam frasa format.
U	Simbol mata uang ISO ganda. Anda tidak dapat menentukan karakter ini dalam frasa format.
A	Nama mata uang ganda lengkap. Anda tidak dapat menentukan karakter ini dalam frasa format.

Karakter pemformatan data untuk Signed Zone Decimal, pemformatan data numerik bergaya Teradata

Anda dapat menggunakan karakter berikut dalam frase format fungsi TEXT_TO_INT_ALT dan TEXT_TO_NUMERIC_ALT untuk nilai desimal yang dikategorikan ditandatangani.

Karakter terakhir dari string input	Konversi numerik
{atau 0	no... 0
A atau 1	no... 1
B atau 2	no... 2
C atau 3	no... 3
D atau 4	no... 4
E atau 5	No... 5
F atau 6	No... 6
G atau 7	No... 7
H atau 8	No... 8
Saya atau 9	No... 9
}	- n... 0
J	- n... 1
K	- n... 2
L	- n... 3
M	- n... 4
T	- n... 5
O	- n... 6

Karakter terakhir dari string input	Konversi numerik
P	- 7
Q	- n... 8
R	- n... 9

Fungsi tanggal dan waktu

Di bagian ini, Anda dapat menemukan informasi tentang fungsi skalar tanggal dan waktu yang didukung Amazon Redshift.

Topik

- Ringkasan fungsi tanggal dan waktu
- Fungsi tanggal dan waktu dalam transaksi
- Fungsi khusus node pemimpin yang tidak digunakan lagi
- + Operator (Penggabungan)
- Fungsi ADD_MONTHS
- Fungsi AT TIME ZONE
- Fungsi CONVERT_TIMEZONE
- Fungsi CURRENT_DATE
- Fungsi DATE_CMP
- Fungsi DATE_CMP_TIMESTAMP
- Fungsi DATE_CMP_TIMESTAMPTZ
- Fungsi DATEADD
- Fungsi DATEDIFF
- Fungsi DATE_PART
- Fungsi DATE_PART_YEAR
- Fungsi DATE_TRUNC
- Fungsi EKSTRAK
- fungsi GETDATE
- Fungsi INTERVAL_CMP

- Fungsi LAST_DAY
- Fungsi MONTHS_BETWEEN
- fungsi NEXT_DAY
- fungsi SYSDATE
- Fungsi TIMEOFDAY
- Fungsi TIMESTAMP_CMP
- Fungsi TIMESTAMP_CMP_DATE
- Fungsi TIMESTAMP_CMP_TIMESTAMPTZ
- Fungsi TIMESTAMPTZ_CMP
- Fungsi TIMESTAMPTZ_CMP_DATE
- Fungsi TIMESTAMPTZ_CMP_TIMESTAMP
- Fungsi TIMEZONE
- Fungsi TO_TIMESTAMP
- Fungsi TRUNC
- Bagian tanggal untuk fungsi tanggal atau stempel waktu

Ringkasan fungsi tanggal dan waktu

Fungsi	Sintaksis	Pengembalian
+ Operator (Penggabungan) Menggabungkan tanggal ke waktu di kedua sisi simbol + dan mengembalikan TIMESTAMP atau TIMESTAMPTZ.	tanggal+waktu	`TIMESTAMP` atau `TIMESTAMPTZ`
ADD_MONTHS Menambahkan jumlah bulan yang ditentukan ke tanggal atau stempel waktu.	ADD_MONTHS ({tanggal | cap waktu}, bilangan bulat)	`TIMESTAMP`
DI ZONA WAKTU	DI ZONA WAKTU 'zona waktu'	`TIMESTAMP` atau

Fungsi	Sintaksis	Pengembalian
Menentukan zona waktu yang akan digunakan dengan ekspresi TIMESTAMP atau TIMESTAMPTZ.		`TIMESTAMPZ`
CONVERT_TIMEZONE Mengkonversi stempel waktu dari satu zona waktu ke zona waktu lainnya.	CONVERT_TIMEZONE (['zona waktu',] 'zona waktu', stempel waktu)	`TIMESTAMP`
CURRENT_DATE Mengembalikan tanggal di zona waktu sesi saat ini (UTC secara default) untuk memulai transaksi saat ini.	CURRENT_DATE	`DATE`
DATE_CMP Membandingkan dua tanggal dan kembali `0` jika tanggal identik, `1` jika date1 lebih besar, dan *`-1`* jika date2 lebih besar.	DATE_CMP (tanggal1, tanggal2)	`INTEGER`
DATE_CMP_STEMPEL WAKTU Membandingkan tanggal dengan waktu dan kembali `0` jika nilainya identik, jika tanggal lebih besar dan `1` `-1` jika stempel waktu lebih besar.	DATE_CMP_TIMESTAMP (tanggal, stempel waktu)	`INTEGER`
DATE_CMP_TIMESTAMPTZ Membandingkan tanggal dan stempel waktu dengan zona waktu dan mengembalikan `0` jika nilainya identik, jika tanggal lebih besar dan `1` `-1` jika timestamptz lebih besar.	DATE_CMP_TIMESTAMPTZ (tanggal, cap waktu)	`INTEGER`
DATE_PART_YEAR Ekstrak tahun dari tanggal.	DATE_PART_YEAR (tanggal)	`INTEGER`

Fungsi	Sintaksis	Pengembalian
DATEADD Menambah tanggal atau waktu dengan interval tertentu.	DATEADD (datepart, interval, {tanggal | waktu | jadwal | cap waktu})	`TIMESTAMP` atau `TIME` atau `TIMETZ`
DATEDIFF Mengembalikan selisih antara dua tanggal atau waktu untuk bagian tanggal tertentu, seperti hari atau bulan.	DATEDIFF (datepart, {date | time | timetz | timestamp} , {tanggal | waktu | jadwal waktu})	`BIGINT`
DATE_PART Mengekstrak nilai bagian tanggal dari tanggal atau waktu.	DATE_PART (bagian tanggal, {tanggal | cap waktu})	`DOUBLE`
DATE_TRUNC Mempotong stempel waktu berdasarkan bagian tanggal.	DATE_TRUNC ('datepart', stempel waktu)	`TIMESTAMP`
EKSTRAK Mengekstrak bagian tanggal atau waktu dari stempel waktu, timestamptz, waktu, atau jadwal.	EKSTRAK (datepart DARI sumber)	`INTEGER or DOUBLE`
GETDATE Mengembalikan tanggal dan waktu saat ini di zona waktu sesi saat ini (UTC secara default). Tanda kurung diperlukan.	GETDATE ()	`TIMESTAMP`

Fungsi	Sintaksis	Pengembalian
INTERVAL_CMP Membandingkan dua interval dan kembali 0 jika intervalnya sama, 1 jika interval1 lebih besar, dan -1 jika interval2 lebih besar.	INTERVAL_CMP (interval1, interval2)	INTEGER
HARI TERAKHIR Mengembalikan tanggal hari terakhir bulan yang berisi tanggal.	LAST_DAY (tanggal)	DATE
BULAN_ANTARA Mengembalikan jumlah bulan antara dua tanggal.	MONTHS_BETWEEN (tanggal, tanggal)	FLOAT8
HARI BERIKUTNYA Mengembalikan tanggal contoh pertama hari yang lebih lambat dari tanggal.	NEXT_DAY (tanggal, hari)	DATE
SYSDATE Mengembalikan tanggal dan waktu di UTC untuk memulai transaksi saat ini.	SYSDATE	TIMESTAMP
WAKTUHARI Mengembalikan hari kerja, tanggal, dan waktu saat ini di zona waktu sesi saat ini (UTC secara default) sebagai nilai string.	WAKTUHARI ()	VARCHAR

Fungsi	Sintaksis	Pengembalian
TIMESTAMP_CMP Membandingkan dua stempel waktu dan kembali *0* jika stempel waktu sama, jika stempel waktu1 lebih besar, dan 1 jika stempel waktu2 lebih besar. *-1*	TIMESTAMP_CMP (cap waktu1, cap waktu2)	INTEGER
TIMESTAMP_CMP_DATE Membandingkan stempel waktu dengan tanggal dan mengembalikan *0* jika nilainya identik, jika stempel waktu lebih besar, dan *1* *-1* jika tanggal lebih besar.	TIMESTAMP_CMP_DATE (stempel waktu, tanggal)	INTEGER
TIMESTAMP_CMP_TIMESTAMPTZ Membandingkan stempel waktu dengan timestamp dengan zona waktu dan mengembalikan *0* jika nilainya sama, jika stempel waktu lebih besar, dan *1* jika timestamptz lebih besar. *-1*	TIMESTAMP_CMP_TIMESTAMPTZ (cap waktu, cap waktu)	INTEGER
STAMPTZ_CMP Membandingkan dua stempel waktu dengan nilai zona waktu dan mengembalikan *0* jika nilainya sama, jika timestamptz1 lebih besar, dan 1 jika timestamptz2 lebih besar. *-1*	TIMESTAMPTZ_CMP (stemamptz1, cap waktuz2)	INTEGER
TIMESTAMPTZ_CMP_DATE Membandingkan nilai stempel waktu dengan zona waktu dan tanggal dan mengembalikan *0* jika nilainya sama, jika timestamptz lebih besar, dan *1* jika tanggal lebih besar. *-1*	TIMESTAMPTZ_CMP_DATE (timestamptz, tanggal)	INTEGER

Fungsi	Sintaksis	Pengembalian
TIMESTAMPTZ_CMP_TIMESTAMP Membandingkan stempel waktu dengan zona waktu dengan stempel waktu dan mengembalikan *0* jika nilainya sama, jika timestamptz lebih besar, dan *1* jika stempel waktu lebih besar. *-1*	TIMESTAMPTZ_CMP_TIMESTAMP (cap waktu, stempel waktu)	`INTEGER`
ZONA WAKTU Mengembalikan timestamp untuk zona waktu tertentu dan nilai timestamp.	TIMEZONE ('zona waktu' {timestamp | timestamptz)	`TIMESTAMP` atau `TIMESTAMPTZ`
TO_TIMESTAMP Mengembalikan timestamp dengan zona waktu untuk format timestamp dan zona waktu yang ditentukan.	TO_TIMESTAMP ('stempel waktu', 'format')	`TIMESTAMPTZ`
BATANG Memangkas stempel waktu dan mengembalikan tanggal.	TRUNC (stempel waktu)	`DATE`

> **Note**
>
> Detik kabisat tidak dipertimbangkan dalam perhitungan waktu berlalu.

Fungsi tanggal dan waktu dalam transaksi

Ketika Anda menjalankan fungsi berikut dalam blok transaksi (BEGIN... END), fungsi mengembalikan tanggal mulai atau waktu transaksi saat ini, bukan awal dari pernyataan saat ini.

- SYSDATE
- TIMESTAMP

- CURRENT_DATE

Fungsi-fungsi berikut selalu mengembalikan tanggal mulai atau waktu pernyataan saat ini, bahkan ketika mereka berada dalam blok transaksi.

- GETDATE
- WAKTUHARI

Fungsi khusus node pemimpin yang tidak digunakan lagi

Fungsi tanggal berikut tidak digunakan lagi karena hanya berjalan pada node pemimpin. Untuk informasi selengkapnya, lihat Node—hanya fungsi utama.

- USIA. Gunakan Fungsi DATEDIFF sebagai gantinya.
- CURRENT_TIME. Gunakan fungsi GETDATE atau SYSDATE sebagai gantinya.
- CURRENT_TIMESTAMP. Gunakan fungsi GETDATE atau SYSDATE sebagai gantinya.
- WAKTU LOKAL. Gunakan fungsi GETDATE atau SYSDATE sebagai gantinya.
- STEMPEL WAKTU LOKAL. Gunakan fungsi GETDATE atau SYSDATE sebagai gantinya.
- TIDAK TERBATAS
- SEKARANG. Gunakan fungsi GETDATE atau SYSDATE sebagai gantinya. Jika Anda menggunakan fungsi NOW dalam tampilan terwujud, fungsi ini akan disetel ke stempel waktu pembuatan tampilan terwujud, bukan stempel waktu saat ini.

+ Operator (Penggabungan)

Menggabungkan TANGGAL ke TIME atau TIMETZ di kedua sisi simbol + dan mengembalikan TIMESTAMP atau TIMESTAMPTZ.

Sintaksis

```
date + {time | timetz}
```

Urutan argumen dapat dibalik. Misalnya, waktu+tanggal.

Argumen

tanggal

Kolom tipe data DATE atau ekspresi yang secara implisit mengevaluasi tipe. DATE

waktu

Kolom tipe data TIME atau ekspresi yang secara implisit mengevaluasi tipe. TIME

jadwal

Kolom tipe data TIMETZ atau ekspresi yang secara implisit mengevaluasi tipe. TIMETZ

Jenis pengembalian

TIMESTAMP jika masukan adalah tanggal+waktu.

TIMESTAMPTZ jika masukan adalah tanggal + jadwal.

Contoh

Contoh pengaturan

Untuk mengatur tabel TIME_TEST dan TIMETZ_TEST yang digunakan dalam contoh, gunakan perintah berikut.

```
create table time_test(time_val time);

insert into time_test values
('20:00:00'),
('00:00:00.5550'),
('00:58:00');

create table timetz_test(timetz_val timetz);

insert into timetz_test values
('04:00:00+00'),
('00:00:00.5550+00'),
('05:58:00+00');
```

Contoh dengan kolom waktu

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut menggabungkan tanggal literal dan kolom TIME_VAL.

```
select date '2000-01-02' + time_val as ts from time_test;

ts
---------------------
2000-01-02 20:00:00
2000-01-02 00:00:00.5550
2000-01-02 00:58:00
```

Contoh berikut menggabungkan tanggal literal dan literal waktu.

```
select date '2000-01-01' + time '20:00:00' as ts;

         ts
---------------------
 2000-01-01 20:00:00
```

Contoh berikut menggabungkan literal waktu dan tanggal literal.

```
select time '20:00:00' + date '2000-01-01' as ts;

         ts
---------------------
 2000-01-01 20:00:00
```

Contoh dengan kolom TIMETZ

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;
```

```
timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut menggabungkan tanggal literal dan kolom TIMETZ_VAL.

```
select date '2000-01-01' + timetz_val as ts from timetz_test;
ts
---------------------
2000-01-01 04:00:00+00
2000-01-01 00:00:00.5550+00
2000-01-01 05:58:00+00
```

Contoh berikut menggabungkan kolom TIMETZ_VAL dan tanggal literal.

```
select timetz_val + date '2000-01-01' as ts from timetz_test;
ts
---------------------
2000-01-01 04:00:00+00
2000-01-01 00:00:00.5550+00
2000-01-01 05:58:00+00
```

Contoh berikut menggabungkan literal DATE dan literal TIMETZ. Contoh mengembalikan TIMESTAMPTZ yang berada di zona waktu UTC secara default. UTC adalah 8 jam di depan PST, jadi hasilnya 8 jam lebih awal dari waktu input.

```
select date '2000-01-01' + timetz '20:00:00 PST' as ts;

          ts
------------------------
 2000-01-02 04:00:00+00
```

Fungsi ADD_MONTHS

ADD_MONTHS menambahkan jumlah bulan yang ditentukan ke nilai atau ekspresi tanggal atau stempel waktu. DATEADDFungsi ini menyediakan fungsionalitas serupa.

Sintaksis

```
ADD_MONTHS( {date | timestamp}, integer)
```

Argumen

tanggal | stempel waktu

Kolom tipe data DATE atau TIMESTAMP atau ekspresi yang secara implisit mengevaluasi ke atau tipe. DATE TIMESTAMP Jika tanggal adalah hari terakhir bulan itu, atau jika bulan yang dihasilkan lebih pendek, fungsi mengembalikan hari terakhir bulan dalam hasilnya. Untuk tanggal lain, hasilnya berisi nomor hari yang sama dengan ekspresi tanggal.

bilangan bulat

Nilai tipe dataINTEGER. Gunakan angka negatif untuk mengurangi bulan dari tanggal.

Jenis pengembalian

TIMESTAMP

Contoh

Query berikut menggunakan fungsi ADD_MONTHS di dalam fungsi TRUNC. Fungsi TRUNC menghapus waktu hari dari hasil ADD_MONTHS. Fungsi ADD_MONTHS menambahkan 12 bulan ke setiap nilai dari kolom CALDATE. Nilai dalam kolom CALDATE adalah tanggal.

```
select distinct trunc(add_months(caldate, 12)) as calplus12,
trunc(caldate) as cal
from date
order by 1 asc;

 calplus12  |    cal
------------+------------
 2009-01-01 | 2008-01-01
 2009-01-02 | 2008-01-02
 2009-01-03 | 2008-01-03
...
(365 rows)
```

Contoh berikut menggunakan fungsi ADD_MONTHS untuk menambahkan 1 bulan ke stempel waktu.

```
select add_months('2008-01-01 05:07:30', 1);

add_months
---------------------
2008-02-01 05:07:30
```

Contoh berikut menunjukkan perilaku ketika fungsi ADD_MONTHS beroperasi pada tanggal dengan bulan yang memiliki jumlah hari yang berbeda. Contoh ini menunjukkan bagaimana fungsi menangani penambahan 1 bulan ke 31 Maret dan menambahkan 1 bulan ke 30 April. April memiliki 30 hari, jadi menambahkan 1 bulan ke 31 Maret menghasilkan 30 April. Mei memiliki 31 hari, jadi menambahkan 1 bulan ke 30 April menghasilkan 31 Mei.

```
select add_months('2008-03-31',1);

add_months
---------------------
2008-04-30 00:00:00

select add_months('2008-04-30',1);

add_months
---------------------
2008-05-31 00:00:00
```

Fungsi AT TIME ZONE

AT TIME ZONE menentukan zona waktu mana yang akan digunakan dengan ekspresi TIMESTAMP atau TIMESTAMPTZ.

Sintaksis

```
AT TIME ZONE 'timezone'
```

Argumen

zona waktu

TIMEZONEUntuk nilai pengembalian. Zona waktu dapat ditentukan sebagai nama zona waktu (seperti **'Africa/Kampala'** atau**'Singapore'**) atau sebagai singkatan zona waktu (seperti **'UTC'** atau**'PDT'**).

Untuk melihat daftar nama zona waktu yang didukung, jalankan perintah berikut.

```
select pg_timezone_names();
```

Untuk melihat daftar singkatan zona waktu yang didukung, jalankan perintah berikut.

```
select pg_timezone_abbrevs();
```

Untuk informasi selengkapnya dan contoh tambahan, lihat Catatan penggunaan zona waktu.

Jenis pengembalian

TIMESTAMPTZ bila digunakan dengan ekspresi TIMESTAMP. TIMESTAMP bila digunakan dengan ekspresi TIMESTAMPTZ.

Contoh

Contoh berikut mengkonversi nilai timestamp tanpa zona waktu dan menafsirkannya sebagai waktu MST (UTC+7 di POSIX). Contoh mengembalikan nilai tipe data TIMESTAMPTZ untuk zona waktu UTC. Jika Anda mengonfigurasi zona waktu default ke zona waktu selain UTC, Anda mungkin melihat hasil yang berbeda.

```
SELECT TIMESTAMP '2001-02-16 20:38:40' AT TIME ZONE 'MST';

timezone
------------------------
2001-02-17 03:38:40+00
```

Contoh berikut mengambil timestamp masukan dengan nilai zona waktu di mana zona waktu yang ditentukan adalah EST (UTC+5 di POSIX) dan mengubahnya menjadi MST (UTC+7 di POSIX). Contoh mengembalikan nilai tipe data TIMESTAMP.

```
SELECT TIMESTAMPTZ '2001-02-16 20:38:40-05' AT TIME ZONE 'MST';

timezone
------------------------
2001-02-16 18:38:40
```

Fungsi CONVERT_TIMEZONE

CONVERT_TIMEZONE mengonversi stempel waktu dari satu zona waktu ke zona waktu lainnya. Fungsi ini secara otomatis menyesuaikan waktu musim panas.

Sintaksis

```
CONVERT_TIMEZONE( ['source_timezone',] 'target_timezone', 'timestamp')
```

Argumen

source_timezone

(Opsional) Zona waktu stempel waktu saat ini. Defaultnya adalah UTC. Untuk informasi selengkapnya, lihat Catatan penggunaan zona waktu.

target_zona waktu

Zona waktu untuk stempel waktu baru. Untuk informasi selengkapnya, lihat Catatan penggunaan zona waktu.

stempel waktu

Kolom timestamp atau ekspresi yang secara implisit mengkonversi ke stempel waktu.

Jenis pengembalian

TIMESTAMP

Catatan penggunaan zona waktu

source_timezone atau target_timezone dapat ditentukan sebagai nama zona waktu (seperti " atau 'Singapura') Africa/Kampala atau sebagai singkatan zona waktu (seperti 'UTC' atau 'PDT'). Anda tidak perlu mengubah nama zona waktu menjadi nama atau singkatan menjadi singkatan. Misalnya, Anda dapat memilih stempel waktu dari nama zona waktu sumber 'Singapura' dan mengubahnya menjadi stempel waktu dalam singkatan zona waktu 'PDT'.

> **Note**
>
> Hasil penggunaan nama zona waktu atau singkatan zona waktu dapat berbeda karena waktu musiman lokal, seperti waktu musim panas.

Menggunakan nama zona waktu

Untuk melihat daftar nama zona waktu saat ini dan lengkap, jalankan perintah berikut.

```
select pg_timezone_names();
```

Setiap baris berisi string yang dipisahkan koma dengan nama zona waktu, singkatan, offset UTC, dan indikator jika zona waktu mengamati waktu musim panas (atau). `t` `f` Misalnya, cuplikan berikut menunjukkan dua baris yang dihasilkan. Baris pertama adalah zona waktu`Antarctica/South Pole`, singkatan`NZDT`, dengan `13:00:00` offset dari UTC, dan `f` untuk menunjukkan itu tidak mengamati waktu musim panas. Baris kedua adalah zona waktu`Europe/Paris`, singkatan`CET`, dengan `01:00:00` offset dari UTC, dan `f` untuk menunjukkan itu mengamati waktu musim panas.

```
pg_timezone_names
------------------
(Antarctica/South_Pole,NZDT,13:00:00,t)
(Europe/Paris,CET,01:00:00,f)
```

Jalankan pernyataan SQL untuk mendapatkan seluruh daftar dan menemukan nama zona waktu. Sekitar 600 baris dikembalikan. Meskipun beberapa nama zona waktu yang dikembalikan adalah inisialisme atau akronim yang dikapitalisasi (misalnya; GB, RRC, ROK), fungsi CONVERT_TIMEZONE memperlakukannya sebagai nama zona waktu, bukan singkatan zona waktu.

Jika Anda menentukan zona waktu menggunakan nama zona waktu, CONVERT_TIMEZONE secara otomatis menyesuaikan waktu musim panas (DST), atau protokol musiman lokal lainnya, seperti Waktu Musim Panas, Waktu Standar, atau Waktu Musim Dingin, yang berlaku untuk zona waktu tersebut selama tanggal dan waktu yang ditentukan oleh 'stempel waktu'. Misalnya, 'Europe/London' mewakili UTC di musim dingin dan menambahkan satu jam di musim panas. Perhatikan bahwa Amazon Redshift menggunakan Database Zona Waktu IANA sebagai sumber otoritatif spesifikasi zona waktu.

Menggunakan singkatan zona waktu

Untuk melihat daftar singkatan zona waktu saat ini dan lengkap, jalankan perintah berikut.

```
select pg_timezone_abbrevs();
```

Hasilnya berisi string yang dipisahkan koma dengan singkatan zona waktu, offset UTC, dan indikator jika zona waktu mengamati waktu musim panas (atau). `t` `f` Misalnya, cuplikan berikut menunjukkan

dua baris yang dihasilkan. Baris pertama berisi singkatan untuk Pacific Daylight TimePDT, dengan `-07:00:00` offset dari UTC, dan `t` untuk menunjukkan itu mengamati waktu musim panas. Baris kedua berisi singkatan untuk Waktu Standar PasifikPST, dengan `-08:00:00` offset dari UTC, dan `f` untuk menunjukkan itu tidak mengamati waktu musim panas.

```
pg_timezone_abbrevs
--------------------
(PDT,-07:00:00,t)
(PST,-08:00:00,f)
```

Jalankan pernyataan SQL untuk mendapatkan seluruh daftar dan temukan singkatan berdasarkan indikator offset dan daylight saving time. Sekitar 200 baris dikembalikan.

Singkatan zona waktu mewakili offset tetap dari UTC. Jika Anda menentukan zona waktu menggunakan singkatan zona waktu, CONVERT_TIMEZONE menggunakan offset tetap dari UTC dan tidak menyesuaikan untuk protokol musiman lokal apa pun.

Menggunakan POSIX-style format

Spesifikasi zona POSIX-style waktu adalah dalam bentuk StdOffset atau StdOffsetDST, di mana STD adalah singkatan zona waktu, offset adalah offset numerik dalam jam barat dari UTC, dan DST adalah singkatan zona musim panas opsional. Daylight saving time diasumsikan satu jam lebih cepat dari offset yang diberikan.

POSIX-style Format zona waktu menggunakan offset positif di sebelah barat Greenwich, berbeda dengan ISO-8601 konvensi, yang menggunakan offset positif di timur Greenwich.

Berikut ini adalah contoh zona POSIX-style waktu:

- PST8
- PST8PDT
- EST5
- EST5EDT

> **Note**
>
> Amazon Redshift tidak memvalidasi spesifikasi zona POSIX-style waktu, sehingga dimungkinkan untuk mengatur zona waktu ke nilai yang tidak valid. Misalnya, perintah berikut tidak mengembalikan kesalahan, meskipun menetapkan zona waktu ke nilai yang tidak valid.

```
set timezone to ‘xxx36’;
```

Contoh

Banyak contoh menggunakan kumpulan data sampel TICKIT. Untuk informasi selengkapnya, lihat Contoh database.

Contoh berikut mengkonversi nilai timestamp dari zona waktu UTC default untuk PST.

```
select convert_timezone('PST', '2008-08-21 07:23:54');

 convert_timezone
------------------------
2008-08-20 23:23:54
```

Contoh berikut mengkonversi nilai timestamp dalam kolom LISTTIME dari zona waktu UTC default ke PST. Meskipun stempel waktu berada dalam periode waktu siang hari, itu diubah menjadi waktu standar karena zona waktu target ditentukan sebagai singkatan (PST).

```
select listtime, convert_timezone('PST', listtime) from listing
where listid = 16;

     listtime       |   convert_timezone
--------------------+-------------------
2008-08-24 09:36:12     2008-08-24 01:36:12
```

Contoh berikut mengonversi timestamp kolom LISTTIME dari zona waktu UTC default ke zona waktu. US/Pacific Zona waktu target menggunakan nama zona waktu, dan stempel waktu berada dalam periode waktu siang hari, sehingga fungsi mengembalikan waktu siang hari.

```
select listtime, convert_timezone('US/Pacific', listtime) from listing
where listid = 16;

     listtime       |   convert_timezone
--------------------+---------------------
2008-08-24 09:36:12 | 2008-08-24 02:36:12
```

Contoh berikut mengkonversi string timestamp dari EST ke PST:

```
select convert_timezone('EST', 'PST', '20080305 12:25:29');

 convert_timezone
-------------------
2008-03-05 09:25:29
```

Contoh berikut mengonversi stempel waktu ke Waktu Standar Timur AS karena zona waktu target menggunakan nama zona waktu (America/New_York) dan stempel waktu berada dalam periode waktu standar.

```
select convert_timezone('America/New_York', '2013-02-01 08:00:00');

 convert_timezone
---------------------
2013-02-01 03:00:00
(1 row)
```

Contoh berikut mengubah stempel waktu menjadi US Eastern Daylight Time karena zona waktu target menggunakan nama zona waktu (America/New_York) dan stempel waktu berada dalam periode waktu siang hari.

```
select convert_timezone('America/New_York', '2013-06-01 08:00:00');

 convert_timezone
---------------------
2013-06-01 04:00:00
(1 row)
```

Contoh berikut menunjukkan penggunaan offset.

```
SELECT CONVERT_TIMEZONE('GMT','NEWZONE +2','2014-05-17 12:00:00') as newzone_plus_2,
CONVERT_TIMEZONE('GMT','NEWZONE-2:15','2014-05-17 12:00:00') as newzone_minus_2_15,
CONVERT_TIMEZONE('GMT','America/Los_Angeles+2','2014-05-17 12:00:00') as la_plus_2,
CONVERT_TIMEZONE('GMT','GMT+2','2014-05-17 12:00:00') as gmt_plus_2;

   newzone_plus_2    | newzone_minus_2_15  |      la_plus_2      |     gmt_plus_2
---------------------+---------------------+---------------------+---------------------
2014-05-17 10:00:00 | 2014-05-17 14:15:00 | 2014-05-17 10:00:00 | 2014-05-17 10:00:00
(1 row)
```

Fungsi CURRENT_DATE

CURRENT_DATE mengembalikan tanggal di zona waktu sesi saat ini (UTC secara default) dalam format default:. YYYY-MM-DD

> **Note**
>
> CURRENT_DATE mengembalikan tanggal mulai untuk transaksi saat ini, bukan untuk awal pernyataan saat ini. Pertimbangkan skenario di mana Anda memulai transaksi yang berisi beberapa pernyataan pada 10/01 /08 23:59, dan pernyataan yang berisi CURRENT_DATE berjalan pada /08 00:00. 10/02 CURRENT_DATE kembali`10/01/08`, tidak. `10/02/08`

Sintaksis

```
CURRENT_DATE
```

Jenis pengembalian

DATE

Contoh

Contoh berikut mengembalikan tanggal saat ini (di Wilayah AWS mana fungsi berjalan).

```
select current_date;

   date
------------
2008-10-01
```

Contoh berikut membuat tabel, menyisipkan baris di mana default kolom `todays_date` adalah CURRENT_DATE, dan kemudian memilih semua baris dalam tabel.

```
CREATE TABLE insert_dates(
    label varchar(128) NOT NULL,
    todays_date DATE DEFAULT CURRENT_DATE);
```

```
INSERT INTO insert_dates(label)
VALUES('Date row inserted');

SELECT * FROM insert_dates;


 label            | todays_date
------------------+-------------
Date row inserted | 2023-05-10
```

Fungsi DATE_CMP

DATE_CMP membandingkan dua tanggal. Fungsi kembali `0` jika tanggal identik, 1 jika date1 lebih besar, dan `-1` jika date2 lebih besar.

Sintaksis

```
DATE_CMP(date1, date2)
```

Argumen

tanggal1

Kolom tipe data `DATE` atau ekspresi yang mengevaluasi `DATE` tipe.

tanggal2

Kolom tipe data `DATE` atau ekspresi yang mengevaluasi `DATE` tipe.

Jenis pengembalian

INTEGER

Contoh

Kueri berikut membandingkan nilai DATE di kolom CALDATE dengan tanggal 4 Januari 2008 dan mengembalikan apakah nilai di CALDATE adalah sebelum (`-1`), sama dengan (), atau setelah (`01`) 4 Januari 2008:

```
select caldate, '2008-01-04',
date_cmp(caldate,'2008-01-04')
```

```
from date
order by dateid
limit 10;

 caldate   |  ?column?  | date_cmp
-----------+------------+----------
2008-01-01 | 2008-01-04 |       -1
2008-01-02 | 2008-01-04 |       -1
2008-01-03 | 2008-01-04 |       -1
2008-01-04 | 2008-01-04 |        0
2008-01-05 | 2008-01-04 |        1
2008-01-06 | 2008-01-04 |        1
2008-01-07 | 2008-01-04 |        1
2008-01-08 | 2008-01-04 |        1
2008-01-09 | 2008-01-04 |        1
2008-01-10 | 2008-01-04 |        1
(10 rows)
```

Fungsi DATE_CMP_TIMESTAMP

DATE_CMP_TIMESTAMP membandingkan tanggal dengan stempel waktu dan mengembalikan `0` jika nilainya identik, jika tanggal lebih besar secara kronologis dan jika stempel waktu lebih besar. `1` `-1`

Sintaksis

```
DATE_CMP_TIMESTAMP(date, timestamp)
```

Argumen

tanggal

Kolom tipe data `DATE` atau ekspresi yang mengevaluasi `DATE` tipe.

stempel waktu

Kolom tipe data `TIMESTAMP` atau ekspresi yang mengevaluasi `TIMESTAMP` tipe.

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan tanggal dengan 2008-06-18 LISTTIME. Nilai kolom LISTTIME adalah stempel waktu. Daftar yang dibuat sebelum tanggal ini kembali1; daftar yang dibuat setelah tanggal ini kembali-1.

```
select listid, '2008-06-18', listtime,
date_cmp_timestamp('2008-06-18', listtime)
from listing
order by 1, 2, 3, 4
limit 10;

 listid |  ?column?  |      listtime       | date_cmp_timestamp
--------+------------+---------------------+--------------------
      1 | 2008-06-18 | 2008-01-24 06:43:29 |                  1
      2 | 2008-06-18 | 2008-03-05 12:25:29 |                  1
      3 | 2008-06-18 | 2008-11-01 07:35:33 |                 -1
      4 | 2008-06-18 | 2008-05-24 01:18:37 |                  1
      5 | 2008-06-18 | 2008-05-17 02:29:11 |                  1
      6 | 2008-06-18 | 2008-08-15 02:08:13 |                 -1
      7 | 2008-06-18 | 2008-11-15 09:38:15 |                 -1
      8 | 2008-06-18 | 2008-11-09 05:07:30 |                 -1
      9 | 2008-06-18 | 2008-09-09 08:03:36 |                 -1
     10 | 2008-06-18 | 2008-06-17 09:44:54 |                  1
(10 rows)
```

Fungsi DATE_CMP_TIMESTAMPTZ

DATE_CMP_TIMESTAMPTZ membandingkan tanggal dengan stempel waktu dengan zona waktu dan mengembalikan 0 jika nilainya identik, jika tanggal lebih besar secara kronologis dan jika timestamptz lebih besar. 1 -1

Sintaksis

```
DATE_CMP_TIMESTAMPTZ(date, timestamptz)
```

Argumen

tanggal

Kolom tipe data DATE atau ekspresi yang secara implisit mengevaluasi tipe. DATE

timestamptz

Kolom tipe data TIMESTAMPTZ atau ekspresi yang secara implisit mengevaluasi tipe. TIMESTAMPTZ

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan tanggal dengan 2008-06-18 LISTTIME. Daftar yang dibuat sebelum tanggal ini kembali1; daftar yang dibuat setelah tanggal ini kembali-1.

```
select listid, '2008-06-18', CAST(listtime AS timestamptz),
date_cmp_timestamptz('2008-06-18', CAST(listtime AS timestamptz))
from listing
order by 1, 2, 3, 4
limit 10;

 listid |  ?column?  |      timestamptz       | date_cmp_timestamptz
--------+------------+------------------------+----------------------
      1 | 2008-06-18 | 2008-01-24 06:43:29+00 |                    1
      2 | 2008-06-18 | 2008-03-05 12:25:29+00 |                    1
      3 | 2008-06-18 | 2008-11-01 07:35:33+00 |                   -1
      4 | 2008-06-18 | 2008-05-24 01:18:37+00 |                    1
      5 | 2008-06-18 | 2008-05-17 02:29:11+00 |                    1
      6 | 2008-06-18 | 2008-08-15 02:08:13+00 |                   -1
      7 | 2008-06-18 | 2008-11-15 09:38:15+00 |                   -1
      8 | 2008-06-18 | 2008-11-09 05:07:30+00 |                   -1
      9 | 2008-06-18 | 2008-09-09 08:03:36+00 |                   -1
     10 | 2008-06-18 | 2008-06-17 09:44:54+00 |                    1
(10 rows)
```

Fungsi DATEADD

Menambah nilai DATE, TIME, TIMETZ, atau TIMESTAMP dengan interval tertentu.

Sintaksis

```
DATEADD( datepart, interval, {date|time|timetz|timestamp} )
```

Argumen

datepart

Bagian tanggal (tahun, bulan, hari, atau jam, misalnya) tempat fungsi beroperasi. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.

interval

Integer yang menentukan interval (jumlah hari, misalnya) untuk ditambahkan ke ekspresi target. Sebuah integer negatif mengurangi interval.

tanggal | waktu | timetz | stempel waktu

Kolom TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP atau ekspresi yang secara implisit mengkonversi ke TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP. Ekspresi DATE, TIME, TIMETZ, atau TIMESTAMP harus berisi bagian tanggal yang ditentukan.

Jenis pengembalian

TIMESTAMP atau TIME atau TIMETZ tergantung pada tipe data input.

Contoh dengan kolom DATE

Contoh berikut menambahkan 30 hari untuk setiap tanggal di bulan November yang ada di tabel DATE.

```
select dateadd(day,30,caldate) as novplus30
from date
where month='NOV'
order by dateid;

novplus30
---------------------
2008-12-01 00:00:00
2008-12-02 00:00:00
2008-12-03 00:00:00
...
(30 rows)
```

Contoh berikut menambahkan 18 bulan ke nilai tanggal literal.

```
select dateadd(month,18,'2008-02-28');
```

```
date_add
---------------------
2009-08-28 00:00:00
(1 row)
```

Nama kolom default untuk fungsi DATEADD adalah DATE_ADD. Stempel waktu default untuk nilai tanggal adalah. `00:00:00`

Contoh berikut menambahkan 30 menit ke nilai tanggal yang tidak menentukan stempel waktu.

```
select dateadd(m,30,'2008-02-28');

date_add
---------------------
2008-02-28 00:30:00
(1 row)
```

Anda dapat memberi nama bagian tanggal secara lengkap atau menyingkatnya. Dalam hal ini, m berarti menit, bukan bulan.

Contoh dengan kolom TIME

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut menambahkan 5 menit untuk setiap TIME_VAL dalam tabel TIME_TEST.

```
select dateadd(minute,5,time_val) as minplus5 from time_test;

minplus5
---------------
20:05:00
```

```
00:05:00.5550
01:03:00
```

Contoh berikut menambahkan 8 jam ke nilai waktu literal.

```
select dateadd(hour, 8, time '13:24:55');

date_add
---------------
21:24:55
```

Contoh berikut menunjukkan kapan waktu berjalan lebih dari 24:00:00 atau di bawah 00:00:00.

```
select dateadd(hour, 12, time '13:24:55');

date_add
---------------
01:24:55
```

Contoh dengan kolom TIMETZ

Nilai output dalam contoh ini ada di UTC yang merupakan zona waktu default.

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut menambahkan 5 menit untuk setiap TIMETZ_VAL dalam tabel TIMETZ_TEST.

```
select dateadd(minute,5,timetz_val) as minplus5_tz from timetz_test;

minplus5_tz
---------------
04:05:00+00
```

```
00:05:00.5550+00
06:03:00+00
```

Contoh berikut menambahkan 2 jam ke nilai timetz literal.

```
select dateadd(hour, 2, timetz '13:24:55 PST');

date_add
---------------
23:24:55+00
```

Contoh dengan kolom TIMESTAMP

Nilai output dalam contoh ini ada di UTC yang merupakan zona waktu default.

Contoh tabel berikut TIMESTAMP_TEST memiliki kolom TIMESTAMP_VAL (tipe TIMESTAMP) dengan tiga nilai disisipkan.

```
SELECT timestamp_val FROM timestamp_test;

timestamp_val
------------------
1988-05-15 10:23:31
2021-03-18 17:20:41
2023-06-02 18:11:12
```

Contoh berikut menambahkan 20 tahun hanya ke nilai TIMESTAMP_VAL di TIMESTAMP_TEST dari sebelum tahun 2000.

```
SELECT dateadd(year,20,timestamp_val)
FROM timestamp_test
WHERE timestamp_val < to_timestamp('2000-01-01 00:00:00', 'YYYY-MM-DD HH:MI:SS');

date_add
---------------
2008-05-15 10:23:31
```

Contoh berikut menambahkan 5 detik ke nilai stempel waktu literal yang ditulis tanpa indikator detik.

```
SELECT dateadd(second, 5, timestamp '2001-06-06');
```

```
date_add
---------------
2001-06-06 00:00:05
```

Catatan penggunaan

Fungsi DATEADD (bulan,...) dan ADD_MONTHS menangani tanggal yang jatuh pada akhir bulan secara berbeda:

- ADD_MONTHS: Jika tanggal yang Anda tambahkan adalah hari terakhir bulan itu, hasilnya selalu hari terakhir dari bulan hasil, terlepas dari panjang bulan. Misalnya, 30 April+1 bulan adalah 31 Mei.

  ```
  select add_months('2008-04-30',1);

  add_months
  ---------------------
  2008-05-31 00:00:00
  (1 row)
  ```

- DATEADD: Jika ada lebih sedikit hari pada tanggal yang Anda tambahkan daripada di bulan hasil, hasilnya adalah hari yang sesuai dari bulan hasil, bukan hari terakhir bulan itu. Misalnya, 30 April+1 bulan adalah 30 Mei.

  ```
  select dateadd(month,1,'2008-04-30');

  date_add
  ---------------------
  2008-05-30 00:00:00
  (1 row)
  ```

Fungsi DATEADD menangani tanggal tahun kabisat 02-29 secara berbeda saat menggunakan dateadd (month, 12,...) atau dateadd (year, 1,...).

```
select dateadd(month,12,'2016-02-29');

date_add
---------------------
2017-02-28 00:00:00

select dateadd(year, 1, '2016-02-29');
```

```
date_add
---------------------
2017-03-01 00:00:00
```

Fungsi DATEDIFF

DATEDIFF mengembalikan perbedaan antara bagian tanggal dari dua ekspresi tanggal atau waktu.

Sintaksis

```
DATEDIFF( datepart, {date|time|timetz|timestamp}, {date|time|timetz|timestamp} )
```

Argumen

datepart

Bagian spesifik dari nilai tanggal atau waktu (tahun, bulan, atau hari, jam, menit, detik, milidetik, atau mikrodetik) tempat fungsi beroperasi. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.

Secara khusus, DATEDIFF menentukan jumlah batas bagian tanggal yang disilangkan antara dua ekspresi. Misalnya, anggaplah Anda menghitung perbedaan tahun antara dua tanggal, `12-31-2008` dan`01-01-2009`. Dalam hal ini, fungsi mengembalikan 1 tahun meskipun fakta bahwa tanggal-tanggal ini hanya terpisah satu hari. Jika Anda menemukan perbedaan jam antara dua stempel waktu, `01-01-2009 8:30:00` dan`01-01-2009 10:00:00`, hasilnya adalah 2 jam. Jika Anda menemukan perbedaan jam antara dua stempel waktu, `8:30:00` dan`10:00:00`, hasilnya adalah 2 jam.

tanggal | waktu | timetz | stempel waktu

Kolom atau ekspresi TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP yang secara implisit dikonversi ke TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP. Ekspresi harus berisi tanggal atau bagian waktu yang ditentukan. Jika tanggal atau waktu kedua lebih lambat dari tanggal atau waktu pertama, hasilnya positif. Jika tanggal atau waktu kedua lebih awal dari tanggal atau waktu pertama, hasilnya negatif.

Jenis pengembalian

BIGINT

Contoh dengan kolom DATE

Contoh berikut menemukan perbedaan, dalam jumlah minggu, antara dua nilai tanggal literal.

```
select datediff(week,'2009-01-01','2009-12-31') as numweeks;

numweeks
----------
52
(1 row)
```

Contoh berikut menemukan perbedaan, dalam jam, antara dua nilai tanggal literal. Ketika Anda tidak memberikan nilai waktu untuk tanggal, defaultnya adalah 00:00:00.

```
select datediff(hour, '2023-01-01', '2023-01-03 05:04:03');

date_diff
----------
53
(1 row)
```

Contoh berikut menemukan perbedaan, dalam hari, antara dua nilai TIMESTAMETZ literal.

```
Select datediff(days, 'Jun 1,2008  09:59:59 EST', 'Jul 4,2008  09:59:59 EST')

date_diff
----------
33
```

Contoh berikut menemukan perbedaan, dalam hari, antara dua tanggal dalam baris tabel yang sama.

```
select * from date_table;

start_date |   end_date
-----------+-----------
2009-01-01 | 2009-03-23
2023-01-04 | 2024-05-04
(2 rows)

select datediff(day, start_date, end_date) as duration from date_table;

duration
```

```
---------
      81
     486
(2 rows)
```

Contoh berikut menemukan perbedaan, dalam jumlah kuartal, antara nilai literal di masa lalu dan tanggal hari ini. Contoh ini mengasumsikan bahwa tanggal saat ini adalah 5 Juni 2008. Anda dapat memberi nama bagian tanggal secara lengkap atau menyingkatnya. Nama kolom default untuk fungsi DATEDIFF adalah DATE_DIFF.

```
select datediff(qtr, '1998-07-01', current_date);

date_diff
-----------
40
(1 row)
```

Contoh berikut bergabung dengan tabel PENJUALAN dan DAFTAR untuk menghitung berapa hari setelah mereka terdaftar, tiket apa pun dijual untuk daftar 1000 hingga 1005. Penantian terpanjang untuk penjualan daftar ini adalah 15 hari, dan yang terpendek kurang dari satu hari (0 hari).

```
select priceperticket,
datediff(day, listtime, saletime) as wait
from sales, listing where sales.listid = listing.listid
and sales.listid between 1000 and 1005
order by wait desc, priceperticket desc;

priceperticket | wait
---------------+------
 96.00         |   15
 123.00        |   11
 131.00        |    9
 123.00        |    6
 129.00        |    4
 96.00         |    4
 96.00         |    0
(7 rows)
```

Contoh ini menghitung jumlah rata-rata jam penjual menunggu semua penjualan tiket.

```
select avg(datediff(hours, listtime, saletime)) as avgwait
```

```
from sales, listing
where sales.listid = listing.listid;

avgwait
---------
465
(1 row)
```

Contoh dengan kolom TIME

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut menemukan perbedaan jumlah jam antara kolom TIME_VAL dan literal waktu.

```
select datediff(hour, time_val, time '15:24:45') from time_test;

 date_diff
-----------
        -5
        15
        15
```

Contoh berikut menemukan perbedaan jumlah menit antara dua nilai waktu literal.

```
select datediff(minute, time '20:00:00', time '21:00:00') as nummins;

nummins
----------
60
```

Contoh dengan kolom TIMETZ

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut menemukan perbedaan jumlah jam, antara literal TIMETZ dan timetz_val.

```
select datediff(hours, timetz '20:00:00 PST', timetz_val) as numhours from timetz_test;

numhours
----------
0
-4
1
```

Contoh berikut menemukan perbedaan jumlah jam, antara dua nilai TIMETZ literal.

```
select datediff(hours, timetz '20:00:00 PST', timetz '00:58:00 EST') as numhours;

numhours
----------
1
```

Fungsi DATE_PART

DATE_PART mengekstrak nilai bagian tanggal dari ekspresi. DATE_PART adalah sinonim dari fungsi PGDATE_PART.

Sintaksis

```
DATE_PART(datepart, {date|timestamp})
```

Argumen

datepart

Pengidentifikasi literal atau string dari bagian tertentu dari nilai tanggal (misalnya, tahun, bulan, atau hari) di mana fungsi beroperasi. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.

{tanggal | stempel waktu}

Kolom tanggal, kolom stempel waktu, atau ekspresi yang secara implisit mengkonversi ke tanggal atau stempel waktu. Kolom atau ekspresi dalam tanggal atau stempel waktu harus berisi bagian tanggal yang ditentukan dalam datepart.

Jenis pengembalian

DOUBLE

Contoh

Nama kolom default untuk fungsi DATE_PART adalah. `pgdate_part`

Untuk informasi selengkapnya tentang data yang digunakan dalam beberapa contoh ini, lihatDatabase sampel.

Contoh berikut menemukan menit dari stempel waktu literal.

```
SELECT DATE_PART(minute, timestamp '20230104 04:05:06.789');

pgdate_part
-----------
          5
```

Contoh berikut menemukan nomor minggu dari literal stempel waktu. Perhitungan angka minggu mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(week, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         18
```

Contoh berikut menemukan hari dalam sebulan dari stempel waktu literal.

```
SELECT DATE_PART(day, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          2
```

Contoh berikut menemukan hari dalam seminggu dari stempel waktu literal. Perhitungan angka hari dalam seminggu adalah bilangan bulat dari 0-6, dimulai dengan hari Minggu.

```
SELECT DATE_PART(dayofweek, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          1
```

Contoh berikut menemukan abad dari stempel waktu literal. Perhitungan abad mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(century, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         21
```

Contoh berikut menemukan milenium dari literal stempel waktu. Perhitungan milenium mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(millennium, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          3
```

Contoh berikut menemukan mikrodetik dari literal stempel waktu. Perhitungan mikrodetik mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(microsecond, timestamp '20220502 04:05:06.789');

pgdate_part
```

```
-----------
    789000
```

Contoh berikut menemukan bulan dari tanggal literal.

```
SELECT DATE_PART(month, date '20220502');

pgdate_part
-----------
          5
```

Contoh berikut menerapkan fungsi DATE_PART ke kolom dalam tabel.

```
SELECT date_part(w, listtime) AS weeks, listtime
FROM listing
WHERE listid=10


weeks |      listtime
------+---------------------
 25   | 2008-06-17 09:44:54
(1 row)
```

Anda dapat memberi nama bagian tanggal secara lengkap atau menyingkatnya; dalam hal ini, w berarti minggu.

Bagian tanggal hari minggu mengembalikan bilangan bulat dari 0-6, dimulai dengan hari Minggu. Gunakan DATE_PART dengan dow (DAYOFWEEK) untuk melihat acara pada hari Sabtu.

```
SELECT date_part(dow, starttime) AS dow, starttime
FROM event
WHERE date_part(dow, starttime)=6
ORDER BY 2,1;

 dow |      starttime
-----+---------------------
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
...
(1147 rows)
```

Fungsi DATE_PART_YEAR

Fungsi DATE_PART_YEAR mengekstrak tahun dari tanggal.

Sintaksis

```
DATE_PART_YEAR(date)
```

Pendapat

tanggal

Kolom tipe data DATE atau ekspresi yang secara implisit mengevaluasi tipe. DATE

Jenis pengembalian

INTEGER

Contoh

Contoh berikut menemukan tahun dari tanggal literal.

```
SELECT DATE_PART_YEAR(date '20220502 04:05:06.789');

date_part_year
---------------
2022
```

Contoh berikut mengekstrak tahun dari kolom CALDATE. Nilai dalam kolom CALDATE adalah tanggal. Untuk informasi selengkapnya tentang data yang digunakan dalam contoh ini, lihatDatabase sampel.

```
select caldate, date_part_year(caldate)
from date
order by
dateid limit 10;

 caldate   | date_part_year
-----------+----------------
2008-01-01 |           2008
2008-01-02 |           2008
2008-01-03 |           2008
```

```
2008-01-04 |           2008
2008-01-05 |           2008
2008-01-06 |           2008
2008-01-07 |           2008
2008-01-08 |           2008
2008-01-09 |           2008
2008-01-10 |           2008
(10 rows)
```

Fungsi DATE_TRUNC

Fungsi DATE_TRUNC memotong ekspresi stempel waktu atau literal berdasarkan bagian tanggal yang Anda tentukan, seperti jam, hari, atau bulan.

Sintaksis

```
DATE_TRUNC('datepart', timestamp)
```

Argumen

datepart

Bagian tanggal untuk memotong nilai stempel waktu. Stempel waktu input dipotong sesuai presisi datepart input. Misalnya, `month` terpotong ke hari pertama bulan itu. Format yang valid adalah sebagai berikut:

- mikrodetik, mikrodetik
- milidetik, milidetik
- kedua, detik
- menit, menit
- jam, jam
- hari, hari
- minggu, minggu
- bulan, bulan
- seperempat, kuartal
- tahun, tahun
- dekade, dekade
- abad, berabad-abad

- milenium, milenium

Untuk informasi selengkapnya tentang singkatan dari beberapa format, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu

stempel waktu

Kolom timestamp atau ekspresi yang secara implisit mengkonversi ke stempel waktu.

Jenis pengembalian

TIMESTAMP

Contoh

Pisahkan stempel waktu masukan ke yang kedua.

```
SELECT DATE_TRUNC('second', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:06
```

Pisahkan stempel waktu masukan ke menit.

```
SELECT DATE_TRUNC('minute', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:00
```

Memangkas stempel waktu input ke jam.

```
SELECT DATE_TRUNC('hour', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:00:00
```

Memangkas stempel waktu input ke hari itu.

```
SELECT DATE_TRUNC('day', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 00:00:00
```

Pisahkan stempel waktu masukan ke hari pertama dalam sebulan.

```
SELECT DATE_TRUNC('month', TIMESTAMP '20200430 04:05:06.789');
```

```
date_trunc
2020-04-01 00:00:00
```

Potong stempel waktu input ke hari pertama seperempat.

```
SELECT DATE_TRUNC('quarter', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

Pisahkan stempel waktu masukan ke hari pertama dalam setahun.

```
SELECT DATE_TRUNC('year', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-01-01 00:00:00
```

Potong stempel waktu input ke hari pertama abad.

```
SELECT DATE_TRUNC('millennium', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2001-01-01 00:00:00
```

Pisahkan stempel waktu input ke hari Senin dalam seminggu.

```
select date_trunc('week', TIMESTAMP '20220430 04:05:06.789');
date_trunc
2022-04-25 00:00:00
```

Dalam contoh berikut, fungsi DATE_TRUNC menggunakan bagian tanggal 'minggu' untuk mengembalikan tanggal untuk hari Senin setiap minggu.

```
select date_trunc('week', saletime), sum(pricepaid) from sales where
saletime like '2008-09%' group by date_trunc('week', saletime) order by 1;

date_trunc  |    sum
------------+-------------
2008-09-01  | 2474899
2008-09-08  | 2412354
2008-09-15  | 2364707
2008-09-22  | 2359351
2008-09-29  |  705249
```

Fungsi EKSTRAK

Fungsi EXTRACT mengembalikan bagian tanggal atau waktu dari nilai TIMESTAMP, TIMESTAMPTZ, TIME, TIMETZ, INTERVAL YEAR TO MONTH, atau INTERVAL DAY TO SECOND. Contohnya termasuk hari, bulan, tahun, jam, menit, detik, milidetik, atau mikrodetik dari stempel waktu.

Sintaksis

```
EXTRACT(datepart FROM source)
```

Argumen

datepart

Subbidang tanggal atau waktu untuk mengekstrak, seperti hari, bulan, tahun, jam, menit, detik, milidetik, atau mikrodetik. Untuk nilai yang mungkin, lihatBagian tanggal untuk fungsi tanggal atau stempel waktu.

sumber

Kolom atau ekspresi yang mengevaluasi tipe data TIMESTAMP, TIMESTAMPTZ, WAKTU, TIMETZ, INTERVAL TAHUN KE BULAN, atau INTERVAL HARI KE KEDUA.

Jenis pengembalian

INTEGER jika nilai sumber mengevaluasi tipe data TIMESTAMP, WAKTU, TIMETZ, INTERVAL TAHUN KE BULAN, atau INTERVAL HARI KE KEDUA.

PRESISI GANDA jika nilai sumber mengevaluasi tipe data TIMESTAMPTZ.

Contoh dengan TIMESTAMP

Contoh berikut menentukan angka minggu untuk penjualan di mana harga yang dibayarkan adalah $10.000 atau lebih. Contoh ini menggunakan data TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

```
select salesid, extract(week from saletime) as weeknum
from sales
where pricepaid > 9999
order by 2;
```

```
salesid | weeknum
--------+---------
 159073 |       6
 160318 |       8
 161723 |      26
```

Contoh berikut mengembalikan nilai menit dari nilai timestamp literal.

```
select extract(minute from timestamp '2009-09-09 12:08:43');

date_part
-----------
8
```

Contoh berikut mengembalikan nilai milidetik dari nilai timestamp literal.

```
select extract(ms from timestamp '2009-09-09 12:08:43.101');

date_part
-----------
101
```

Contoh dengan TIMESTAMPTZ

Contoh berikut mengembalikan nilai tahun dari nilai timestamptz literal.

```
select extract(year from timestamptz '1.12.1997 07:37:16.00 PST');

date_part
-----------
1997
```

Contoh dengan waktu

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
--------------------
20:00:00
```

```
00:00:00.5550
00:58:00
```

Contoh berikut mengekstrak menit dari setiap time_val.

```
select extract(minute from time_val) as minutes from time_test;

minutes
-----------
         0
         0
         58
```

Contoh berikut mengekstrak jam dari setiap time_val.

```
select extract(hour from time_val) as hours from time_test;

hours
-----------
         20
         0
         0
```

Contoh berikut mengekstrak milidetik dari nilai literal.

```
select extract(ms from time '18:25:33.123456');

 date_part
-----------
     123
```

Contoh dengan TIMETZ

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
```

```
00:00:00.5550+00
05:58:00+00
```

Contoh berikut mengekstrak jam dari setiap timetz_val.

```
select extract(hour from timetz_val) as hours from time_test;

hours
-----------
         4
         0
         5
```

Contoh berikut mengekstrak milidetik dari nilai literal. Literal tidak dikonversi ke UTC sebelum ekstraksi diproses.

```
select extract(ms from timetz '18:25:33.123456 EST');

 date_part
-----------
     123
```

Contoh berikut mengembalikan jam offset zona waktu dari UTC dari nilai timetz literal.

```
select extract(timezone_hour from timetz '1.12.1997 07:37:16.00 PDT');

date_part
-----------
-7
```

Contoh dengan INTERVAL TAHUN KE BULAN dan INTERVAL HARI KE KEDUA

Contoh berikut mengekstrak bagian hari 1 dari INTERVAL DAY TO SECOND yang mendefinisikan 36 jam, yaitu 1 hari 12 jam.

```
select EXTRACT('days' from INTERVAL '36 hours' DAY TO SECOND)

 date_part
------------------
 1
```

Contoh berikut mengekstrak bagian bulan 3 dari TAHUN KE BULAN yang mendefinisikan 15 bulan, yaitu 1 tahun 3 bulan.

```
select EXTRACT('month' from INTERVAL '15 months' YEAR TO MONTH)

 date_part
------------------
 3
```

Contoh berikut mengekstrak bagian bulan 6 dari 30 bulan yaitu 2 tahun 6 bulan.

```
select EXTRACT('month' from INTERVAL '30' MONTH)

 date_part
------------------
 6
```

Contoh berikut mengekstrak bagian jam 2 dari 50 jam yaitu 2 hari 2 jam.

```
select EXTRACT('hours' from INTERVAL '50' HOUR)

 date_part
------------------
 2
```

Contoh berikut mengekstrak bagian menit 11 dari 1 jam 11 menit 11.123 detik.

```
select EXTRACT('minute' from INTERVAL '70 minutes 70.123 seconds' MINUTE TO SECOND)

 date_part
------------------
 11
```

Contoh berikut mengekstrak bagian detik 1.11 dari 1 hari 1 jam 1 menit 1,11 detik.

```
select EXTRACT('seconds' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND)

 date_part
------------------
 1.11
```

Contoh berikut mengekstrak jumlah total jam dalam INTERVAL. Setiap bagian diekstraksi dan ditambahkan ke total.

```
select EXTRACT('days' from INTERVAL '50' HOUR) * 24 + EXTRACT('hours' from INTERVAL
 '50' HOUR)

 ?column?
------------------
 50
```

Contoh berikut mengekstrak jumlah detik dalam INTERVAL. Setiap bagian diekstraksi dan ditambahkan ke total.

```
select EXTRACT('days' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 86400 +
       EXTRACT('hours' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 3600 +
       EXTRACT('minutes' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 60 +
       EXTRACT('seconds' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND)

 ?column?
------------------
 90061.11
```

fungsi GETDATE

GETDATE mengembalikan tanggal dan waktu saat ini di zona waktu sesi saat ini (UTC secara default). Ini mengembalikan tanggal mulai atau waktu pernyataan saat ini, bahkan ketika itu berada dalam blok transaksi.

Sintaksis

```
GETDATE()
```

Tanda kurung diperlukan.

Jenis pengembalian

TIMESTAMP

Contoh

Contoh berikut menggunakan fungsi GETDATE untuk mengembalikan stempel waktu penuh untuk tanggal saat ini.

```
select getdate();

timestamp
---------------------
2008-12-04 16:10:43
```

Contoh berikut menggunakan fungsi GETDATE di dalam fungsi TRUNC untuk mengembalikan tanggal saat ini tanpa waktu.

```
select trunc(getdate());

trunc
------------
2008-12-04
```

Fungsi INTERVAL_CMP

INTERVAL_CMP membandingkan dua interval dan kembali 1 jika interval pertama lebih besar, -1 jika interval kedua lebih besar, dan 0 jika intervalnya sama. Untuk informasi selengkapnya, lihat Contoh literal interval tanpa sintaks kualifier.

Sintaks

```
INTERVAL_CMP(interval1, interval2)
```

Argumen

interval1

Nilai literal interval.

interval2

Nilai literal interval.

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan nilai `3 days` to`1 year`.

```
select interval_cmp('3 days','1 year');

interval_cmp
--------------
-1
```

Contoh ini membandingkan nilainya `7 days` dengan. `1 week`

```
select interval_cmp('7 days','1 week');

interval_cmp
--------------
0
```

Contoh berikut membandingkan nilai `1 year` to`3 days`.

```
select interval_cmp('1 year','3 days');

interval_cmp
--------------
1
```

Fungsi LAST_DAY

LAST_DAY mengembalikan tanggal hari terakhir bulan yang berisi tanggal. Tipe pengembalian selalu DATE, terlepas dari tipe data argumen tanggal.

Untuk informasi selengkapnya tentang mengambil bagian tanggal tertentu, lihatFungsi DATE_TRUNC.

Sintaksis

```
LAST_DAY( { date | timestamp } )
```

Argumen

tanggal | stempel waktu

Kolom tipe data `DATE` atau `TIMESTAMP` atau ekspresi yang secara implisit mengevaluasi ke atau tipe. `DATE` `TIMESTAMP`

Jenis pengembalian

DATE

Contoh

Contoh berikut mengembalikan tanggal hari terakhir di bulan berjalan.

```
select last_day(sysdate);

  last_day
------------
 2014-01-31
```

Contoh berikut mengembalikan jumlah tiket yang terjual untuk masing-masing dari 7 hari terakhir dalam sebulan. Nilai dalam kolom SALETIME adalah stempel waktu.

```
select datediff(day, saletime, last_day(saletime)) as "Days Remaining", sum(qtysold)
from sales
where datediff(day, saletime, last_day(saletime)) < 7
group by 1
order by 1;

days remaining |  sum
---------------+-------
             0 | 10140
             1 | 11187
             2 | 11515
             3 | 11217
             4 | 11446
             5 | 11708
             6 | 10988
(7 rows)
```

Fungsi MONTHS_BETWEEN

MONTHS_BETWEEN menentukan jumlah bulan antara dua tanggal.

Jika kencan pertama lebih lambat dari tanggal kedua, hasilnya positif; jika tidak, hasilnya negatif.

Jika salah satu argumen adalah nol, hasilnya adalah NULL.

Sintaksis

```
MONTHS_BETWEEN( date1, date2 )
```

Argumen

tanggal1

Kolom tipe data DATE atau ekspresi yang secara implisit mengevaluasi tipe. DATE

tanggal2

Kolom tipe data DATE atau ekspresi yang secara implisit mengevaluasi tipe. DATE

Jenis pengembalian

FLOAT8

Bagian bilangan bulat dari hasil didasarkan pada perbedaan antara nilai tahun dan bulan dari tanggal. Bagian fraksional dari hasil dihitung dari nilai hari dan stempel waktu tanggal dan mengasumsikan bulan 31 hari.

Jika date1 dan date2 keduanya berisi tanggal yang sama dalam satu bulan (misalnya, 1/15 /14 dan 2/15 /14) atau hari terakhir bulan tersebut (misalnya, 8/31 /14 dan 9/30 /14), maka hasilnya adalah bilangan bulat berdasarkan nilai tahun dan bulan tanggal, terlepas dari apakah bagian stempel waktu cocok, jika ada.

Contoh

Contoh berikut mengembalikan bulan antara 1/18 /1969 dan/1969. 3/18

```
select months_between('1969-01-18', '1969-03-18')
as months;

months
----------
-2
```

Contoh berikut mengembalikan bulan antara 1/18 /1969 dan/1969. 1/18

```
select months_between('1969-01-18', '1969-01-18')
as months;
```

```
months
----------
0
```

Contoh berikut mengembalikan bulan antara pertunjukan pertama dan terakhir dari suatu acara.

```
select eventname,
min(starttime) as first_show,
max(starttime) as last_show,
months_between(max(starttime),min(starttime)) as month_diff
from event
group by eventname
order by eventname
limit 5;

eventname         first_show             last_show              month_diff
-------------------------------------------------------------------------
.38 Special       2008-01-21 19:30:00.0  2008-12-25 15:00:00.0  11.12
3 Doors Down      2008-01-03 15:00:00.0  2008-12-01 19:30:00.0  10.94
70s Soul Jam      2008-01-16 19:30:00.0  2008-12-07 14:00:00.0  10.7
A Bronx Tale      2008-01-21 19:00:00.0  2008-12-15 15:00:00.0  10.8
A Catered Affair  2008-01-08 19:30:00.0  2008-12-19 19:00:00.0  11.35
```

fungsi NEXT_DAY

NEXT_DAY mengembalikan tanggal contoh pertama dari hari yang ditentukan yang lebih lambat dari tanggal yang diberikan.

Jika nilai hari adalah hari yang sama dalam seminggu dengan tanggal yang diberikan, kejadian berikutnya dari hari itu dikembalikan.

Sintaksis

```
NEXT_DAY( { date | timestamp }, day )
```

Argumen

tanggal | stempel waktu

Kolom tipe data DATE atau TIMESTAMP atau ekspresi yang secara implisit mengevaluasi ke atau tipe. DATE TIMESTAMP

hari

Sebuah string yang berisi nama setiap hari. Kapitalisasi tidak masalah.

Nilai yang valid adalah sebagai berikut.

Hari	Nilai
Minggu	Su, Minggu, Minggu
Senin	M, Mo, Sen, Senin
Selasa	Tu, Sel, Selasa
Rabu	W, Kami, Rabu, Rabu
Kamis	Th, Kam, Kamis, Kamis
Jumat	F, Fr, Jum, Jumat
Sabtu	Sa, Sab, Sabtu

Jenis pengembalian

DATE

Contoh

Contoh berikut mengembalikan tanggal Selasa pertama setelah 8/20 /2014.

```
select next_day('2014-08-20','Tuesday');

next_day
-----------
2014-08-26
```

Contoh berikut mengembalikan tanggal Selasa pertama setelah 1/1 /2008 pukul 5:54:44.

```
select listtime, next_day(listtime, 'Tue') from listing limit 1;
```

```
listtime            | next_day
--------------------+----------
2008-01-01 05:54:44 | 2008-01-08
```

Contoh berikut mendapat target tanggal pemasaran untuk kuartal ketiga.

```
select username, (firstname ||' '|| lastname) as name,
eventname, caldate, next_day (caldate, 'Monday') as marketing_target
from sales, date, users, event
where sales.buyerid = users.userid
and sales.eventid = event.eventid
and event.dateid = date.dateid
and date.qtr = 3
order by marketing_target, eventname, name;

username  |     name          |     eventname        |    caldate   |
 marketing_target
----------+-------------------+----------------------+--------------
+------------------
MBO26QSG  |   Callum Atkinson | .38 Special          |  2008-07-06  | 2008-07-07
WCR50YIU  |   Erasmus Alvarez | A Doll's House       |  2008-07-03  | 2008-07-07
CKT70OIE  |   Hadassah Adkins | Ana Gabriel          |  2008-07-06  | 2008-07-07
VVG07OUO  |   Nathan Abbott   | Armando Manzanero    |  2008-07-04  | 2008-07-07
GEW77SII  |   Scarlet Avila   | August: Osage County |  2008-07-06  | 2008-07-07
ECR71CVS  |   Caryn Adkins    | Ben Folds            |  2008-07-03  | 2008-07-07
KUW82CYU  |   Kaden Aguilar   | Bette Midler         |  2008-07-01  | 2008-07-07
WZE78DJZ  |   Kay Avila       | Bette Midler         |  2008-07-01  | 2008-07-07
HXY04NVE  |   Dante Austin    | Britney Spears       |  2008-07-02  | 2008-07-07
URY81YWF  |   Wilma Anthony   | Britney Spears       |  2008-07-02  | 2008-07-07
```

fungsi SYSDATE

SYSDATE mengembalikan tanggal dan waktu saat ini di zona waktu sesi saat ini (UTC secara default).

> **Note**
>
> SYSDATE mengembalikan tanggal dan waktu mulai untuk transaksi saat ini, bukan untuk dimulainya pernyataan saat ini.

Sintaksis

```
SYSDATE
```

Fungsi ini tidak memerlukan argumen.

Jenis pengembalian

TIMESTAMP

Contoh

Contoh berikut menggunakan fungsi SYSDATE untuk mengembalikan stempel waktu penuh untuk tanggal saat ini.

```
select sysdate;

timestamp
----------------------------
2008-12-04 16:10:43.976353
```

Contoh berikut menggunakan fungsi SYSDATE di dalam fungsi TRUNC untuk mengembalikan tanggal saat ini tanpa waktu.

```
select trunc(sysdate);

trunc
------------
2008-12-04
```

Kueri berikut mengembalikan informasi penjualan untuk tanggal yang berada di antara tanggal ketika kueri dikeluarkan dan tanggal berapa pun 120 hari sebelumnya.

```
select salesid, pricepaid, trunc(saletime) as saletime, trunc(sysdate) as now
from sales
where saletime between trunc(sysdate)-120 and trunc(sysdate)
order by saletime asc;

 salesid | pricepaid |  saletime  |    now
---------+-----------+------------+------------
```

```
91535    |    670.00 | 2008-08-07 | 2008-12-05
91635    |    365.00 | 2008-08-07 | 2008-12-05
91901    |   1002.00 | 2008-08-07 | 2008-12-05
...
```

Fungsi TIMEOFDAY

TIMEOFDAY adalah alias khusus yang digunakan untuk mengembalikan hari kerja, tanggal, dan waktu sebagai nilai string. Ia mengembalikan string time of day untuk pernyataan saat ini, bahkan ketika itu berada dalam blok transaksi.

Sintaksis

```
TIMEOFDAY()
```

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut mengembalikan tanggal dan waktu saat ini dengan menggunakan fungsi TIMEOFDAY.

```
select timeofday();

timeofday
------------
Thu Sep 19 22:53:50.333525 2013 UTC
```

Fungsi TIMESTAMP_CMP

Membandingkan nilai dua stempel waktu dan mengembalikan integer. Jika stempel waktu identik, fungsi kembali. `0` Jika stempel waktu pertama lebih besar, fungsi kembali. `1` Jika stempel waktu kedua lebih besar, fungsi kembali. `-1`

Sintaksis

```
TIMESTAMP_CMP(timestamp1, timestamp2)
```

Argumen

stempel waktu1

Kolom tipe data TIMESTAMP atau ekspresi yang secara implisit mengevaluasi tipe. TIMESTAMP

stempel waktu2

Kolom tipe data TIMESTAMP atau ekspresi yang secara implisit mengevaluasi tipe. TIMESTAMP

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan stempel waktu dan menunjukkan hasil perbandingan.

```
SELECT TIMESTAMP_CMP('2008-01-24 06:43:29', '2008-01-24 06:43:29'),
 TIMESTAMP_CMP('2008-01-24 06:43:29', '2008-02-18 02:36:48'), TIMESTAMP_CMP('2008-02-18
 02:36:48', '2008-01-24 06:43:29');

timestamp_cmp  | timestamp_cmp | timestamp_cmp
---------------+---------------+---------------
             0 |            -1 |             1
```

Contoh berikut membandingkan LISTTIME dan SALETIME untuk daftar. Nilai untuk TIMESTAMP_CMP adalah -1 untuk semua listing karena stempel waktu untuk penjualan adalah setelah stempel waktu untuk daftar.

```
select listing.listid, listing.listtime,
sales.saletime, timestamp_cmp(listing.listtime, sales.saletime)
from listing, sales
where listing.listid=sales.listid
order by 1, 2, 3, 4
limit 10;

 listid |      listtime       |      saletime       | timestamp_cmp
--------+---------------------+---------------------+---------------
      1 | 2008-01-24 06:43:29 | 2008-02-18 02:36:48 |            -1
      4 | 2008-05-24 01:18:37 | 2008-06-06 05:00:16 |            -1
      5 | 2008-05-17 02:29:11 | 2008-06-06 08:26:17 |            -1
```

```
      5 | 2008-05-17 02:29:11 | 2008-06-09 08:38:52 |           -1
      6 | 2008-08-15 02:08:13 | 2008-08-31 09:17:02 |           -1
     10 | 2008-06-17 09:44:54 | 2008-06-26 12:56:06 |           -1
     10 | 2008-06-17 09:44:54 | 2008-07-10 02:12:36 |           -1
     10 | 2008-06-17 09:44:54 | 2008-07-16 11:59:24 |           -1
     10 | 2008-06-17 09:44:54 | 2008-07-22 02:23:17 |           -1
     12 | 2008-07-25 01:45:49 | 2008-08-04 03:06:36 |           -1
(10 rows)
```

Contoh ini menunjukkan bahwa TIMESTAMP_CMP mengembalikan 0 untuk stempel waktu yang identik:

```
select listid, timestamp_cmp(listtime, listtime)
from listing
order by 1 , 2
limit 10;

 listid | timestamp_cmp
--------+---------------
      1 |             0
      2 |             0
      3 |             0
      4 |             0
      5 |             0
      6 |             0
      7 |             0
      8 |             0
      9 |             0
     10 |             0
(10 rows)
```

Fungsi TIMESTAMP_CMP_DATE

TIMESTAMP_CMP_DATE membandingkan nilai stempel waktu dan tanggal. Jika nilai stempel waktu dan tanggal identik, fungsi kembali. `0` Jika stempel waktu lebih besar secara kronologis, fungsi kembali. 1 Jika tanggal lebih besar, fungsi kembali`-1`.

Sintaksis

```
TIMESTAMP_CMP_DATE(timestamp, date)
```

Argumen

stempel waktu

Kolom tipe data TIMESTAMP atau ekspresi yang secara implisit mengevaluasi tipe. TIMESTAMP

tanggal

Kolom tipe data DATE atau ekspresi yang secara implisit mengevaluasi tipe. DATE

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan LISTTIME dengan tanggal. 2008-06-18 Daftar yang dibuat setelah tanggal ini kembali1; daftar yang dibuat sebelum tanggal ini kembali-1. Nilai LISTTIME adalah stempel waktu.

```
select listid, listtime,
timestamp_cmp_date(listtime, '2008-06-18')
from listing
order by 1, 2, 3
limit 10;


 listid |      listtime       | timestamp_cmp_date
--------+---------------------+--------------------
      1 | 2008-01-24 06:43:29 |                 -1
      2 | 2008-03-05 12:25:29 |                 -1
      3 | 2008-11-01 07:35:33 |                  1
      4 | 2008-05-24 01:18:37 |                 -1
      5 | 2008-05-17 02:29:11 |                 -1
      6 | 2008-08-15 02:08:13 |                  1
      7 | 2008-11-15 09:38:15 |                  1
      8 | 2008-11-09 05:07:30 |                  1
      9 | 2008-09-09 08:03:36 |                  1
     10 | 2008-06-17 09:44:54 |                 -1
(10 rows)
```

Fungsi TIMESTAMP_CMP_TIMESTAMPTZ

TIMESTAMP_CMP_TIMESTAMPTZ membandingkan nilai ekspresi stempel waktu dengan stempel waktu dengan ekspresi zona waktu. Jika stempel waktu dan stempel waktu dengan nilai zona waktu identik, fungsi kembali. `0` Jika stempel waktu lebih besar secara kronologis, fungsi kembali. `1` Jika stempel waktu dengan zona waktu lebih besar, fungsi kembali. `-1`

Sintaksis

```
TIMESTAMP_CMP_TIMESTAMPTZ(timestamp, timestamptz)
```

Argumen

stempel waktu

Kolom tipe data `TIMESTAMP` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMP`

timestamptz

Kolom tipe data `TIMESTAMPTZ` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMPTZ`

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan stempel waktu dengan stempel waktu dengan zona waktu dan menunjukkan hasil perbandingan.

```
SELECT TIMESTAMP_CMP_TIMESTAMPTZ('2008-01-24 06:43:29', '2008-01-24 06:43:29+00'),
 TIMESTAMP_CMP_TIMESTAMPTZ('2008-01-24 06:43:29', '2008-02-18 02:36:48+00'),
 TIMESTAMP_CMP_TIMESTAMPTZ('2008-02-18 02:36:48', '2008-01-24 06:43:29+00');

timestamp_cmp_timestamptz  | timestamp_cmp_timestamptz | timestamp_cmp_timestamptz
---------------------------+---------------------------+--------------------------
             0             |            -1             |             1
```

Fungsi TIMESTAMPTZ_CMP

TIMESTAMPTZ_CMP membandingkan nilai dua timestamp dengan nilai zona waktu dan mengembalikan bilangan bulat. Jika stempel waktu identik, fungsi kembali. `0` Jika stempel waktu pertama lebih besar secara kronologis, fungsi akan kembali. 1 Jika stempel waktu kedua lebih besar, fungsi kembali. `-1`

Sintaksis

```
TIMESTAMPTZ_CMP(timestamptz1, timestamptz2)
```

Argumen

stempel waktuz1

Kolom tipe data `TIMESTAMPTZ` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMPTZ`

stempel waktu

Kolom tipe data `TIMESTAMPTZ` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMPTZ`

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan stempel waktu dengan zona waktu dan menunjukkan hasil perbandingan.

```
SELECT TIMESTAMPTZ_CMP('2008-01-24 06:43:29+00', '2008-01-24 06:43:29+00'),
 TIMESTAMPTZ_CMP('2008-01-24 06:43:29+00', '2008-02-18 02:36:48+00'),
 TIMESTAMPTZ_CMP('2008-02-18 02:36:48+00', '2008-01-24 06:43:29+00');

timestamptz_cmp  | timestamptz_cmp | timestamptz_cmp
-----------------+-----------------+----------------
        0        |       -1        |       1
```

Fungsi TIMESTAMPTZ_CMP_DATE

TIMESTAMPTZ_CMP_DATE membandingkan nilai stempel waktu dan tanggal. Jika nilai stempel waktu dan tanggal identik, fungsi kembali. `0` Jika stempel waktu lebih besar secara kronologis, fungsi kembali. 1 Jika tanggal lebih besar, fungsi kembali`-1`.

Sintaksis

```
TIMESTAMPTZ_CMP_DATE(timestamptz, date)
```

Argumen

timestamptz

Kolom tipe data `TIMESTAMPTZ` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMPTZ`

tanggal

Kolom tipe data `DATE` atau ekspresi yang secara implisit mengevaluasi tipe. `DATE`

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan LISTTIME sebagai stempel waktu dengan zona waktu dengan tanggal. `2008-06-18` Daftar yang dibuat setelah tanggal ini kembali`1`; daftar yang dibuat sebelum tanggal ini kembali`-1`.

```
select listid, CAST(listtime as timestamptz) as tstz,
timestamp_cmp_date(tstz, '2008-06-18')
from listing
order by 1, 2, 3
limit 10;


 listid |          tstz          | timestamptz_cmp_date
--------+------------------------+----------------------
      1 | 2008-01-24 06:43:29+00 |                   -1
```

```
      2 | 2008-03-05 12:25:29+00 |              -1
      3 | 2008-11-01 07:35:33+00 |               1
      4 | 2008-05-24 01:18:37+00 |              -1
      5 | 2008-05-17 02:29:11+00 |              -1
      6 | 2008-08-15 02:08:13+00 |               1
      7 | 2008-11-15 09:38:15+00 |               1
      8 | 2008-11-09 05:07:30+00 |               1
      9 | 2008-09-09 08:03:36+00 |               1
     10 | 2008-06-17 09:44:54+00 |              -1
(10 rows)
```

Fungsi TIMESTAMPTZ_CMP_TIMESTAMP

TIMESTAMPTZ_CMP_TIMESTAMP membandingkan nilai stempel waktu dengan ekspresi zona waktu dengan ekspresi stempel waktu. Jika stempel waktu dengan zona waktu dan nilai timestamp identik, fungsi kembali. `0` Jika stempel waktu dengan zona waktu lebih besar secara kronologis, fungsi kembali. 1 Jika stempel waktu lebih besar, fungsi kembali. `-1`

Sintaksis

```
TIMESTAMPTZ_CMP_TIMESTAMP(timestamptz, timestamp)
```

Argumen

timestamptz

Kolom tipe data `TIMESTAMPTZ` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMPTZ`

stempel waktu

Kolom tipe data `TIMESTAMP` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMP`

Jenis pengembalian

INTEGER

Contoh

Contoh berikut membandingkan stempel waktu dengan zona waktu dengan stempel waktu dan menunjukkan hasil perbandingan.

```
SELECT TIMESTAMPTZ_CMP_TIMESTAMP('2008-01-24 06:43:29+00', '2008-01-24 06:43:29'),
 TIMESTAMPTZ_CMP_TIMESTAMP('2008-01-24 06:43:29+00', '2008-02-18 02:36:48'),
 TIMESTAMPTZ_CMP_TIMESTAMP('2008-02-18 02:36:48+00', '2008-01-24 06:43:29');

timestamptz_cmp_timestamp  | timestamptz_cmp_timestamp | timestamptz_cmp_timestamp
---------------------------+---------------------------+--------------------------
             0             |            -1             |             1
```

Fungsi TIMEZONE

TIMEZONE mengembalikan timestamp untuk zona waktu tertentu dan nilai timestamp.

Untuk informasi dan contoh tentang cara mengatur zona waktu, lihattimezone.

Untuk informasi dan contoh tentang cara mengonversi zona waktu, lihatCONVERT_TIMEZONE.

Sintaksis

```
TIMEZONE('timezone', { timestamp | timestamptz })
```

Argumen

zona waktu

Zona waktu untuk nilai kembali. Zona waktu dapat ditentukan sebagai nama zona waktu (seperti **'Africa/Kampala'** atau**'Singapore'**) atau sebagai singkatan zona waktu (seperti **'UTC'** atau**'PDT'**). Untuk melihat daftar nama zona waktu yang didukung, jalankan perintah berikut.

```
select pg_timezone_names();
```

Untuk melihat daftar singkatan zona waktu yang didukung, jalankan perintah berikut.

```
select pg_timezone_abbrevs();
```

Perhatikan bahwa Amazon Redshift menggunakan Database Zona Waktu IANA sebagai sumber otoritatif spesifikasi zona waktu. Untuk informasi selengkapnya dan contoh tambahan, lihat Catatan penggunaan zona waktu.

stempel waktu | timestamptz

Ekspresi yang menghasilkan tipe TIMESTAMP atau TIMESTAMPTZ, atau nilai yang secara implisit dapat dipaksa ke stempel waktu atau stempel waktu dengan zona waktu.

Jenis pengembalian

TIMESTAMPTZ bila digunakan dengan ekspresi TIMESTAMP.

TIMESTAMP bila digunakan dengan ekspresi TIMESTAMPTZ.

Contoh

Berikut ini mengembalikan timestamp untuk zona waktu UTC menggunakan timestamp `2008-06-17 09:44:54` dari zona waktu PST.

```
SELECT TIMEZONE('PST', '2008-06-17 09:44:54');

timezone
-----------------------
2008-06-17 17:44:54+00
```

Berikut ini mengembalikan timestamp untuk zona waktu PST menggunakan timestamp dengan zona waktu UTC. `2008-06-17 09:44:54+00`

```
SELECT TIMEZONE('PST', timestamptz('2008-06-17 09:44:54+00'));

timezone
-----------------------
2008-06-17 01:44:54
```

Fungsi TO_TIMESTAMP

TO_TIMESTAMP mengonversi string TIMESTAMP ke TIMESTAMPTZ. Untuk daftar fungsi tanggal dan waktu tambahan untuk Amazon Redshift, lihat. Fungsi tanggal dan waktu

Sintaksis

```
to_timestamp(timestamp, format)
```

```
to_timestamp (timestamp, format, is_strict)
```

Argumen

stempel waktu

Sebuah string yang mewakili nilai timestamp dalam format yang ditentukan oleh format. Jika argumen ini dibiarkan kosong, nilai stempel waktu defaultnya. `0001-01-01 00:00:00`

format

Sebuah string literal yang mendefinisikan format nilai timestamp. Format yang menyertakan zona waktu (**`TZ`**,**`tz`**, atau**`OF`**) tidak didukung sebagai input. Untuk format stempel waktu yang valid, lihat. String format tanggal-waktu

is_strict

Nilai Boolean opsional yang menentukan apakah kesalahan dikembalikan jika nilai timestamp masukan berada di luar jangkauan. Ketika is_strict disetel ke TRUE, kesalahan dikembalikan jika ada nilai di luar rentang. Ketika is_strict disetel ke FALSE, yang merupakan default, maka nilai overflow diterima.

Jenis pengembalian

TIMESTAMPTZ

Contoh

Contoh berikut menunjukkan penggunaan fungsi TO_TIMESTAMP untuk mengonversi string TIMESTAMP ke TIMESTAMPTZ.

```
select sysdate, to_timestamp(sysdate, 'YYYY-MM-DD HH24:MI:SS') as second;

timestamp                  | second
---------------------------   ---------------------
2021-04-05 19:27:53.281812 | 2021-04-05 19:27:53+00
```

Dimungkinkan untuk melewatkan TO_TIMESTAMP bagian dari tanggal. Bagian tanggal yang tersisa diatur ke nilai default. Waktu termasuk dalam output:

```
SELECT TO_TIMESTAMP('2017','YYYY');

to_timestamp
---------------------------
```

```
2017-01-01 00:00:00+00
```

Pernyataan SQL berikut mengonversi string '2011-12-18 24:38:15 'menjadi TIMESTAMPTZ. Hasilnya adalah TIMESTAMPTZ yang jatuh pada hari berikutnya karena jumlah jam lebih dari 24 jam:

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS');

to_timestamp
---------------------
2011-12-19 00:38:15+00
```

Pernyataan SQL berikut mengonversi string '2011-12-18 24:38:15 'menjadi TIMESTAMPTZ. Hasilnya adalah kesalahan karena nilai waktu dalam stempel waktu lebih dari 24 jam:

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS', TRUE);

ERROR:  date/time field time value out of range: 24:38:15.0
```

Fungsi TRUNC

Mempotong a `TIMESTAMP` dan mengembalikan a. `DATE`

Fungsi ini juga dapat memotong angka. Untuk informasi selengkapnya, lihat Fungsi TRUNC.

Sintaks

```
TRUNC(timestamp)
```

Argumen

stempel waktu

Kolom tipe data `TIMESTAMP` atau ekspresi yang secara implisit mengevaluasi tipe. `TIMESTAMP`

Untuk mengembalikan nilai stempel waktu dengan `00:00:00` waktu, lemparkan hasil fungsi ke a. `TIMESTAMP`

Jenis pengembalian

DATE

Contoh

Contoh berikut mengembalikan bagian tanggal dari hasil fungsi SYSDATE (yang mengembalikan stempel waktu).

```
SELECT SYSDATE;

+----------------------------+
|         timestamp          |
+----------------------------+
| 2011-07-21 10:32:38.248109 |
+----------------------------+

SELECT TRUNC(SYSDATE);

+------------+
|   trunc    |
+------------+
| 2011-07-21 |
+------------+
```

Contoh berikut menerapkan fungsi TRUNC ke kolom. `TIMESTAMP` Jenis pengembalian adalah tanggal.

```
SELECT TRUNC(starttime) FROM event
ORDER BY eventid LIMIT 1;

+------------+
|   trunc    |
+------------+
| 2008-01-25 |
+------------+
```

Contoh berikut mengembalikan nilai timestamp dengan `00:00:00` sebagai waktu dengan mentransmisikan hasil fungsi TRUNC ke a. `TIMESTAMP`

```
SELECT CAST((TRUNC(SYSDATE)) AS TIMESTAMP);

+---------------------+
|        trunc        |
+---------------------+
```

```
| 2011-07-21 00:00:00 |
+---------------------+
```

Bagian tanggal untuk fungsi tanggal atau stempel waktu

Tabel berikut mengidentifikasi nama bagian tanggal dan waktu bagian dan singkatan yang diterima sebagai argumen untuk fungsi berikut:

- DATEADD
- DATEDIFF
- DATE_PART
- EKSTRAK

Tanggal paruh waktu atau paruh waktu	Singkatan
milenium, milenium	mil, mil
abad, berabad-abad	c, sen, sen
dekade, dekade	Desember, decs
jangka waktu	epoch (didukung oleh) EKSTRAK
tahun, tahun	y, thn, thn
seperempat, kuartal	qtr, qtrs
bulan, bulan	mon, mons
minggu, minggu	w
hari dalam seminggu	dayofweek, dow, dw, hari kerja (didukung oleh dan) DATE_PART Fungsi EKSTRAK Mengembalikan integer dari 0-6, dimulai dengan hari Minggu.

Tanggal paruh waktu atau paruh waktu	Singkatan
	Note Bagian tanggal DOW berperilaku berbeda dari bagian tanggal hari minggu (D) yang digunakan untuk string format datetime. D didasarkan pada bilangan bulat 1-7, di mana hari Minggu adalah 1. Untuk informasi selengkap nya, lihat String format tanggal-waktu.
hari dalam setahun	dayofyear, doy, dy, yearday (didukung oleh) EKSTRAK
hari, hari	d
jam, jam	h, jam, jam
menit, menit	m, min, menit
kedua, detik	s, detik, detik
milidetik, milidetik	ms, msec, msec, mdetik, mdetik, milidetik, milidetik, milidetik, milidetik
mikrodetik, mikrodetik	mikrosec, mikrodetik, mikrodetik, usecond, useconds, us, usec, usec
zona waktu, zona waktu_jam , zona waktu_menit	Didukung oleh EKSTRAK untuk timestamp dengan zona waktu (TIMESTAMPTZ) saja.

Variasi hasil dengan detik, milidetik, dan mikrodetik

Perbedaan kecil dalam hasil kueri terjadi ketika fungsi tanggal yang berbeda menentukan detik, milidetik, atau mikrodetik sebagai bagian tanggal:

- Fungsi EXTRACT mengembalikan bilangan bulat untuk bagian tanggal yang ditentukan saja, mengabaikan bagian tanggal tingkat yang lebih tinggi dan lebih rendah. Jika bagian tanggal yang ditentukan adalah detik, milidetik dan mikrodetik tidak termasuk dalam hasil. Jika bagian tanggal

yang ditentukan adalah milidetik, detik dan mikrodetik tidak termasuk. Jika bagian tanggal yang ditentukan adalah mikrodetik, detik dan milidetik tidak termasuk.

- Fungsi DATE_PART mengembalikan bagian detik lengkap dari stempel waktu, terlepas dari bagian tanggal yang ditentukan, mengembalikan nilai desimal atau bilangan bulat sesuai kebutuhan.

Misalnya, bandingkan hasil kueri berikut:

```
create table seconds(micro timestamp);

insert into seconds values('2009-09-21 11:10:03.189717');

select extract(sec from micro) from seconds;

date_part
-----------
3

select date_part(sec, micro) from seconds;

pgdate_part
-------------
3.189717
```

Catatan CENTURY, EPOCH, DECADE, dan MIL

CENTURY atau CENTURY

Amazon Redshift menafsirkan CENTURY untuk memulai dengan tahun ## #1 dan diakhiri dengan tahun: ###0

```
select extract (century from timestamp '2000-12-16 12:21:13');
date_part
-----------
20

select extract (century from timestamp '2001-12-16 12:21:13');
date_part
-----------
21
```

EPOCH

Implementasi Amazon Redshift EPOCH relatif terhadap 1970-01-01 00:00:00.000 000 terlepas dari zona waktu tempat cluster berada. Anda mungkin perlu mengimbangi hasil dengan perbedaan jam tergantung pada zona waktu di mana cluster berada.

Contoh berikut menunjukkan hal berikut:

1. Membuat tabel bernama EVENT_EXAMPLE berdasarkan tabel EVENT. Perintah CREATE AS ini menggunakan fungsi DATE_PART untuk membuat kolom tanggal (disebut PGDATE_PART secara default) untuk menyimpan nilai epoch untuk setiap peristiwa.
2. Memilih kolom dan tipe data EVENT_EXAMPLE dari PG_TABLE_DEF.
3. Memilih EVENTNAME, STARTTIME, dan PGDATE_PART dari tabel EVENT_EXAMPLE untuk melihat format tanggal dan waktu yang berbeda.
4. Memilih EVENTNAME dan STARTTIME dari EVENT EXAMPLE apa adanya. Mengonversi nilai epoch di PGDATE_PART menggunakan interval 1 detik ke stempel waktu tanpa zona waktu, dan mengembalikan hasilnya dalam kolom bernama CONVERTED_TIMESTAMP.

```
create table event_example
as select eventname, starttime, date_part(epoch, starttime) from event;

select "column", type from pg_table_def where tablename='event_example';

     column    |            type
---------------+-----------------------------
 eventname     | character varying(200)
 starttime     | timestamp without time zone
 pgdate_part   | double precision
(3 rows)
```

```
select eventname, starttime, pgdate_part from event_example;

   eventname          |      starttime      | pgdate_part
----------------------+---------------------+-------------
 Mamma Mia!           | 2008-01-01 20:00:00 |  1199217600
 Spring Awakening     | 2008-01-01 15:00:00 |  1199199600
 Nas                  | 2008-01-01 14:30:00 |  1199197800
 Hannah Montana       | 2008-01-01 19:30:00 |  1199215800
 K.D. Lang            | 2008-01-01 15:00:00 |  1199199600
 Spamalot             | 2008-01-02 20:00:00 |  1199304000
 Macbeth              | 2008-01-02 15:00:00 |  1199286000
```

```
 The Cherry Orchard   | 2008-01-02 14:30:00 |  1199284200
 Macbeth              | 2008-01-02 19:30:00 |  1199302200
 Demi Lovato          | 2008-01-02 19:30:00 |  1199302200


select eventname,
starttime,
timestamp with time zone 'epoch' + pgdate_part * interval '1 second' AS
 converted_timestamp
from event_example;

       eventname      |      starttime      | converted_timestamp
----------------------+---------------------+---------------------
 Mamma Mia!           | 2008-01-01 20:00:00 | 2008-01-01 20:00:00
 Spring Awakening     | 2008-01-01 15:00:00 | 2008-01-01 15:00:00
 Nas                  | 2008-01-01 14:30:00 | 2008-01-01 14:30:00
 Hannah Montana       | 2008-01-01 19:30:00 | 2008-01-01 19:30:00
 K.D. Lang            | 2008-01-01 15:00:00 | 2008-01-01 15:00:00
 Spamalot             | 2008-01-02 20:00:00 | 2008-01-02 20:00:00
 Macbeth              | 2008-01-02 15:00:00 | 2008-01-02 15:00:00
 The Cherry Orchard   | 2008-01-02 14:30:00 | 2008-01-02 14:30:00
 Macbeth              | 2008-01-02 19:30:00 | 2008-01-02 19:30:00
 Demi Lovato          | 2008-01-02 19:30:00 | 2008-01-02 19:30:00
 ...
```

DEKADE atau DEKADE

Amazon Redshift menafsirkan DECADE atau DECADECADES DATEPART berdasarkan kalender umum. Misalnya, karena kalender umum dimulai dari tahun 1, dekade pertama (dekade 1) adalah 0001-01-01 hingga 0009-12-31, dan dekade kedua (dekade 2) adalah 0010-01-01 hingga 0019-12-31. Misalnya, dekade 201 membentang dari 2000-01-01 hingga 2009-12-31:

```
select extract(decade from timestamp '1999-02-16 20:38:40');
date_part
-----------
200

select extract(decade from timestamp '2000-02-16 20:38:40');
date_part
-----------
201

select extract(decade from timestamp '2010-02-16 20:38:40');
```

```
date_part
-----------
202
```

MIL atau MILS

Amazon Redshift menafsirkan MIL untuk memulai dengan hari pertama tahun #001 dan diakhiri dengan hari terakhir tahun ini: #000

```
select extract (mil from timestamp '2000-12-16 12:21:13');
date_part
-----------
2

select extract (mil from timestamp '2001-12-16 12:21:13');
date_part
-----------
3
```

Fungsi hash

Topik

- Fungsi CHECKSUM
- Fungsi FarmFingerprint64
- Fungsi FUNC_SHA1
- Fungsi FNV_HASH
- Fungsi MD5
- Fungsi SHA
- Fungsi SHA1
- Fungsi SHA2
- MURMUR3_32_HASH

Fungsi hash adalah fungsi matematika yang mengubah nilai input numerik menjadi nilai lain.

Fungsi CHECKSUM

Menghitung nilai checksum untuk membangun indeks hash.

Sintaksis

```
CHECKSUM(expression)
```

Pendapat

ekspresi

Ekspresi input harus berupa tipe data VARCHAR, INTEGER, atau DECIMAL.

Jenis pengembalian

Fungsi CHECKSUM mengembalikan bilangan bulat.

Contoh

Contoh berikut menghitung nilai checksum untuk kolom COMMISSION:

```
select checksum(commission)
from sales
order by salesid
limit 10;

checksum
----------
10920
1140
5250
2625
2310
5910
11820
2955
8865
975
(10 rows)
```

Fungsi FarmFingerprint64

Menghitung nilai farmhash dari argumen input menggunakan fungsi. `Fingerprint64`

Sintaksis

```
farmFingerprint64(expression)
```

Pendapat

ekspresi

Ekspresi input harus tipe VARBYTE data VARCHAR atau.

Jenis pengembalian

`farmFingerprint64`Fungsi mengembalikan a`BIGINT`.

Contoh

Contoh berikut mengembalikan `farmFingerprint64` nilai `Amazon Redshift` input sebagai tipe VARCHAR data.

```
SELECT farmFingerprint64('Amazon Redshift');
```

```
  farmfingerprint64
---------------------
 8085098817162212970
```

Contoh berikut mengembalikan `farmFingerprint64` nilai `Amazon Redshift` input sebagai tipe VARBYTE data.

```
SELECT farmFingerprint64('Amazon Redshift'::varbyte);
```

```
  farmfingerprint64
---------------------
 8085098817162212970
```

Fungsi FUNC_SHA1

Sinonim dari fungsi SHA1.

Lihat Fungsi SHA1.

Fungsi FNV_HASH

Menghitung fungsi hash FNV-1a non-kriptografi 64-bit untuk semua tipe data dasar.

Sintaksis

```
FNV_HASH(value [, seed])
```

Argumen

nilai

Nilai input yang akan di-hash. Amazon Redshift menggunakan representasi biner dari nilai untuk hash nilai input; misalnya, nilai INTEGER di-hash menggunakan 4 byte dan nilai BIGINT di-hash menggunakan 8 byte. Juga, hashing input CHAR dan VARCHAR tidak mengabaikan spasi akhir.

benih

Benih BIGINT dari fungsi hash adalah opsional. Jika tidak diberikan, Amazon Redshift menggunakan benih FNV default. Ini memungkinkan menggabungkan hash dari beberapa kolom tanpa konversi atau penggabungan apa pun.

Jenis pengembalian

BIGINT

Contoh

Contoh berikut mengembalikan hash FNV dari sebuah angka, string 'Amazon Redshift', dan penggabungan keduanya.

```
select fnv_hash(1);
        fnv_hash
----------------------
 -5968735742475085980
(1 row)
```

```
select fnv_hash('Amazon Redshift');
      fnv_hash
```

```
---------------------
 7783490368944507294
(1 row)
```

```
select fnv_hash('Amazon Redshift', fnv_hash(1));
       fnv_hash
----------------------
 -2202602717770968555
(1 row)
```

Catatan penggunaan

- Untuk menghitung hash tabel dengan beberapa kolom, Anda dapat menghitung hash FNV dari kolom pertama dan meneruskannya sebagai benih ke hash kolom kedua. Kemudian, ia meneruskan hash FNV dari kolom kedua sebagai benih ke hash kolom ketiga.

  Contoh berikut membuat benih untuk hash tabel dengan beberapa kolom.

  ```
  select fnv_hash(column_3, fnv_hash(column_2, fnv_hash(column_1))) from sample_table;
  ```

- Properti yang sama dapat digunakan untuk menghitung hash dari gabungan string.

  ```
  select fnv_hash('abcd');
         fnv_hash
  ---------------------
   -281581062704388899
  (1 row)
  ```

  ```
  select fnv_hash('cd', fnv_hash('ab'));
        fnv_hash
  ---------------------
   -281581062704388899
  (1 row)
  ```

- Fungsi hash menggunakan jenis input untuk menentukan jumlah byte untuk hash. Gunakan casting untuk menerapkan jenis tertentu, jika perlu.

  Contoh berikut menggunakan berbagai jenis input untuk menghasilkan hasil yang berbeda.

  ```
  select fnv_hash(1::smallint);
        fnv_hash
  ```

```
--------------------
 589727492704079044
(1 row)
```

```
select fnv_hash(1);
       fnv_hash
----------------------
 -5968735742475085980
(1 row)
```

```
select fnv_hash(1::bigint);
       fnv_hash
----------------------
 -8517097267634966620
(1 row)
```

Fungsi MD5

Menggunakan fungsi hash kriptografi MD5 untuk mengubah string panjang variabel menjadi string 32 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 128-bit.

Sintaksis

```
MD5(string)
```

Argumen

senar

String dengan panjang variabel.

Jenis pengembalian

Fungsi MD5 mengembalikan string 32 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 128-bit.

Contoh

Contoh berikut menunjukkan nilai 128-bit untuk string 'Amazon Redshift':

```
select md5('Amazon Redshift');
md5
----------------------------------
f7415e33f972c03abd4f3fed36748f7a
(1 row)
```

Fungsi SHA

Sinonim dari fungsi SHA1.

Lihat Fungsi SHA1.

Fungsi SHA1

Fungsi SHA1 menggunakan fungsi hash kriptografi SHA1 untuk mengubah string panjang variabel menjadi string 40 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 160-bit.

Sintaksis

SHA1 adalah sinonim dari Fungsi SHA dan. Fungsi FUNC_SHA1

```
SHA1(string)
```

Argumen

senar

String dengan panjang variabel.

Jenis pengembalian

Fungsi SHA1 mengembalikan string 40 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 160-bit.

Contoh

Contoh berikut mengembalikan nilai 160-bit untuk kata 'Amazon Redshift':

```
select sha1('Amazon Redshift');
```

Fungsi SHA2

Fungsi SHA2 menggunakan fungsi hash kriptografi SHA2 untuk mengubah string panjang variabel menjadi string karakter. String karakter adalah representasi teks dari nilai heksadesimal checksum dengan jumlah bit yang ditentukan.

Sintaksis

```
SHA2(string, bits)
```

Argumen

senar

String dengan panjang variabel.

bilangan bulat

Jumlah bit dalam fungsi hash. Nilai yang valid adalah 0 (sama dengan 256), 224, 256, 384, dan 512.

Jenis pengembalian

Fungsi SHA2 mengembalikan string karakter yang merupakan representasi teks dari nilai heksadesimal checksum atau string kosong jika jumlah bit tidak valid.

Contoh

Contoh berikut mengembalikan nilai 256-bit untuk kata 'Amazon Redshift':

```
select sha2('Amazon Redshift', 256);
```

MURMUR3_32_HASH

Fungsi MURMUR3_32_HASH menghitung hash non-kriptografi Murmur3A 32-bit untuk semua tipe data umum termasuk tipe numerik dan string.

Sintaksis

```
MURMUR3_32_HASH(value [, seed])
```

Argumen

nilai

Nilai input untuk hash. Amazon Redshift melakukan hash representasi biner dari nilai input. Perilaku ini mirip denganFungsi FNV_HASH, tetapi nilainya dikonversi ke representasi biner yang ditentukan oleh spesifikasi hash Murmur3 Apache Iceberg 32-bit.

benih

Benih INT dari fungsi hash. Argumen ini opsional. Jika tidak diberikan, Amazon Redshift menggunakan benih default 0. Ini memungkinkan menggabungkan hash dari beberapa kolom tanpa konversi atau penggabungan apa pun.

Jenis pengembalian

Fungsi mengembalikan sebuah INT.

Contoh

Contoh berikut mengembalikan hash Murmur3 dari sebuah angka, string 'Amazon Redshift', dan penggabungan keduanya.

```
select MURMUR3_32_HASH(1);

    MURMUR3_32_HASH
----------------------
 1392991556
(1 row)
```

```
select MURMUR3_32_HASH('Amazon Redshift');

    MURMUR3_32_HASH
----------------------
 -1563580564
(1 row)
```

```
select MURMUR3_32_HASH('Amazon Redshift', MURMUR3_32_HASH(1));

    MURMUR3_32_HASH
----------------------
```

```
 -1346554171
(1 row)
```

Catatan penggunaan

Untuk menghitung hash tabel dengan beberapa kolom, Anda dapat menghitung hash Murmur3 dari kolom pertama dan meneruskannya sebagai benih ke hash kolom kedua. Kemudian, ia meneruskan hash Murmur3 dari kolom kedua sebagai benih ke hash kolom ketiga.

Contoh berikut membuat benih untuk hash tabel dengan beberapa kolom.

```
select MURMUR3_32_HASH(column_3, MURMUR3_32_HASH(column_2, MURMUR3_32_HASH(column_1)))
 from sample_table;
```

Properti yang sama dapat digunakan untuk menghitung hash dari gabungan string.

```
select MURMUR3_32_HASH('abcd');

   MURMUR3_32_HASH
---------------------
 1139631978
(1 row)
```

```
select MURMUR3_32_HASH('cd', MURMUR3_32_HASH('ab'));

   MURMUR3_32_HASH
---------------------
 1711522338
(1 row)
```

Fungsi hash menggunakan jenis input untuk menentukan jumlah byte untuk hash. Gunakan casting untuk menerapkan jenis tertentu, jika perlu.

Contoh berikut menggunakan jenis input yang berbeda untuk menghasilkan hasil yang berbeda.

```
select MURMUR3_32_HASH(1, MURMUR3_32_HASH(1));

   MURMUR3_32_HASH
--------------------
 -1193428387
(1 row)
```

```
select MURMUR3_32_HASH(1);

   MURMUR3_32_HASH
----------------------
 1392991556
(1 row)
```

```
select MURMUR3_32_HASH(1, MURMUR3_32_HASH(2));

   MURMUR3_32_HASH
----------------------
 1179621905
(1 row)
```

HyperLogLog fungsi

Berikut ini, Anda dapat menemukan deskripsi untuk HyperLogLog fungsi SQL yang didukung Amazon Redshift.

Topik

- Fungsi HLL
- Fungsi HLL_CREATE_SKETCH
- Fungsi HLL_CARDINALITY
- Fungsi HLL_COMBINE
- Fungsi HLL_COMBINE_SKETCHES

Fungsi HLL

Fungsi HLL mengembalikan HyperLogLog kardinalitas nilai ekspresi masukan. Fungsi HLL bekerja dengan tipe data apa pun kecuali tipe data HLLSKETCH. Fungsi HLL mengabaikan nilai NULL. Ketika tidak ada baris dalam tabel atau semua baris adalah NULL, kardinalitas yang dihasilkan adalah 0.

Sintaksis

```
HLL (aggregate_expression)
```

Pendapat

ekspresi agregat

Setiap ekspresi valid yang memberikan nilai ke agregat, seperti nama kolom. Fungsi ini mendukung setiap tipe data sebagai input kecuali HLLSKETCH, GEOMETRY, GEOGRAPHY, dan VARBYTE.

Jenis pengembalian

Fungsi HLL mengembalikan nilai BIGINT atau INT8.

Contoh

Contoh berikut mengembalikan kardinalitas kolom `an_int` dalam tabel`a_table`.

```
CREATE TABLE a_table(an_int INT);
INSERT INTO a_table VALUES (1), (2), (3), (4);

SELECT hll(an_int) AS cardinality FROM a_table;
cardinality
-------------
4
```

Fungsi HLL_CREATE_SKETCH

Fungsi HLL_CREATE_SKETCH mengembalikan tipe data HLLSKETCH yang merangkum nilai ekspresi masukan. Fungsi HLL_CREATE_SKETCH bekerja dengan tipe data apa pun dan mengabaikan nilai NULL. Ketika tidak ada baris dalam tabel atau semua baris adalah NULL, sketsa yang dihasilkan tidak memiliki pasangan nilai indeks seperti`{"version":1,"logm":15,"sparse":{"indices":[],"values":[]}}`.

Sintaksis

```
HLL_CREATE_SKETCH (aggregate_expression)
```

Pendapat

ekspresi agregat

Setiap ekspresi valid yang memberikan nilai ke agregat, seperti nama kolom. Nilai NULL diabaikan. Fungsi ini mendukung setiap tipe data sebagai input kecuali HLLSKETCH, GEOMETRY, GEOGRAPHY, dan VARBYTE.

Jenis pengembalian

Fungsi HLL_CREATE_SKETCH mengembalikan nilai HLLSKETCH.

Contoh

Contoh berikut mengembalikan tipe HLLSKETCH untuk kolom `an_int` dalam tabel. `a_table` Objek JSON digunakan untuk mewakili HyperLogLog sketsa jarang saat mengimpor, mengekspor, atau mencetak sketsa. Representasi string (dalam format Base64) digunakan untuk mewakili HyperLogLog sketsa padat.

```
CREATE TABLE a_table(an_int INT);
INSERT INTO a_table VALUES (1), (2), (3), (4);

SELECT hll_create_sketch(an_int) AS sketch FROM a_table;
sketch
-----------------------------------------------------------------------------------------------
{"version":1,"logm":15,"sparse":{"indices":
[20812342,20850007,22362299,47158030],"values":[1,2,1,1]}}
(1 row)
```

Fungsi HLL_CARDINALITY

Fungsi HLL_CARDINALITY mengembalikan kardinalitas dari tipe data input HLLSKETCH.

Sintaksis

```
HLL_CARDINALITY (hllsketch_expression)
```

Pendapat

hllsketch_ekspresi

Setiap ekspresi valid yang mengevaluasi ke tipe HLLSKETCH, seperti nama kolom. Nilai input adalah tipe data HLLSKETCH.

Jenis pengembalian

Fungsi HLL_CARDINALITY mengembalikan nilai BIGINT atau INT8.

Contoh

Contoh berikut mengembalikan kardinalitas kolom `sketch` dalam tabel`hll_table`.

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
 (5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

SELECT hll_cardinality(sketch) AS cardinality FROM hll_table;
cardinality
-------------
6
4
(2 rows)
```

Fungsi HLL_COMBINE

Fungsi agregat HLL_COMBINE mengembalikan tipe data HLLSKETCH yang menggabungkan semua masukan nilai HLLSKETCH.

Kombinasi dua atau lebih HyperLogLog sketsa adalah HLLSKETCH baru yang merangkum informasi tentang penyatuan nilai-nilai berbeda yang diwakili oleh setiap sketsa input. Setelah menggabungkan sketsa, Amazon Redshift mengekstrak kardinalitas penyatuan dua atau lebih kumpulan data. Untuk informasi selengkapnya tentang cara menggabungkan beberapa sketsa, lihatContoh: Mengembalikan HyperLogLog sketsa dari menggabungkan beberapa sketsa.

Sintaksis

```
HLL_COMBINE (hllsketch_expression)
```

Pendapat

hllsketch_ekspresi

Setiap ekspresi valid yang mengevaluasi ke tipe HLLSKETCH, seperti nama kolom. Nilai input adalah tipe data HLLSKETCH.

Jenis pengembalian

Fungsi HLL_COMBINE mengembalikan tipe HLLSKETCH.

Contoh

Contoh berikut mengembalikan nilai gabungan HLLSKETCH dalam tabel. `hll_table`

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
 (5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

SELECT hll_combine(sketch) AS sketches FROM hll_table;
sketches
--------------------------------------------------------------------------------------------------
{"version":1,"logm":15,"sparse":{"indices":
[20812342,20850007,22362299,40314817,42650774,47158030],"values":[1,2,1,3,2,1]}}
(1 row)
```

Fungsi HLL_COMBINE_SKETCHES

HLL_COMBINE_SKETCHES adalah fungsi skalar yang mengambil sebagai input dua nilai HLLSKETCH dan menggabungkannya menjadi satu HLLSKETCH.

Kombinasi dua atau lebih HyperLogLog sketsa adalah HLLSKETCH baru yang merangkum informasi tentang penyatuan nilai-nilai berbeda yang diwakili oleh setiap sketsa input.

Sintaksis

```
HLL_COMBINE_SKETCHES (hllsketch_expression1, hllsketch_expression2)
```

Pendapat

hllsketch_expression1 dan hllsketch_expression2

Setiap ekspresi valid yang mengevaluasi ke tipe HLLSKETCH, seperti nama kolom.

Jenis pengembalian

Fungsi HLL_COMBINE_SKETCHES mengembalikan tipe HLLSKETCH.

Contoh

Contoh berikut mengembalikan nilai gabungan HLLSKETCH dalam tabel. `hll_table`

```
WITH tbl1(x, y)
     AS (SELECT Hll_create_sketch(1),
                Hll_create_sketch(2)
         UNION ALL
         SELECT Hll_create_sketch(3),
                Hll_create_sketch(4)
         UNION ALL
         SELECT Hll_create_sketch(5),
                Hll_create_sketch(6)
         UNION ALL
         SELECT Hll_create_sketch(7),
                Hll_create_sketch(8)),
     tbl2(x, y)
     AS (SELECT Hll_create_sketch(9),
                Hll_create_sketch(10)
         UNION ALL
         SELECT Hll_create_sketch(11),
                Hll_create_sketch(12)
         UNION ALL
         SELECT Hll_create_sketch(13),
                Hll_create_sketch(14)
         UNION ALL
         SELECT Hll_create_sketch(15),
                Hll_create_sketch(16)
```

```
        UNION ALL
        SELECT Hll_create_sketch(NULL),
               Hll_create_sketch(NULL)),
     tbl3(x, y)
     AS (SELECT *
        FROM   tbl1
        UNION ALL
        SELECT *
        FROM   tbl2)
SELECT Hll_combine_sketches(x, y)
FROM   tbl3;
```

Fungsi JSON

Topik

- Fungsi JSON_PARSE
- Fungsi CAN_JSON_PARSE
- Fungsi JSON_SERIALIZE
- Fungsi JSON_SERIALIZE_TO_VARBYTE
- Text-based Fungsi JSON

> **Note**
>
> Kami menyarankan Anda menggunakan fungsi-fungsi berikut untuk bekerja dengan JSON:
>
> - Fungsi JSON_PARSE
> - Fungsi CAN_JSON_PARSE
> - Fungsi JSON_SERIALIZE
> - Fungsi JSON_SERIALIZE_TO_VARBYTE
>
> Dengan JSON_PARSE, Anda hanya perlu mengonversi teks JSON ke nilai tipe SUPER sekali saat konsumsi, setelah itu Anda dapat mengoperasikan nilai SUPER. Amazon Redshift mengurai nilai SUPER lebih efisien daripada VARCHAR, yang merupakan output untuk fungsi JSON berbasis teks. Untuk informasi lebih lanjut tentang bekerja dengan tipe data SUPER, bukaSemi-structured data dalam Amazon Redshift.

Ketika Anda perlu menyimpan satu set pasangan kunci-nilai yang relatif kecil, Anda dapat menghemat ruang dengan menyimpan data dalam format JSON. Karena string JSON dapat disimpan dalam satu kolom, menggunakan JSON mungkin lebih efisien daripada menyimpan data Anda dalam format tabular. Misalnya, misalkan Anda memiliki tabel sparse, di mana Anda perlu memiliki banyak kolom untuk sepenuhnya mewakili semua atribut yang mungkin, tetapi sebagian besar nilai kolom adalah NULL untuk baris tertentu atau kolom tertentu. Dengan menggunakan JSON untuk penyimpanan, Anda mungkin dapat menyimpan data untuk satu baris dalam pasangan key:value dalam satu string JSON dan menghilangkan kolom tabel yang jarang terisi.

Selain itu, Anda dapat dengan mudah memodifikasi string JSON untuk menyimpan pasangan kunci:nilai tambahan saat skema JSON Anda berubah tanpa perlu menambahkan kolom ke tabel.

Sebaiknya gunakan JSON dengan hemat. JSON bukanlah pilihan yang baik untuk menyimpan kumpulan data yang lebih besar karena, dengan menyimpan data yang berbeda dalam satu kolom, JSON tidak menggunakan arsitektur penyimpanan kolom Amazon Redshift. Meskipun Amazon Redshift mendukung fungsi JSON di atas kolom CHAR dan VARCHAR, sebaiknya gunakan SUPER untuk memproses data dalam format serialisasi JSON. SUPER menggunakan representasi tanpa skema pasca-parse yang dapat secara efisien menanyakan data hierarkis. Untuk informasi selengkapnya tentang tipe data SUPER, lihatSemi-structured data dalam Amazon Redshift.

JSON menggunakan string teks yang UTF-8 dikodekan, sehingga string JSON dapat disimpan sebagai tipe data CHAR atau VARCHAR.

String JSON harus diformat JSON dengan benar, sesuai dengan aturan berikut:

- JSON tingkat root dapat berupa objek JSON atau array JSON. Objek JSON adalah kumpulan pasangan key:value yang dipisahkan koma yang dikelilingi oleh kurung kurung keriting.

  Sebagai contoh, `{"one":1, "two":2}` .
- Array JSON adalah kumpulan nilai yang dipisahkan koma yang dikelilingi oleh tanda kurung.

  Contohnya adalah sebagai berikut: `["first", {"one":1}, "second", 3, null]`
- Array JSON menggunakan indeks berbasis nol; elemen pertama dalam array berada di posisi 0. Dalam pasangan kunci: nilai JSON, kuncinya adalah string dalam tanda kutip ganda.
- Nilai JSON dapat berupa salah satu dari berikut ini:
  - Objek JSON
  - array
  - string

    - Diwakili menggunakan tanda kutip ganda
  - number
    - Termasuk bilangan bulat, desimal, dan float
  - boolean
  - null
- Objek kosong dan array kosong adalah nilai JSON yang valid.
- Bidang JSON peka huruf besar/kecil.
- Ruang putih antara elemen struktural JSON (seperti`{ }, [ ]`) diabaikan.

Fungsi Amazon Redshift JSON dan perintah Amazon Redshift COPY menggunakan metode yang sama untuk bekerja dengan JSON-formatted data. Untuk informasi selengkapnya tentang bekerja dengan JSON, lihat SALIN dari format JSON

Fungsi JSON_PARSE

Fungsi JSON_PARSE mengurai data dalam format JSON dan mengubahnya menjadi representasi. `SUPER`

Untuk memasukkan ke dalam tipe `SUPER` data menggunakan perintah INSERT atau UPDATE, gunakan fungsi JSON_PARSE. Saat Anda menggunakan JSON_PARSE () untuk mengurai string JSON menjadi `SUPER` nilai, batasan tertentu berlaku. Untuk informasi tambahan, lihat Opsi penguraian untuk SUPER.

Sintaksis

```
JSON_PARSE( {json_string | binary_value} )
```

Argumen

json_string

Ekspresi yang mengembalikan JSON berseri sebagai `VARCHAR` tipe `VARBYTE` atau.

nilai_biner

Nilai biner tipe VARBYTE.

Jenis pengembalian

SUPER

Contoh

Untuk mengubah array JSON `[10001,10002,"abc"]` menjadi tipe SUPER data, gunakan contoh berikut.

```
SELECT JSON_PARSE('[10001,10002,"abc"]');

+---------------------+
|     json_parse      |
+---------------------+
| [10001,10002,"abc"] |
+---------------------+
```

Untuk memastikan bahwa fungsi mengubah array JSON menjadi tipe SUPER data, gunakan contoh berikut. Untuk informasi selengkapnya, lihat Fungsi JSON_TYPEOF

```
SELECT JSON_TYPEOF(JSON_PARSE('[10001,10002,"abc"]'));

+-------------+
| json_typeof |
+-------------+
| array       |
+-------------+
```

Fungsi CAN_JSON_PARSE

Fungsi CAN_JSON_PARSE mengurai data dalam format JSON dan mengembalikan `true` jika hasilnya dapat dikonversi ke nilai menggunakan fungsi JSON_PARSE. SUPER

Sintaksis

```
CAN_JSON_PARSE( {json_string | binary_value} )
```

Argumen

json_string

Ekspresi yang mengembalikan JSON serial dalam VARCHAR bentuk.

nilai_biner

Nilai biner tipe VARBYTE.

Jenis pengembalian

BOOLEAN

Catatan penggunaan

- CAN_JSON_PARSE mengembalikan false untuk string kosong. Ini mengembalikan NULL ketika argumen input adalah null.

Contoh

Contoh berikut menunjukkan CAN_JSON_PARSE berjalan pada array JSON yang dibentuk dengan benar menggunakan kondisi CASE. Ini mengembalikan true, jadi Amazon Redshift menjalankan fungsi JSON_PARSE pada nilai contoh.

```
SELECT CASE
            WHEN CAN_JSON_PARSE('[10001,10002,"abc"]')
            THEN JSON_PARSE('[10001,10002,"abc"]')
        END;

 case
---------------------
'[10001,10002,"abc"]'
```

Contoh berikut menunjukkan CAN_JSON_PARSE berjalan pada nilai yang bukan format JSON menggunakan kondisi CASE. Ini mengembalikan false, jadi Amazon Redshift mengembalikan segmen dalam klausa ELSE dari kondisi CASE sebagai gantinya.

```
SELECT CASE
            WHEN CAN_JSON_PARSE('This is a string.')
            THEN JSON_PARSE('This is a string.')
            ELSE 'This is not JSON.'
        END;

 case
---------------------
"This is not JSON."
```

Fungsi JSON_SERIALIZE

Fungsi JSON_SERIALIZE membuat serial SUPER ekspresi menjadi representasi JSON tekstual untuk mengikuti RFC 8259. Untuk informasi selengkapnya tentang RFC tersebut, lihat Format Per JavaScript tukaran Data Notasi Objek (JSON).

Batas SUPER ukuran kira-kira sama dengan batas blok, dan bat VARCHAR asnya lebih kecil dari batas SUPER ukuran. Oleh karena itu, fungsi JSON_SERIALIZE mengembalikan kesalahan ketika format JSON melebihi batas VARCHAR sistem. Untuk memeriksa ukuran SUPER ekspresi, lihat fung JSON_UKURAN sinya.

Sintaksis

```
JSON_SERIALIZE(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

VARCHAR

> **Note**
>
> Nilai VARCHAR yang dikembalikan selalu berupa string JSON non-null. Jika super_expression adalah NULL, JSON_SERIALIZE mengembalikan string JSON. `'null'`

Contoh

Untuk membuat serial SUPER nilai ke string, gunakan contoh berikut.

```
SELECT JSON_SERIALIZE(JSON_PARSE('[10001,10002,"abc"]'));

+---------------------+
|   json_serialize    |
+---------------------+
| [10001,10002,"abc"] |
```

```
+---------------------+
```

Fungsi JSON_SERIALIZE_TO_VARBYTE

Fungsi JSON_SERIALIZE_TO_VARBYTE mengonversi SUPER nilai ke string JSON yang mirip dengan JSON_SERIALIZE (), tetapi disimpan dalam nilai sebagai gantinya. VARBYTE

Sintaksis

```
JSON_SERIALIZE_TO_VARBYTE(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

VARBYTE

Contoh

Untuk membuat serial SUPER nilai dan mengembalikan hasilnya dalam VARBYTE format, gunakan contoh berikut.

```
SELECT JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'));

+----------------------------------------+
|       json_serialize_to_varbyte        |
+----------------------------------------+
| 5b31303030312c31303030322c22616263225d |
+----------------------------------------+
```

Untuk membuat serial SUPER nilai dan memberikan hasilnya ke VARCHAR format, gunakan contoh berikut. Untuk informasi selengkapnya, lihat Fungsi CAST.

```
SELECT CAST((JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'))) AS VARCHAR);

+---------------------------+
| json_serialize_to_varbyte |
+---------------------------+
```

```
| [10001,10002,"abc"]       |
+---------------------------+
```

Text-based Fungsi JSON

Fungsi di bagian ini mengurai nilai JSON sebagai VARCHAR. Untuk mengurai JSON, sebaiknya gunakan fungsi berikut, yang mengurai nilai JSON sebagai SUPER. Amazon Redshift mengurai nilai SUPER lebih efisien daripada VARCHAR.

- Fungsi JSON_PARSE
- Fungsi CAN_JSON_PARSE
- Fungsi JSON_SERIALIZE
- Fungsi JSON_SERIALIZE_TO_VARBYTE

Topik

- Fungsi IS_VALID_JSON
- Fungsi IS_VALID_JSON_ARRAY
- Fungsi JSON_ARRAY_LENGTH
- Fungsi JSON_EXTRACT_ARRAY_ELEMENT_TEXT
- Fungsi JSON_EXTRACT_PATH_TEXT

Fungsi IS_VALID_JSON

> **Note**
>
> CAN_JSON_PARSE dan fungsi terkaitnya mengurai nilai JSON sebagai SUPER, yang diurai oleh Amazon Redshift lebih efisien daripada VARCHAR.
> Alih-alih menggunakan IS_VALID_JSON, kami sarankan Anda memvalidasi string JSON Anda menggunakan. Fungsi CAN_JSON_PARSE

Fungsi IS_VALID_JSON memvalidasi string JSON. Fungsi mengembalikan Boolean `true` jika string dibentuk JSON dengan benar atau `false` jika string cacat. Untuk memvalidasi array JSON, gunakan Fungsi IS_VALID_JSON_ARRAY

Untuk informasi selengkapnya, lihat Fungsi JSON.

Sintaks

```
IS_VALID_JSON('json_string')
```

Argumen

json_string

Sebuah string atau ekspresi yang mengevaluasi ke string JSON.

Jenis pengembalian

BOOLEAN

Contoh

Untuk membuat tabel dan menyisipkan string JSON untuk pengujian, gunakan contoh berikut.

```
CREATE TABLE test_json(id int IDENTITY(0,1), json_strings VARCHAR);

-- Insert valid JSON strings --
INSERT INTO test_json(json_strings) VALUES
('{"a":2}'),
('{"a":{"b":{"c":1}}}'),
('{"a": [1,2,"b"]}');

-- Insert invalid JSON strings --
INSERT INTO test_json(json_strings) VALUES
('{{}}'),
('{1:"a"}'),
('[1,2,3]');
```

Untuk memvalidasi string dalam contoh sebelumnya, gunakan contoh berikut.

```
SELECT id, json_strings, IS_VALID_JSON(json_strings)
FROM test_json
ORDER BY id;

+----+--------------------+---------------+
| id |    json_strings    | is_valid_json |
+----+--------------------+---------------+
|  0 | {"a":2}            | true          |
```

```
|  4 | {"a":{"b":{"c":1}}} | true          |
|  8 | {"a": [1,2,"b"]}    | true          |
| 12 | {{}}                | false         |
| 16 | {1:"a"}             | false         |
| 20 | [1,2,3]             | false         |
+----+---------------------+---------------+
```

Fungsi IS_VALID_JSON_ARRAY

> **Note**
>
> JSON_PARSE dan fungsi terkaitnya mengurai nilai JSON sebagai SUPER, yang diurai oleh Amazon Redshift lebih efisien daripada VARCHAR.
> Alih-alih menggunakan IS_VALID_JSON_ARRAY, kami sarankan Anda mengurai string JSON menggunakan untuk mendapatkan nilai SUPER. Fungsi JSON_PARSE Kemudian, gunakan Fungsi IS_ARRAY fungsi untuk mengonfirmasi bahwa array terbentuk dengan benar.

Fungsi IS_VALID_JSON_ARRAY memvalidasi array JSON. Fungsi mengembalikan Boolean `true` jika array dibentuk JSON dengan benar atau `false` jika array salah format. Untuk memvalidasi string JSON, gunakan Fungsi IS_VALID_JSON

Untuk informasi selengkapnya, lihat Fungsi JSON.

Sintaks

```
IS_VALID_JSON_ARRAY('json_array')
```

Argumen

json_array

Sebuah string atau ekspresi yang mengevaluasi ke array JSON.

Jenis pengembalian

BOOLEAN

Contoh

Untuk membuat tabel dan menyisipkan string JSON untuk pengujian, gunakan contoh berikut.

```
CREATE TABLE test_json_arrays(id int IDENTITY(0,1), json_arrays VARCHAR);

-- Insert valid JSON array strings --
INSERT INTO test_json_arrays(json_arrays)
VALUES('[]'),
('["a","b"]'),
('["a",["b",1,["c",2,3,null]]]');

-- Insert invalid JSON array strings --
INSERT INTO test_json_arrays(json_arrays)
VALUES('{"a":1}'),
('a'),
('[1,2,]');
```

Untuk memvalidasi string dalam contoh sebelumnya, gunakan contoh berikut.

```
SELECT json_arrays, IS_VALID_JSON_ARRAY(json_arrays)
FROM test_json_arrays ORDER BY id;

+-----------------------------+---------------------+
|         json_arrays         | is_valid_json_array |
+-----------------------------+---------------------+
| []                          | true                |
| ["a","b"]                   | true                |
| ["a",["b",1,["c",2,3,null]]] | true                |
| {"a":1}                     | false               |
| a                           | false               |
| [1,2,]                      | false               |
+-----------------------------+---------------------+
```

Fungsi JSON_ARRAY_LENGTH

> **Note**
>
> JSON_PARSE dan fungsi terkaitnya mengurai nilai JSON sebagai SUPER, yang diurai oleh Amazon Redshift lebih efisien daripada VARCHAR.

Alih-alih menggunakan JSON_ARRAY_LENGTH, kami sarankan Anda mengurai string JSON menggunakan untuk mendapatkan nilai SUPER. Fungsi JSON_PARSE Kemudian, gunakan Fungsi GET_ARRAY_LENGTH untuk mendapatkan panjang array Anda.

Fungsi JSON_ARRAY_LENGTH mengembalikan jumlah elemen dalam array luar string JSON. Jika argumen null_if_invalid disetel ke `true` dan string JSON tidak valid, fungsi kembali alih- `NULL` alih mengembalikan kesalahan.

Untuk informasi selengkapnya, lihat Fungsi JSON.

Sintaks

```
JSON_ARRAY_LENGTH('json_array' [, null_if_invalid ] )
```

Argumen

json_array

Array JSON yang diformat dengan benar.

null_if_invalid

(Opsional) `BOOLEAN` Nilai yang menentukan apakah akan dikembalikan `NULL` jika string JSON input tidak valid alih-alih mengembalikan kesalahan. Untuk mengembalikan `NULL` jika JSON tidak valid, tentukan `true` (`t`). Untuk mengembalikan kesalahan jika JSON tidak valid, tentukan `false` (`f`). Nilai default-nya `false`.

Jenis pengembalian

`INTEGER`

Contoh

Untuk mengembalikan jumlah elemen dalam array, gunakan contoh berikut.

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14]');

+------------------+
| json_array_length |
```

```
+-------------------+
|                 5 |
+-------------------+
```

Untuk mengembalikan kesalahan karena JSON tidak valid, gunakan contoh berikut.

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14');

ERROR: invalid json array object [11,12,13,{"f1":21,"f2":[25,26]},14
```

Untuk menyet el null_if_invalid ke true, sehingga pernyataan dikembalikan alih- NULL alih mengembalikan kesalahan untuk JSON yang tidak valid, gunakan contoh berikut.

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14',true);

+-------------------+
| json_array_length |
+-------------------+
| NULL              |
+-------------------+
```

Fungsi JSON_EXTRACT_ARRAY_ELEMENT_TEXT

> **Note**
>
> JSON_PARSE dan fungsi terkaitnya mengurai nilai JSON sebagai SUPER, yang diurai oleh Amazon Redshift lebih efisien daripada VARCHAR.
> Alih-alih menggunakan JSON_EXTRACT_ARRAY_ELEMENT_TEXT, kami sarankan Anda mengurai string JSON menggunakan untuk mendapatkan nilai SUPER. Fungsi JSON_PARSE Kemudian, kueri elemen yang Anda inginkan menggunakan indeks array, menggunakan `value[element position]` sintaks. Untuk informasi lebih lanjut tentang kueri elemen array dalam nilai SUPER, buka. Mengkueri data semi-terstruktur

Fungsi JSON_EXTRACT_ARRAY_ELEMENT_TEXT mengembalikan elemen array JSON di larik terluar string JSON, menggunakan indeks berbasis nol. Elemen pertama dalam array berada di posisi 0. Jika indeks negatif atau di luar batas, JSON_EXTRACT_ARRAY_ELEMENT_TEXT kembali. NULL Jika argumen null_if_invalid disetel ke TRUE dan string JSON tidak valid, fungsi kembali alih- NULL alih mengembalikan kesalahan.

Untuk informasi selengkapnya, lihat Fungsi JSON.

Sintaks

```
JSON_EXTRACT_ARRAY_ELEMENT_TEXT('json string', pos [, null_if_invalid ] )
```

Argumen

json_string

String JSON yang diformat dengan benar.

pos

Sebuah INTEGER mewakili indeks elemen array yang akan dikembalikan, menggunakan indeks array berbasis nol.

null_if_invalid

(Opsional) BOOLEAN Nilai yang menentukan apakah akan dikembalikan NULL jika string JSON input tidak valid alih-alih mengembalikan kesalahan. Untuk mengembalikan NULL jika JSON tidak valid, tentukan true (t). Untuk mengembalikan kesalahan jika JSON tidak valid, tentukan false (f). Nilai default-nya false.

Jenis pengembalian

VARCHAR

Sebuah VARCHAR string yang mewakili elemen array JSON yang direferensikan oleh pos.

Contoh

Untuk mengembalikan elemen array pada posisi 2, yang merupakan elemen ketiga dari indeks array berbasis nol, gunakan contoh berikut.

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('[111,112,113]', 2);

+---------------------------------+
| json_extract_array_element_text |
+---------------------------------+
```

```
|                              113 |
+----------------------------------+
```

Untuk mengembalikan kesalahan karena JSON tidak valid, gunakan contoh berikut.

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('["a",["b",1,["c",2,3,null,]]]',1);

ERROR: invalid json array object ["a",["b",1,["c",2,3,null,]]]
```

Untuk menyet el null_if_invalid ke true, sehingga pernyataan kembali alih- NULL alih mengembalikan kesalahan untuk JSON yang tidak valid, gunakan contoh berikut.

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('["a",["b",1,["c",2,3,null,]]]',1,true);

+----------------------------------+
| json_extract_array_element_text |
+----------------------------------+
| NULL                             |
+----------------------------------+
```

Pertimbangkan contoh pernyataan berikut. Jika string JSON yang disediakan atau indeksnya adalah NULL, JSON_EXTRACT_ARRAY_ELEMENT_TEXT mengembalikan NULL terlepas dari nilai parameter lainnya.

```
--Statement where json_string is NULL.
SELECT json_extract_array_element_text(NULL, 0)

 json_extract_array_element_text
---------------------------------
                            NULL

--Statement where pos is NULL and json_string is invalid JSON.
SELECT json_extract_array_element_text('invalid_json', NULL);

 json_extract_array_element_text
---------------------------------
                            NULL

--Statement where json_string is NULL and null_if_invalid is FALSE.
SELECT json_extract_array_element_text(NULL, 0, FALSE);
```

```
 json_extract_array_element_text
--------------------------------
                            NULL
```

Pertimbangkan contoh pernyataan berikut. Ketika null_if_invalid adalah TRUE, JSON_EXTRACT_ARRA Y_ELEMENT_TEXT mengembalikan NULL ketika json_string adalah JSON tidak valid. Jika null_if_invalid adalah FALSE atau tidak disetel, fungsi mengembalikan kesalahan ketika json_string tidak valid.

```
--Statement with invalid JSON where null_if_invalid is TRUE.
SELECT json_extract_array_element_text('invalid_json', 0, TRUE);

 json_extract_array_element_text
--------------------------------
                            NULL

--Statement with invalid JSON where null_if_invalid is FALSE.
SELECT json_extract_array_element_text('invalid_json', 0);

ERROR:  JSON parsing error
```

Pertimbangkan contoh berikut, di mana json_string adalah JSON yang valid, dan pos mengacu pada nilai JSON. `null` Dalam hal ini, JSON_EXTRACT_ARRAY_ELEMENT_TEXT mengembalikan NULL, terlepas dari nilai null_if_invalid.

```
--Statement selecting a null value.
SELECT json_extract_array_element_text('[null]', 0);

  json_extract_array_element_text
---------------------------------
                             NULL
```

Fungsi JSON_EXTRACT_PATH_TEXT

> **Note**
>
> JSON_PARSE dan fungsi terkaitnya mengurai nilai JSON sebagai SUPER, yang diurai oleh Amazon Redshift lebih efisien daripada VARCHAR.
> Alih-alih menggunakan JSON_EXTRACT_PATH_TEXT, kami sarankan Anda mengurai string JSON menggunakan untuk mendapatkan nilai SUPER. Fungsi JSON_PARSE Kemudian,

kueri elemen yang Anda inginkan menggunakan `value.attribute` sintaks. Untuk informasi lebih lanjut tentang kueri elemen array dalam nilai SUPER, buka. Mengkueri data semi-terstruktur

Fungsi JSON_EXTRACT_PATH_TEXT mengembalikan nilai untuk pasangan kunci-nilai yang direferensikan oleh serangkaian elemen jalur dalam string JSON. Jalur JSON dapat bersarang hingga kedalaman lima tingkat. Elemen jalur peka huruf besar/kecil. Jika elemen jalur tidak ada dalam string JSON, JSON_EXTRACT_PATH_TEXT kembali. NULL

Jika argumen null_if_invalid disetel ke TRUE dan string JSON tidak valid, fungsi kembali alih- NULL alih mengembalikan kesalahan.

JSON_EXTRACT_PATH_TEXT memiliki ukuran data maksimum 16MB. Jadi, jika ada catatan JSON yang lebih besar dari 16MB, memprosesnya dengan JSON_EXTRACT_PATH_TEXT menghasilkan kesalahan.

Untuk informasi tentang fungsi JSON tambahan, lihatFungsi JSON. Untuk informasi selengkapnya tentang bekerja dengan JSON, lihatSALIN dari format JSON.

Sintaksis

```
JSON_EXTRACT_PATH_TEXT('json_string', 'path_elem' [,'path_elem'[, …] ]
 [, null_if_invalid ] )
```

Argumen

json_string

String JSON yang diformat dengan benar.

path_element

Elemen jalur dalam string JSON. Diperlukan satu elemen jalur. Elemen jalur tambahan dapat ditentukan, hingga lima tingkat kedalaman.

null_if_invalid

(Opsional) BOOLEAN Nilai yang menentukan apakah akan dikembalikan NULL jika string JSON input tidak valid alih-alih mengembalikan kesalahan. Untuk mengembalikan NULL jika JSON tidak

valid, tentukan TRUE (t). Untuk mengembalikan kesalahan jika JSON tidak valid, tentukan FALSE (f). Nilai default-nya FALSE.

Dalam string JSON, Amazon Redshift mengenali \n sebagai karakter baris baru dan \t sebagai karakter tab. Untuk memuat garis miring terbalik, lepaskan dengan backslash (). \\ Untuk informasi selengkapnya, lihat Karakter melarikan diri di JSON.

Jenis pengembalian

VARCHAR

Sebuah VARCHAR string yang mewakili nilai JSON yang direferensikan oleh elemen jalur.

Contoh

Untuk mengembalikan nilai jalur'f4', 'f6', gunakan contoh berikut.

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}','f4', 'f6');

+------------------------+
| json_extract_path_text |
+------------------------+
| star                   |
+------------------------+
```

Untuk mengembalikan kesalahan karena JSON tidak valid, gunakan contoh berikut.

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4', 'f6');

ERROR: invalid json object {"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}
```

Untuk menyet el null_if_invalid ke TR UE, sehingga pernyataan kembali NULL untuk JSON yang tidak valid alih-alih mengembalikan kesalahan, gunakan contoh berikut.

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4',
 'f6',true);

+------------------------+
| json_extract_path_text |
+------------------------+
```

```
| NULL                    |
+-------------------------+
```

Pertimbangkan contoh berikut, yang memilih nilai untuk jalur`'farm', 'barn', 'color'`, di mana nilai yang diambil berada di tingkat ketiga, gunakan contoh berikut. Sampel ini diformat dengan alat lint JSON, untuk membuatnya lebih mudah dibaca.

```
SELECT JSON_EXTRACT_PATH_TEXT('{
    "farm": {
        "barn": {
            "color": "red",
            "feed stocked": true
        }
    }
}', 'farm', 'barn', 'color');
+-------------------------+
| json_extract_path_text |
+-------------------------+
| red                     |
+-------------------------+
```

Untuk kembali NULL karena `'color'` elemen hilang, gunakan contoh berikut. Sampel ini diformat dengan alat lint JSON.

```
SELECT JSON_EXTRACT_PATH_TEXT('{
    "farm": {
        "barn": {}
    }
}', 'farm', 'barn', 'color');

+-------------------------+
| json_extract_path_text |
+-------------------------+
| NULL                    |
+-------------------------+
```

Jika JSON valid, mencoba mengekstrak elemen yang hilang kembaliNULL.

Untuk mengembalikan nilai jalur`'house', 'appliances', 'washing machine', 'brand'`, gunakan contoh berikut.

```
SELECT JSON_EXTRACT_PATH_TEXT('{
```

```
  "house": {
    "address": {
      "street": "123 Any St.",
      "city": "Any Town",
      "state": "FL",
      "zip": "32830"
    },
    "bathroom": {
      "color": "green",
      "shower": true
    },
    "appliances": {
      "washing machine": {
        "brand": "Any Brand",
        "color": "beige"
      },
      "dryer": {
        "brand": "Any Brand",
        "color": "white"
      }
    }
  }
}', 'house', 'appliances', 'washing machine', 'brand');

+------------------------+
| json_extract_path_text |
+------------------------+
| Any Brand              |
+------------------------+
```

Contoh berikut membuat tabel sampel dan mengisinya dengan nilai SUPER, lalu mengembalikan nilai untuk jalur `'f2'` untuk kedua baris.

```
CREATE TABLE json_example(id INT, json_text SUPER);

INSERT INTO json_example VALUES
(1, JSON_PARSE('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}')),
(2, JSON_PARSE('{
    "farm": {
        "barn": {
            "color": "red",
            "feed stocked": true
        }
```

```
    }
}'));

SELECT * FROM json_example;
id          | json_text
------------+--------------------------------------------
1           | {"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}
2           | {"farm":{"barn":{"color":"red","feed stocked":true}}}


SELECT id, JSON_EXTRACT_PATH_TEXT(JSON_SERIALIZE(json_text), 'f2') FROM json_example;

id          | json_text
------------+--------------------------------------------
1           | {"f3":1}
2           |
```

Pertimbangkan contoh pernyataan berikut. Path_elem yang disediakan adalah NULL, jadi JSON_EXTRACT_PATH_TEXT mengembalikan NULL terlepas dari nilai parameter lainnya.

```
--Statement where path_elem is NULL and json_string is valid JSON.
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}',NULL);

 json_extract_path_text
------------------------
                  NULL

--Statement where only one path_elem is NULL.
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}','f4',NULL);

 json_extract_path_text
------------------------
                  NULL

--Statement where path_elem is NULL and json_string is invalid JSON.
SELECT json_extract_path_text('invalid_json', NULL);

 json_extract_path_text
------------------------
                  NULL

--Statement where path_elem is NULL and null_if_invalid is FALSE.
SELECT json_extract_path_text(NULL, 0, FALSE);
```

```
 json_extract_path_text
------------------------
                  NULL
```

Pertimbangkan contoh pernyataan berikut. Ketika null_if_invalid adalah TRUE, JSON_EXTRACT_PATH_TEXT mengembalikan NULL ketika json_string adalah JSON tidak valid. Jika null_if_invalid adalah FALSE atau tidak disetel, fungsi mengembalikan kesalahan ketika json_string tidak valid.

```
--Statement with invalid JSON where null_if_invalid is TRUE.
SELECT json_extract_path_text('invalid_json', 0, TRUE);

 json_extract_path_text
------------------------
                  NULL

--Statement with invalid JSON where null_if_invalid is FALSE.
SELECT json_extract_path_text('invalid_json', 0, FALSE);

ERROR:  JSON parsing error
```

Pertimbangkan contoh berikut, di mana json_string adalah JSON yang valid, dan path_elem mengacu pada nilai JSON. `null` Dalam hal ini, JSON_EXTRACT_PATH_TEXT mengembalikan NULL. Demikian pula, ketika path_elem mengacu pada nilai yang tidak ada, JSON_EXTRACT_PATH_TEXT mengembalikan NULL, terlepas dari nilai null_if_invalid.

```
--Statement selecting a null value.
SELECT json_extract_path_text('[null]', 0);

  json_extract_path_text
-------------------------
                   NULL

--Statement selecting a non-existing value.
SELECT json_extract_path_text('{}', 'a');

  json_extract_path_text
-------------------------
                   NULL
```

Fungsi pembelajaran mesin

Dengan menggunakan pembelajaran mesin Amazon Redshift (ML), Anda dapat melatih model ML menggunakan pernyataan SQL dan memanggilnya dalam kueri SQL untuk prediksi. Penjelasan model Amazon Redshift menyertakan nilai pentingnya fitur untuk membantu Anda memahami bagaimana setiap atribut dalam data pelatihan Anda berkontribusi pada hasil yang diprediksi.

Berikut ini, Anda dapat menemukan deskripsi untuk fungsi pembelajaran mesin untuk SQL yang didukung Amazon Redshift.

Topik

- Fungsi EXPLAIN_MODEL

Fungsi EXPLAIN_MODEL

Fungsi EXPLAIN_MODEL mengembalikan tipe data SUPER yang berisi laporan penjelasan model dalam format JSON. Laporan keterjelasan berisi informasi tentang nilai Shapley untuk semua fitur model.

Fungsi EXPLAIN_MODEL saat ini hanya mendukung model AUTO ON atau AUTO OFF XGBoost.

Ketika laporan keterjelasan tidak tersedia, fungsi mengembalikan status yang ditampilkan pada kemajuan model. Ini termasuk`Waiting for training job to complete`,`Waiting for processing job to complete`, dan`Processing job failed`.

Ketika Anda menjalankan pernyataan CREATE MODEL, status penjelasan menjadi`Waiting for training job to complete`. Ketika model telah dilatih dan permintaan penjelasan dikirim, status penjelasan menjadi`Waiting for processing job to complete`. Ketika penjelasan model berhasil diselesaikan, laporan keterangan lengkap tersedia. Kalau tidak, negara menjadi`Processing job failed`.

Saat menjalankan pernyataan CREATE MODEL, Anda dapat menggunakan `MAX_RUNTIME` parameter opsional untuk menentukan jumlah waktu maksimum yang harus dilakukan pelatihan. Setelah pembuatan model mencapai jumlah waktu tersebut, Amazon Redshift berhenti membuat model. Jika Anda mencapai batas waktu itu saat membuat model autopilot, Amazon Redshift akan mengembalikan model terbaik sejauh ini. Penjelasan model tersedia setelah pelatihan model selesai, jadi jika `MAX_RUNTIME` diatur ke jumlah waktu yang rendah, laporan keterjelasan mungkin tidak tersedia. Waktu pelatihan bervariasi dan tergantung pada kompleksitas model, ukuran data, dan faktor lainnya.

Sintaksis

```
EXPLAIN_MODEL ('schema_name.model_name')
```

Pendapat

nama skema

Nama skema. Jika tidak ada schema_name yang ditentukan, maka skema saat ini dipilih.

nama_model

Nama modul. Nama model dalam skema harus unik.

Jenis pengembalian

Fungsi EXPLAIN_MODEL mengembalikan tipe data SUPER, seperti yang ditunjukkan berikut.

```
{"version":"1.0","explanations":{"kernel_shap":{"label0":{"global_shap_values":
{"x0":0.05,"x1":0.10,"x2":0.30,"x3":0.15},"expected_value":0.50}}}}
```

Contoh

Contoh berikut mengembalikan status penjelasan`waiting for training job to complete`.

```
select explain_model('customer_churn_auto_model');
                 explain_model
------------------------------------------------------------
{"explanations":"waiting for training job to complete"}
(1 row)
```

Ketika penjelasan model berhasil diselesaikan, laporan keterangan lengkap tersedia sebagai berikut.

```
select explain_model('customer_churn_auto_model');
                                        explain_model
-------------------------------------------------------------------------------------------
{"version":"1.0","explanations":{"kernel_shap":{"label0":{"global_shap_values":
{"x0":0.05386043365892927,"x1":0.10801289723274592,"x2":0.23227865827017378,"x3":0.067668513394
(1 row)
```

Karena fungsi EXPLAIN_MODEL mengembalikan tipe data SUPER, Anda dapat menanyakan laporan keterjelasan. Dengan melakukan ini, Anda dapat mengekstrak`global_shap_values`,`expected_value`, atau nilai Shapley khusus fitur.

Contoh berikut mengekstrak `global_shap_values` untuk model.

```
select json_table.report.explanations.kernel_shap.label0.global_shap_values from
 (select explain_model('customer_churn_auto_model') as report) as json_table;
                                                    global_shap_values
-----------------------------------------------------------------------------------------------
{"state":0.10983770427197151,"account_length":0.1772441398408543,"area_code":0.0862682396863959
(1 row)
```

Contoh berikut mengekstrak `global_shap_values` untuk fitur x0.

```
select json_table.report.explanations.kernel_shap.label0.global_shap_values.x0 from
 (select explain_model('customer_churn_auto_model') as report) as json_table;
         x0
------------------------
  0.05386043365892927
(1 row)
```

Jika model dibuat dalam skema tertentu dan Anda memiliki akses ke model yang dibuat, maka Anda dapat menanyakan penjelasan model seperti yang ditunjukkan berikut.

```
-- Check the current schema
SHOW search_path;
   search_path
------------------
  $user, public
(1 row)
-- If you have the privilege to access the model explanation
-- in `test_schema`
SELECT explain_model('test_schema.test_model_name');
                      explain_model
-----------------------------------------------------------
{"explanations":"waiting for training job to complete"}
(1 row)
```

Fungsi matematika

Topik

- Simbol operator matematika
- Fungsi ABS
- Fungsi ACOS
- Fungsi ASIN
- Fungsi ATAN
- Fungsi ATAN2
- Fungsi CBRT
- Fungsi CEILING (atau CEIL)
- Fungsi COS
- Fungsi COT
- Fungsi DEGREES
- Fungsi DEXP
- Fungsi DLOG1
- Fungsi DLOG10
- Fungsi EXP
- Fungsi LANTAI
- Fungsi LN
- Fungsi LOG
- Fungsi MOD
- Fungsi PI
- Fungsi POWER
- Fungsi RADIANS
- Fungsi RANDOM
- Fungsi ROUND
- Fungsi SIN
- Fungsi SIGN
- Fungsi SQRT
- Fungsi TAN

- Fungsi TRUNC

Bagian ini menjelaskan operator matematika dan fungsi yang didukung di Amazon Redshift.

Simbol operator matematika

Tabel berikut mencantumkan operator matematika yang didukung.

Operator yang didukung

Operator	Deskripsi	Contoh	Hasil
+	tambahan	2 + 3	5
-	pengurangan	2 - 3	-1
*	perkalian	2*3	6
/	divisi	4/2	2
%	modul	5% 4	1
^	eksponensiasi	2,0 ^ 3,0	8
|/	akar kuadrat	|/25.0	5
||/	akar kubus	||/27.0	3
@	nilai absolut	@ -5,0	5
<<	geser bit ke kiri	1 << 4	16
>>	geser bit ke kanan	8 >> 2	2
&	bitwise dan	8 & 2	0

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menghitung komisi yang dibayarkan ditambah penanganan $2.00 untuk transaksi tertentu, gunakan contoh berikut.

```
SELECT
    commission,
    (commission + 2.00) AS comm
FROM
    sales
WHERE
    salesid = 10000;

+------------+-------+
| commission | comm  |
+------------+-------+
|      28.05 | 30.05 |
+------------+-------+
```

Untuk menghitung 20 persen dari harga jual untuk transaksi tertentu, gunakan contoh berikut.

```
SELECT pricepaid, (pricepaid * .20) as twentypct
FROM sales
WHERE salesid=10000;

+-----------+-----------+
| pricepaid | twentypct |
+-----------+-----------+
|       187 |      37.4 |
+-----------+-----------+
```

Untuk memperkirakan penjualan tiket berdasarkan pola pertumbuhan berkelanjutan, gunakan contoh berikut. Dalam contoh ini, subquery mengembalikan jumlah tiket yang terjual pada tahun 2008. Hasil itu dikalikan secara eksponensial dengan tingkat pertumbuhan berkelanjutan 5% selama 10 tahun.

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid AND year=2008)^((5::float/100)*10) AS qty10years;
```

```
+------------------+
|    qty10years    |
+------------------+
| 587.664019657491 |
+------------------+
```

Untuk menemukan total harga yang dibayarkan dan komisi untuk penjualan dengan ID tanggal yang lebih besar dari atau sama dengan 2000, gunakan contoh berikut. Kemudian kurangi total komisi dari total harga yang dibayarkan.

```
SELECT SUM(pricepaid) AS sum_price, dateid,
SUM(commission) AS sum_comm, (SUM(pricepaid) - SUM(commission)) AS value
FROM sales
WHERE dateid >= 2000
GROUP BY dateid
ORDER BY dateid
LIMIT 10;

+-----------+--------+----------+-----------+
| sum_price | dateid | sum_comm |   value   |
+-----------+--------+----------+-----------+
|    305885 |   2000 | 45882.75 | 260002.25 |
|    316037 |   2001 | 47405.55 | 268631.45 |
|    358571 |   2002 | 53785.65 | 304785.35 |
|    366033 |   2003 | 54904.95 | 311128.05 |
|    307592 |   2004 |  46138.8 |  261453.2 |
|    333484 |   2005 |  50022.6 |  283461.4 |
|    317670 |   2006 |  47650.5 |  270019.5 |
|    351031 |   2007 | 52654.65 | 298376.35 |
|    313359 |   2008 | 47003.85 | 266355.15 |
|    323675 |   2009 | 48551.25 | 275123.75 |
+-----------+--------+----------+-----------+
```

Fungsi ABS

ABS menghitung nilai absolut dari suatu angka, di mana angka itu dapat berupa literal atau ekspresi yang dievaluasi menjadi angka.

Sintaksis

```
ABS(number)
```

Argumen

jumlah

Angka atau ekspresi yang mengevaluasi ke angka. Ini bisa berupa`SMALLINT`,, `INTEGER`,`BIGINT`,`DECIMAL`,`FLOAT4`,`FLOAT8`, atau `SUPER` tipe.

Jenis pengembalian

ABS mengembalikan tipe data yang sama dengan argumennya.

Contoh

Untuk menghitung nilai absolut`-38`, gunakan contoh berikut.

```
SELECT ABS(-38);

+-----+
| abs |
+-----+
|  38 |
+-----+
```

Untuk menghitung nilai absolut`(14-76)`, gunakan contoh berikut.

```
SELECT ABS(14-76);

+-----+
| abs |
+-----+
|  62 |
+-----+
```

Fungsi ACOS

ACOS adalah fungsi trigonometri yang mengembalikan kosinus busur suatu bilangan. Nilai pengembalian dalam radian dan berada di antara `0` dan`PI`.

Sintaksis

```
ACOS(number)
```

Argumen

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan cosinus busur-1, gunakan contoh berikut.

```
SELECT ACOS(-1);

+-------------------+
|       acos        |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Untuk mengubah cosinus busur .5 ke jumlah derajat yang setara, gunakan contoh berikut.

```
SELECT (ACOS(.5) * 180/(SELECT PI())) AS degrees;

+-------------------+
|      degrees      |
+-------------------+
| 60.00000000000001 |
+-------------------+
```

Fungsi ASIN

ASIN adalah fungsi trigonometri yang mengembalikan sinus busur dari suatu angka. Nilai pengembalian dalam radian dan berada di antara PI/2 dan-PI/2.

Sintaksis

```
ASIN(number)
```

Argumen

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan sinus busur1, gunakan contoh berikut.

```
SELECT ASIN(1) AS halfpi;

+--------------------+
|       halfpi       |
+--------------------+
| 1.5707963267948966 |
+--------------------+
```

Untuk mengubah sinus busur .5 ke jumlah derajat yang setara, gunakan contoh berikut.

```
SELECT (ASIN(.5) * 180/(SELECT PI())) AS degrees;

+--------------------+
|      degrees       |
+--------------------+
| 30.000000000000004 |
+--------------------+
```

Fungsi ATAN

ATAN adalah fungsi trigonometri yang mengembalikan garis tangen busur suatu bilangan. Nilai pengembalian dalam radian dan berada di antara -PI danPI.

Sintaksis

```
ATAN(number)
```

Argumen

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan garis singgung busur 1 dan mengalikannya dengan 4, gunakan contoh berikut.

```
SELECT ATAN(1) * 4 AS pi;

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Untuk mengubah garis singgung busur 1 ke jumlah derajat yang setara, gunakan contoh berikut.

```
SELECT (ATAN(1) * 180/(SELECT PI())) AS degrees;

+---------+
| degrees |
+---------+
|      45 |
+---------+
```

Fungsi ATAN2

ATAN2 adalah fungsi trigonometri yang mengembalikan garis singgung busur dari satu angka dibagi dengan bilangan lain. Nilai pengembalian dalam radian dan berada di antara PI/2 dan-PI/2.

Sintaksis

```
ATAN2(number1, number2)
```

Argumen

nomor1

Sebuah DOUBLE PRECISION angka.

nomor2

Sebuah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan garis singgung busur 2/2 dan mengalikannya dengan 4, gunakan contoh berikut.

```
SELECT ATAN2(2,2) * 4 AS PI;

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Untuk mengubah garis singgung busur 1/0 (yang dievaluasi menjadi 0) ke jumlah derajat yang setara, gunakan contoh berikut.

```
SELECT (ATAN2(1,0) * 180/(SELECT PI())) AS degrees;

+---------+
| degrees |
+---------+
|      90 |
+---------+
```

Fungsi CBRT

Fungsi CBRT adalah fungsi matematika yang menghitung akar kubus dari angka tertentu.

Sintaksis

```
CBRT(number)
```

Argumen

CBRT mengambil DOUBLE PRECISION angka sebagai argumen.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menghitung akar kubus dari komisi yang dibayarkan untuk transaksi tertentu, gunakan contoh berikut.

```
SELECT CBRT(commission) FROM sales WHERE salesid=10000;

+--------------------+
|        cbrt        |
+--------------------+
| 3.0383953904884344 |
+--------------------+
```

Fungsi CEILING (atau CEIL)

Fungsi CEILING atau CEIL digunakan untuk membulatkan angka ke bilangan bulat berikutnya. (Mem Fungsi LANTAI bulatkan angka ke bawah ke bilangan bulat berikutnya.)

Sintaksis

```
{CEIL | CEILING}(number)
```

Argumen

jumlah

Angka atau ekspresi yang dievaluasi menjadi angka. Ini bisa berupaSMALLINT,, INTEGERBIGINT,DECIMAL,FLOAT4,FLOAT8, atau SUPER tipe.

Jenis pengembalian

CEILING dan CEIL mengembalikan tipe data yang sama dengan argumennya.

Ketika input adalah SUPER tipe, output mempertahankan tipe dinamis yang sama dengan input sementara tipe statis tetap tipe SUPER. Jika tipe dinamis SUPER bukan angka, Amazon Redshift mengembalikan nol.

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menghitung plafon komisi yang dibayarkan untuk transaksi penjualan tertentu, gunakan contoh berikut.

```
SELECT CEILING(commission) FROM sales
WHERE salesid=10000;

+---------+
| ceiling |
+---------+
|      29 |
+---------+
```

Fungsi COS

COS adalah fungsi trigonometri yang mengembalikan kosinus suatu bilangan. Nilai pengembalian dalam radian dan antara -1 dan1, inklusif.

Sintaksis

```
COS(double_precision)
```

Argumen

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

Fungsi COS mengembalikan `DOUBLE PRECISION` angka.

Contoh

Untuk mengembalikan kosinus`0`, gunakan contoh berikut.

```
SELECT COS(0);

+-----+
| cos |
+-----+
|   1 |
+-----+
```

Untuk mengembalikan kosinus`pi`, gunakan contoh berikut.

```
SELECT COS(PI());

+-----+
| cos |
+-----+
|  -1 |
+-----+
```

Fungsi COT

COT adalah fungsi trigonometri yang mengembalikan kotangen suatu bilangan. Parameter input harus bukan nol.

Sintaksis

```
COT(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan cotangen dari 1, gunakan contoh berikut.

```
SELECT COT(1);

+--------------------+
|        cot         |
+--------------------+
| 0.6420926159343306 |
+--------------------+
```

Fungsi DEGREES

Mengkonversi sudut dalam radian menjadi setara dalam derajat.

Sintaksis

```
DEGREES(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan derajat setara 0,5 radian, gunakan contoh berikut.

```
SELECT DEGREES(.5);

+-------------------+
|      degrees      |
```

```
+-------------------+
| 28.64788975654116 |
+-------------------+
```

Untuk mengubah radian PI menjadi derajat, gunakan contoh berikut.

```
SELECT DEGREES(pi());

+---------+
| degrees |
+---------+
|     180 |
+---------+
```

Fungsi DEXP

Fungsi DEXP mengembalikan nilai eksponensial dalam notasi ilmiah untuk angka presisi ganda. Satu-satunya perbedaan antara fungsi DEXP dan EXP adalah bahwa parameter untuk DEXP harus a. `DOUBLE PRECISION`

Sintaksis

```
DEXP(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Gunakan fungsi DEXP untuk memperkirakan penjualan tiket berdasarkan pola pertumbuhan yang berkelanjutan. Dalam contoh ini, subquery mengembalikan jumlah tiket yang terjual pada tahun 2008.

Hasil itu dikalikan dengan hasil fungsi DEXP, yang menentukan tingkat pertumbuhan berkelanjutan sebesar 7% selama 10 tahun.

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * DEXP((7::FLOAT/100)*10) qty2010;

+------------------+
|      qty2010     |
+------------------+
| 695447.4837722216 |
+------------------+
```

Fungsi DLOG1

Fungsi DLOG1 mengembalikan logaritma alami dari parameter input. Sinonim dariFungsi LN.

Fungsi DLOG10

DLOG10 mengembalikan logaritma dasar 10 dari parameter input.

Sinonim dariFungsi LOG.

Sintaksis

```
DLOG10(number)
```

Pendapat

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan logaritma basis 10 dari angka 100, gunakan contoh berikut.

```
SELECT DLOG10(100);

+--------+
| dlog10 |
+--------+
|      2 |
+--------+
```

Fungsi EXP

Fungsi EXP mengimplementasikan fungsi eksponensial untuk ekspresi numerik, atau basis logaritma alami,e, yang dinaikkan ke kekuatan ekspresi. Fungsi EXP adalah kebalikan dariFungsi LN.

Sintaksis

```
EXP(expression)
```

Pendapat

ekspresi

Ekspresi harus berupaINTEGER,DECIMAL, atau tipe DOUBLE PRECISION data.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Gunakan fungsi EXP untuk memperkirakan penjualan tiket berdasarkan pola pertumbuhan berkelanjutan. Dalam contoh ini, subquery mengembalikan jumlah tiket yang terjual pada tahun 2008. Hasil itu dikalikan dengan hasil fungsi EXP, yang menentukan tingkat pertumbuhan berkelanjutan 7% selama 10 tahun.

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * EXP((7::FLOAT/100)*10) qty2018;
```

```
+-------------------+
|      qty2018      |
+-------------------+
| 695447.4837722216 |
+-------------------+
```

Fungsi LANTAI

Fungsi FLOOR membulatkan angka ke bawah ke bilangan bulat berikutnya.

Sintaksis

```
FLOOR(number)
```

Pendapat

jumlah

Angka atau ekspresi yang dievaluasi menjadi angka. Ini bisa berupa`SMALLINT`,, `INTEGER`,`BIGINT`,`DECIMAL`,`FLOAT4`,`FLOAT8`, atau `SUPER` tipe.

Jenis pengembalian

FLOOR mengembalikan tipe data yang sama dengan argumennya.

Ketika input adalah `SUPER` tipe, output mempertahankan tipe dinamis yang sama dengan input sementara tipe statis tetap `SUPER` tipe. Jika tipe dinamis `SUPER` bukan angka, Amazon Redshift kembali`NULL`.

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menunjukkan nilai komisi yang dibayarkan untuk transaksi penjualan tertentu sebelum dan sesudah menggunakan fungsi FLOOR, gunakan contoh berikut.

```
SELECT commission
FROM sales
WHERE salesid=10000;
```

```
+------------+
| commission |
+------------+
|      28.05 |
+------------+

SELECT FLOOR(commission)
FROM sales
WHERE salesid=10000;

+-------+
| floor |
+-------+
|    28 |
+-------+
```

Fungsi LN

Mengembalikan logaritma alami dari parameter input.

Sinonim dariFungsi DLOG1.

Sintaksis

```
LN(expression)
```

Pendapat

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

> **Note**
>
> Fungsi ini mengembalikan kesalahan untuk beberapa tipe data jika ekspresi mereferensikan tabel buatan pengguna Amazon Redshift atau tabel sistem STL atau STV Amazon Redshift.

Ekspresi dengan tipe data berikut menghasilkan kesalahan jika mereka mereferensikan tabel yang dibuat pengguna atau sistem. Ekspresi dengan tipe data ini berjalan secara eksklusif pada node pemimpin:

- BOOLEAN
- CHAR
- DATE
- DECIMAL atau NUMERIC
- TIMESTAMP
- VARCHAR

Ekspresi dengan tipe data berikut berhasil berjalan pada tabel yang dibuat pengguna dan tabel sistem STL atau STV:

- BIGINT
- DOUBLE PRECISION
- INTEGER
- REAL
- SMALLINT

Jenis pengembalian

Fungsi LN mengembalikan tipe yang sama dengan ekspresi input.

Contoh

Untuk mengembalikan logaritma alami atau logaritma dasar dari angka 2.7 e 18281828, gunakan contoh berikut.

```
SELECT LN(2.718281828);

+--------------------+
|         ln         |
+--------------------+
| 0.9999999998311267 |
+--------------------+
```

Perhatikan bahwa jawabannya hampir sama dengan 1.

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan logaritma alami dari nilai-nilai di kolom userid di tabel USERS, gunakan contoh berikut.

```
SELECT username, LN(userid) FROM users ORDER BY userid LIMIT 10;

+----------+--------------------+
| username |         ln         |
+----------+--------------------+
| JSG99FHE |                  0 |
| PGL08LJI | 0.6931471805599453 |
| IFT66TXU | 1.0986122886681098 |
| XDZ38RDD | 1.3862943611198906 |
| AEB55QTM | 1.6094379124341003 |
| NDQ15VBM |  1.791759469228055 |
| OWY35QYB | 1.9459101490553132 |
| AZG78YIP | 2.0794415416798357 |
| MSD36KVR | 2.1972245773362196 |
| WKW41AIW |  2.302585092994046 |
+----------+--------------------+
```

Fungsi LOG

Mengembalikan logaritma dari angka.

Jika Anda menggunakan fungsi ini untuk menghitung logaritma dasar 10, Anda juga dapat menggunakannya. Fungsi DLOG10

Sintaksis

```
LOG([base, ]argument)
```

Parameter

dasar

(Opsional) Dasar fungsi logaritma. Angka ini harus positif dan tidak bisa sama1. Jika parameter ini dihilangkan, Amazon Redshift menghitung logaritma dasar 10 dari argumen.

argumen

Argumen fungsi logaritma. Angka ini harus positif. Jika nilai argumennya adalah1, fungsi kembali0.

Jenis pengembalian

Fungsi LOG mengembalikan DOUBLE PRECISION angka.

Contoh

Untuk menemukan logaritma dasar 2 dari 100, gunakan contoh berikut.

```
SELECT LOG(2, 100);
+-------------------+
|        log        |
+-------------------+
| 6.643856189774725 |
+-------------------+
```

Untuk menemukan logaritma dasar 10 dari 100, gunakan contoh berikut. Perhatikan bahwa jika Anda menghilangkan parameter dasar, Amazon Redshift mengasumsikan basis 10.

```
SELECT LOG(100);

+-----+
| log |
+-----+
|   2 |
+-----+
```

Fungsi MOD

Mengembalikan sisa dari dua angka, atau dikenal sebagai operasi modulo. Untuk menghitung hasilnya, parameter pertama dibagi dengan yang kedua.

Sintaksis

```
MOD(number1, number2)
```

Argumen

nomor1

Parameter input pertama adalahINTEGER,SMALLINT,BIGINT, atau DECIMAL angka. Jika salah satu parameter adalah DECIMAL tipe, parameter lainnya juga harus berupa DECIMAL tipe. Jika

salah satu parameter adalah a`INTEGER`, parameter lainnya dapat berupa`INTEGER`,`SMALLINT`, atau`BIGINT`. Kedua parameter juga bisa `SMALLINT` atau`BIGINT`, tetapi satu parameter tidak bisa menjadi a `SMALLINT` jika yang lain adalah a`BIGINT`.

nomor2

Parameter kedua adalah `INTEGER` angka`SMALLINT`,`BIGINT`,, atau`DECIMAL`. Aturan tipe data yang sama berlaku untuk number2 seperti untuk number 1.

Jenis pengembalian

Jenis pengembalian fungsi MOD adalah tipe numerik yang sama dengan parameter input, jika kedua parameter input adalah tipe yang sama. Namun`INTEGER`, jika salah satu parameter input adalah, tipe pengembalian juga akan menjadi`INTEGER`. Jenis pengembalian yang valid adalah `DECIMAL`,`INT`,`SMALLINT`,, dan`BIGINT`.

Catatan penggunaan

Anda dapat menggunakan % sebagai operator modulo.

Contoh

Untuk mengembalikan sisanya ketika angka dibagi dengan yang lain, gunakan contoh berikut.

```
SELECT MOD(10, 4);

+-----+
| mod |
+-----+
|   2 |
+-----+
```

Untuk mengembalikan `DECIMAL` hasil saat menggunakan fungsi MOD, gunakan contoh berikut.

```
SELECT MOD(10.5, 4);

+-----+
| mod |
+-----+
| 2.5 |
+-----+
```

Untuk memberikan angka sebelum menjalankan fungsi MOD, gunakan contoh berikut. Untuk informasi selengkapnya, lihat Fungsi CAST.

```
SELECT MOD(CAST(16.4 AS INTEGER), 5);

+-----+
| mod |
+-----+
|   1 |
+-----+
```

Untuk memeriksa apakah parameter pertama genap dengan membaginya dengan 2, gunakan contoh berikut.

```
SELECT mod(5,2) = 0 AS is_even;

+---------+
| is_even |
+---------+
| false   |
+---------+
```

Untuk menggunakan% sebagai operator modulo, gunakan contoh berikut.

```
SELECT 11 % 4 as remainder;

 +-----------+
| remainder |
+-----------+
|         3 |
+-----------+
```

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan informasi untuk kategori bernomor ganjil dalam tabel KATEGORI, gunakan contoh berikut.

```
SELECT catid, catname
FROM category
WHERE MOD(catid,2)=1
```

```
ORDER BY 1,2;

+-------+-----------+
| catid |  catname  |
+-------+-----------+
|     1 | MLB       |
|     3 | NFL       |
|     5 | MLS       |
|     7 | Plays     |
|     9 | Pop       |
|    11 | Classical |
+-------+-----------+
```

Fungsi PI

Fungsi PI mengembalikan nilai pi ke 14 tempat desimal.

Sintaksis

```
PI()
```

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan nilai pi, gunakan contoh berikut.

```
SELECT PI();

+------------------+
|        pi        |
+------------------+
| 3.141592653589793 |
+------------------+
```

Fungsi POWER

Fungsi POWER adalah fungsi eksponensial yang meningkatkan ekspresi numerik ke pangkat ekspresi numerik kedua. Misalnya, 2 hingga daya ketiga dihitung sebagai`POWER(2,3)`, dengan hasil dari8.

Sintaksis

```
{POW | POWER}(expression1, expression2)
```

Argumen

Ekspresi1

Ekspresi numerik yang akan dinaikkan. Harus berupa`INTEGER`,`DECIMAL`, atau tipe `FLOAT` data.

ekspressi2

Kekuatan untuk meningkatkan ekspresi 1. Harus berupa`INTEGER`,`DECIMAL`, atau tipe `FLOAT` data.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Dalam contoh berikut, fungsi POWER digunakan untuk memperkirakan seperti apa penjualan tiket dalam 10 tahun ke depan, berdasarkan jumlah tiket yang terjual pada tahun 2008 (hasil subquery). Tingkat pertumbuhan ditetapkan sebesar 7% per tahun dalam contoh ini.

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100),10) qty2010;

+-------------------+
|      qty2010      |
+-------------------+
| 679353.7540885945 |
+-------------------+
```

Contoh berikut adalah variasi pada contoh sebelumnya, dengan tingkat pertumbuhan 7% per tahun tetapi interval diatur ke bulan (120 bulan selama 10 tahun).

```
SELECT (SELECT SUM(qtysold) FROM sales, date
```

```
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100/12),120) qty2010;

+-----------------+
|     qty2010     |
+-----------------+
| 694034.54678046 |
+-----------------+
```

Fungsi RADIANS

Fungsi RADIANS mengubah sudut dalam derajat menjadi setara dalam radian.

Sintaksis

```
RADIANS(number)
```

Pendapat

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan ekuivalen radian 180 derajat, gunakan contoh berikut.

```
SELECT RADIANS(180);

+-------------------+
|      radians      |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Fungsi RANDOM

Fungsi RANDOM menghasilkan nilai acak antara 0.0 (inklusif) dan 1.0 (eksklusif).

Sintaksis

```
RANDOM()
```

Jenis pengembalian

`DOUBLE PRECISION`

Catatan penggunaan

Fungsi RANDOM dapat disemai menggunakan nilai SET seed TO. Saat disemai, setiap irisan komputasi menghasilkan urutan pseudo-random yang dapat direproduksi sendiri. Namun, hasil kueri keseluruhan tidak deterministik dalam lingkungan MPP terdistribusi karena Anda tidak dapat memprediksi irisan mana yang akan memproses baris mana. Nilai benih mempengaruhi generator bilangan acak pada setiap node komputasi, tetapi distribusi baris di seluruh irisan bervariasi.

Contoh

Untuk menghitung nilai acak antara 0 dan 99, gunakan contoh berikut. Jika angka acak adalah 0 hingga 1, kueri ini menghasilkan angka acak dari 0 hingga 100.

```
SELECT CAST(RANDOM() * 100 AS INT);

+------+
| int4 |
+------+
|   59 |
+------+
```

Contoh ini menggunakan SET perintah untuk mengatur nilai SEED sehingga RANDOM menghasilkan urutan angka yang dapat diprediksi.

Untuk mengembalikan tiga bilangan bulat RANDOM tanpa menetapkan nilai SEED, gunakan contoh berikut.

```
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|    6 |
```

```
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   68 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   56 |
+------+
```

Untuk mengatur nilai SEED ke.25, dan mengembalikan tiga angka RANDOM lagi, gunakan contoh berikut.

```
SET SEED TO .25;
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   21 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   79 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   12 |
+------+
```

Untuk mengatur ulang nilai SEED ke.25, dan memverifikasi bahwa RANDOM mengembalikan hasil yang sama dengan tiga panggilan sebelumnya, gunakan contoh berikut.

```
SET SEED TO .25;
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   21 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   79 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   12 |
+------+
```

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengambil sampel acak seragam dari 10 item dari tabel PENJUALAN, gunakan contoh berikut.

```
SELECT *
FROM sales
ORDER BY RANDOM()
LIMIT 10;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|   45422 |  51114 |     5983 |   24482 |    4369 |   2118 |       1 |       195 |
 29.25 | 2008-10-19 05:20:07 |
|   42481 |  47638 |     4573 |    6198 |    6479 |   1987 |       4 |      1140 |
   171 | 2008-06-10 09:39:19 |
```

```
|   31494 |  34759 |    18895 |    4719 |    7753 |   2090 |       4 |      1024 |
 153.6 | 2008-09-21 03:44:26 |
|  119388 | 136685 |    21815 |   41905 |    2071 |   1884 |       1 |       359 |
 53.85 | 2008-02-27 10:43:10 |
|  166990 | 225037 |    18529 |    7628 |     746 |   2113 |       1 |      2009 |
 301.35 | 2008-10-14 10:07:44 |
|   11146 |  12096 |    42685 |    6619 |    1876 |   2123 |       1 |        29 |
  4.35 | 2008-10-24 06:23:54 |
|  148537 | 172056 |    15102 |   11787 |    6122 |   1923 |       2 |       480 |
    72 | 2008-04-07 03:58:23 |
|   68945 |  78387 |     7359 |   18323 |    6636 |   1910 |       1 |       457 |
 68.55 | 2008-03-25 08:31:03 |
|   52796 |  59576 |     9909 |   15102 |    7958 |   1951 |       1 |       479 |
 71.85 | 2008-05-05 02:25:08 |
|   90684 | 103522 |    38052 |   21549 |    7384 |   2117 |       1 |       313 |
 46.95 | 2008-10-18 05:43:11 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

Untuk mengambil sampel acak dari 10 item, tetapi pilih item secara proporsional dengan harganya, gunakan contoh berikut. Misalnya, item yang dua kali harga yang lain akan dua kali lebih mungkin muncul dalam hasil kueri.

```
SELECT *
FROM sales
ORDER BY -LOG(RANDOM()) / pricepaid
LIMIT 10;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|  158340 | 208208 |    17082 |   42018 |    1211 |   2160 |       4 |      6852 |
 1027.8 | 2008-11-30 12:21:43 |
|   53250 |  60069 |    12644 |    7066 |    7942 |   1838 |       4 |      1528 |
 229.2 | 2008-01-12 11:24:56 |
|   22929 |  24938 |    47314 |    6503 |     179 |   2000 |       3 |       741 |
 111.15 | 2008-06-23 08:04:50 |
|  164980 | 221181 |     1949 |   19670 |    1471 |   1906 |       1 |      1330 |
 199.5 | 2008-03-21 07:59:51 |
```

```
|  159641 | 211179 |    44897 |   16652 |    7458 |   2128 |       1 |      1019 |
 152.85 | 2008-10-29 02:02:15 |
|   73143 |  83439 |     5716 |    5727 |    7314 |   1903 |       1 |       248 |
  37.2 | 2008-03-18 11:07:42 |
|   84778 |  96749 |    46608 |   32980 |    3883 |   1999 |       2 |       958 |
 143.7 | 2008-06-22 12:13:31 |
|  171096 | 232929 |    43683 |    8536 |    8353 |   1870 |       1 |       929 |
 139.35 | 2008-02-13 01:36:36 |
|   74212 |  84697 |    39809 |   15569 |    5525 |   2105 |       2 |       896 |
 134.4 | 2008-10-06 11:47:50 |
|  158011 | 207556 |    25399 |   16881 |     232 |   2088 |       2 |      2526 |
 378.9 | 2008-09-19 06:00:26 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

Fungsi ROUND

Fungsi ROUND membulatkan angka ke bilangan bulat atau desimal terdekat.

Fungsi ROUND secara opsional dapat menyertakan argumen kedua sebagai `INTEGER` untuk menunjukkan jumlah tempat desimal untuk pembulatan, di kedua arah. Ketika Anda tidak memberikan argumen kedua, fungsi dibulatkan ke bilangan bulat terdekat. Ketika argumen kedua integer ditentukan, fungsi dibulatkan ke angka terdekat dengan tempat desimal integer presisi.

Sintaksis

```
ROUND(number [ , integer ] )
```

Argumen

jumlah

Angka atau ekspresi yang mengevaluasi menjadi angka. Itu bisa berupa`DECIMAL`, `FLOAT8` atau `SUPER` tipe. Amazon Redshift dapat secara implisit mengonversi tipe data numerik lainnya.

bilangan bulat

(Opsional) An `INTEGER` yang menunjukkan jumlah tempat desimal untuk pembulatan di kedua arah. Tipe `SUPER` data tidak didukung untuk argumen ini.

Jenis pengembalian

ROUND mengembalikan tipe data numerik yang sama dengan nomor input.

Ketika input adalah SUPER tipe, output mempertahankan tipe dinamis yang sama dengan input sementara tipe statis tetap SUPER tipe. Jika tipe dinamis SUPER bukan angka, Amazon Redshift kembaliNULL.

Contoh

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk membulatkan komisi yang dibayarkan untuk transaksi tertentu ke bilangan bulat terdekat, gunakan contoh berikut.

```
SELECT commission, ROUND(commission)
FROM sales WHERE salesid=10000;

+------------+-------+
| commission | round |
+------------+-------+
|      28.05 |    28 |
+------------+-------+
```

Untuk membulatkan komisi yang dibayarkan untuk transaksi tertentu ke tempat desimal pertama, gunakan contoh berikut.

```
SELECT commission, ROUND(commission, 1)
FROM sales WHERE salesid=10000;

+------------+-------+
| commission | round |
+------------+-------+
|      28.05 |  28.1 |
+------------+-------+
```

Untuk memperluas presisi ke arah yang berlawanan seperti contoh sebelumnya, gunakan contoh berikut.

```
SELECT commission, ROUND(commission, -1)
FROM sales WHERE salesid=10000;
```

```
+------------+-------+
| commission | round |
+------------+-------+
|      28.05 |    30 |
+------------+-------+
```

Fungsi SIN

SIN adalah fungsi trigonometri yang mengembalikan sinus suatu bilangan. Nilai pengembalian adalah antara `-1` dan`1`.

Sintaksis

```
SIN(number)
```

Pendapat

jumlah

Ang `DOUBLE PRECISION` ka dalam radian.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan sinus`-PI`, gunakan contoh berikut.

```
SELECT SIN(-PI());

+-------------------------+
|           sin           |
+-------------------------+
| -0.00000000000000012246 |
+-------------------------+
```

Fungsi SIGN

Fungsi SIGN mengembalikan tanda (positif atau negatif) dari suatu angka. Hasil dari fungsi SIGN adalah 1 jika argumennya positif, `-1` jika argumennya negatif, atau `0` jika argumennya positif`0`.

Sintaksis

```
SIGN(number)
```

Pendapat

jumlah

Angka atau ekspresi yang mengevaluasi ke angka. Itu bisa berupa`DECIMAL`,`FLOAT8`, atau `SUPER` tipe. Amazon Redshift dapat mengonversi tipe data lain sesuai aturan konversi implisit.

Jenis pengembalian

SIGN mengembalikan tipe data numerik yang sama dengan argumen input. Jika input adalah`DECIMAL`, outputnya adalah`DECIMAL(1,0)`.

Ketika input adalah `SUPER` tipe, output mempertahankan tipe dinamis yang sama dengan input sementara tipe statis tetap `SUPER` tipe. Jika tipe dinamis `SUPER` bukan angka, Amazon Redshift mengembalikan a`NULL`.

Contoh

Contoh berikut menunjukkan bahwa kolom d dalam tabel t2 memiliki `DOUBLE PRECISION` jenisnya karena inputnya adalah `DOUBLE PRECISION` dan kolom n dalam tabel t2 memiliki `NUMERIC(1,0)` output karena inputnya adalah`NUMERIC`.

```
CREATE TABLE t1(d DOUBLE PRECISION, n NUMERIC(12, 2));
INSERT INTO t1 VALUES (4.25, 4.25), (-4.25, -4.25);
CREATE TABLE t2 AS SELECT SIGN(d) AS d, SIGN(n) AS n FROM t1;
SELECT table_name, column_name, data_type FROM SVV_REDSHIFT_COLUMNS WHERE
 table_name='t1' OR table_name='t2';

+------------+-------------+----------------------+
| table_name | column_name |       data_type      |
+------------+-------------+----------------------+
| t1         | d           | double precision     |
| t1         | n           | numeric(12,2)        |
| t2         | d           | double precision     |
| t2         | n           | numeric(1,0)         |
| t1         | col1        | character varying(20) |
+------------+-------------+----------------------+
```

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menentukan tanda komisi yang dibayarkan untuk transaksi tertentu dari tabel PENJUALAN, gunakan contoh berikut.

```
SELECT commission, SIGN(commission)
FROM sales WHERE salesid=10000;

+------------+------+
| commission | sign |
+------------+------+
|      28.05 |    1 |
+------------+------+
```

Fungsi SQRT

Fungsi SQRT mengembalikan akar kuadrat dari suatu NUMERIC nilai. Akar kuadrat adalah angka yang dikalikan dengan sendirinya untuk mendapatkan nilai yang diberikan.

Sintaksis

```
SQRT(expression)
```

Pendapat

ekspresi

Ekspresi harus memilikiINTEGER,DECIMAL, atau tipe FLOAT data, atau tipe data yang secara implisit mengonversi ke tipe data tersebut. Ek spresi dapat mencakup fungsi.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan akar kuadrat dari 16, gunakan contoh berikut.

```
SELECT SQRT(16);
```

```
+------+
| sqrt |
+------+
|    4 |
+------+
```

Untuk mengembalikan akar kuadrat string 16 menggunakan konversi tipe implisit, gunakan contoh berikut.

```
SELECT SQRT('16');

+------+
| sqrt |
+------+
|    4 |
+------+
```

Untuk mengembalikan akar kuadrat 16,4 setelah menggunakan fungsi ROUND, gunakan contoh berikut.

```
SELECT SQRT(ROUND(16.4));

+------+
| sqrt |
+------+
|    4 |
+------+
```

Untuk mengembalikan panjang radius ketika diberi luas lingkaran, gunakan contoh berikut. Ini menghitung radius dalam inci, misalnya, ketika diberikan luas dalam inci persegi. Luas dalam sampel adalah 20.

```
SELECT SQRT(20/PI()) AS radius;

+--------------------+
|      radius        |
+--------------------+
| 2.5231325220201604 |
+--------------------+
```

Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan akar kuadrat untuk nilai COMMISSION dari tabel SALES, gunakan contoh berikut. Kolom KOMISI adalah `DECIMAL` kolom. Contoh ini menunjukkan bagaimana Anda dapat menggunakan fungsi dalam kueri dengan logika kondisional yang lebih kompleks.

```
SELECT SQRT(commission)
FROM sales WHERE salesid < 10 ORDER BY salesid;

+--------------------+
|        sqrt        |
+--------------------+
| 10.449880382090505 |
| 3.3763886032268267 |
|  7.245688373094719 |
|  5.123475382979799 |
|  4.806245936279167 |
|  7.687652437513028 |
| 10.871982339941507 |
| 5.4359911699707535 |
|   9.41541289588513 |
+--------------------+
```

Untuk mengembalikan akar kuadrat bulat untuk set nilai COMMISSION yang sama, gunakan contoh berikut.

```
SELECT ROUND(SQRT(commission))
FROM sales WHERE salesid < 10 ORDER BY salesid;

+-------+
| round |
+-------+
|    10 |
|     3 |
|     7 |
|     5 |
|     5 |
|     8 |
|    11 |
|     5 |
|     9 |
```

```
+-------+
```

Fungsi TAN

TAN adalah fungsi trigonometri yang mengembalikan garis singgung suatu bilangan. Argumen input adalah angka (dalam radian).

Sintaksis

```
TAN(number)
```

Pendapat

jumlah

Sebuah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh

Untuk mengembalikan garis singgung nol, gunakan contoh berikut.

```
SELECT TAN(0);

+-----+
| tan |
+-----+
|   0 |
+-----+
```

Fungsi TRUNC

Fungsi TRUNC memotong angka ke bilangan bulat atau desimal sebelumnya.

Fungsi TRUNC secara opsional dapat menyertakan argumen kedua sebagai INTEGER untuk menunjukkan jumlah tempat desimal untuk pembulatan, di kedua arah. Ketika Anda tidak memberikan argumen kedua, fungsi dibulatkan ke bilangan bulat terdekat. Ketika argumen kedua integer ditentukan, fungsi dibulatkan ke angka terdekat dengan tempat desimal integer presisi.

Fungsi ini juga dapat memotong a TIMESTAMP dan mengembalikan aDATE. Untuk informasi selengkapnya, lihat Fungsi TRUNC.

Sintaks

```
TRUNC(number [ , integer ])
```

Argumen

jumlah

Angka atau ekspresi yang mengevaluasi menjadi angka. Itu bisa berupaDECIMAL, FLOAT8 atau SUPER tipe. Amazon Redshift dapat mengonversi tipe data lain sesuai aturan konversi implisit.

bilangan bulat

(Opsional) An INTEGER yang menunjukkan jumlah tempat desimal presisi, di kedua arah. Jika tidak ada bilangan bulat yang disediakan, angka tersebut dipotong sebagai bilangan bulat; jika bilangan bulat ditentukan, angka tersebut dipotong ke tempat desimal yang ditentukan. Ini tidak didukung untuk tipe SUPER data.

Jenis pengembalian

TRUNC mengembalikan tipe data yang sama dengan nomor input.

Ketika input adalah SUPER tipe, output mempertahankan tipe dinamis yang sama dengan input sementara tipe statis tetap SUPER tipe. Jika tipe dinamis SUPER bukan angka, Amazon Redshift kembaliNULL.

Contoh

Beberapa contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk memotong komisi yang dibayarkan untuk transaksi penjualan tertentu, gunakan contoh berikut.

```
SELECT commission, TRUNC(commission)
FROM sales WHERE salesid=784;

+------------+-------+
```

```
| commission | trunc |
+------------+-------+
|     111.15 |   111 |
+------------+-------+
```

Untuk memotong nilai komisi yang sama ke tempat desimal pertama, gunakan contoh berikut.

```
SELECT commission, TRUNC(commission,1)
FROM sales WHERE salesid=784;

+------------+-------+
| commission | trunc |
+------------+-------+
|     111.15 | 111.1 |
+------------+-------+
```

Untuk memotong komisi dengan nilai negatif untuk argumen kedua, gunakan contoh berikut. Perhatikan bahwa `111.15` dibulatkan ke bawah ke`110`.

```
SELECT commission, TRUNC(commission,-1)
FROM sales WHERE salesid=784;

+------------+-------+
| commission | trunc |
+------------+-------+
|     111.15 |   110 |
+------------+-------+
```

Fungsi objek

Berikut ini adalah fungsi objek SQL yang didukung Amazon Redshift untuk membuat dan mengoperasikan objek tipe SUPER:

Topik

- Fungsi GET_NUMBER_ATTRIBUTES
- Fungsi LOWER_ATTRIBUTE_NAMES
- Fungsi OBJEK
- Fungsi OBJECT_TRANSFORM
- Fungsi UPPER_ATTRIBUTE_NAMES

Fungsi GET_NUMBER_ATTRIBUTES

Mengembalikan hitungan berapa banyak pasangan kunci-nilai yang ada di tingkat akar objek kamus.

Sintaksis

```
GET_NUMBER_ATTRIBUTES( super_expression )
```

Argumen

super_ekspresi

Ekspresi SUPER dari bentuk kamus.

Jenis pengembalian

Fungsi GET_NUMBER_ATTRIBUTES mengembalikan tipe INT.

Catatan

Fungsi ini hanya menghitung atribut langsung dan tidak menyertakan pasangan dalam kamus bersarang.

Contoh

Contoh berikut menunjukkan fungsi GET_NUMBER_ATTRIBUTES.

```
SELECT GET_NUMBER_ATTRIBUTES(JSON_PARSE('{"a": 1, "b": 2, "c": 3}'));
 get_number_attributes
-----------------------
            3
(1 row)
```

Fungsi GET_NUMBER_ATTRIBUTES hanya beroperasi pada tingkat pertama kamus.

```
SELECT GET_NUMBER_ATTRIBUTES(JSON_PARSE('{"a": 1, "b": {"c": 3}}'));
 get_number_attributes
-----------------------
            2
(1 row)
```

Fungsi LOWER_ATTRIBUTE_NAMES

Mengonversi semua nama atribut yang berlaku dalam nilai SUPER menjadi huruf kecil, menggunakan rutinitas konversi kasus yang sama seperti. Fungsi LOWER LOWER_ATTRIBUTE_NAMES mendukung karakter UTF-8 multibyte, hingga maksimal empat byte per karakter.

Untuk mengonversi nama atribut SUPER menjadi huruf besar, gunakan. Fungsi UPPER_ATTRIBUTE_NAMES

Sintaksis

```
LOWER_ATTRIBUTE_NAMES( super_expression )
```

Argumen

super_ekspresi

Ekspresi SUPER.

Jenis pengembalian

SUPER

Catatan penggunaan

Di Amazon Redshift, pengidentifikasi kolom biasanya tidak peka huruf besar/kecil dan diubah menjadi huruf kecil. Jika Anda mengambil data dari format data peka huruf besar-kecil seperti JSON, data tersebut mungkin berisi nama atribut huruf campuran.

Pertimbangkan contoh berikut.

```
CREATE TABLE t1 (s) AS SELECT JSON_PARSE('{"AttributeName": "Value"}');


SELECT s.AttributeName FROM t1;

attributename
-------------
NULL
```

```
SELECT s."AttributeName" FROM t1;

attributename
-------------
NULL
```

Amazon Redshift mengembalikan NULL untuk kedua kueri. Untuk menanyakan`AttributeName`, gunakan LOWER_ATTRIBUTE_NAMES untuk mengonversi nama atribut data menjadi huruf kecil. Pertimbangkan contoh berikut.

```
CREATE TABLE t2 (s) AS SELECT LOWER_ATTRIBUTE_NAMES(s) FROM t1;


SELECT s.attributename FROM t2;

attributename
-------------
"Value"


SELECT s.AttributeName FROM t2;

attributename
-------------
"Value"


SELECT s."attributename" FROM t2;

attributename
-------------
"Value"


SELECT s."AttributeName" FROM t2;

attributename
-------------
"Value"
```

Opsi terkait untuk bekerja dengan nama atribut objek kasus campuran adalah opsi `enable_case_sensitive_super_attribute` konfigurasi, yang memungkinkan

Amazon Redshift mengenali kasus dalam nama atribut SUPER. Ini bisa menjadi solusi alternatif untuk menggunakan LOWER_ATTRIBUTE_NAMES. Untuk informasi lebih lanjut tentang`enable_case_sensitive_super_attribute`, buka[enable_case_sensitive_super_atribut](#).

Contoh

Mengkonversi nama atribut SUPER menjadi huruf kecil

Contoh berikut menggunakan LOWER_ATTRIBUTE_NAMES untuk mengonversi nama atribut dari semua nilai SUPER dalam tabel.

```
-- Create a table and insert several SUPER values.
CREATE TABLE t (i INT, s SUPER);

INSERT INTO t VALUES
  (1, NULL),
  (2, 'A'::SUPER),
  (3, JSON_PARSE('{"AttributeName": "B"}')),
  (4, JSON_PARSE(
     '[{"Subobject": {"C": "C"},
       "Subarray": [{"D": "D"}, "E"]
      }]'));

-- Convert all attribute names to lowercase.
UPDATE t SET s = LOWER_ATTRIBUTE_NAMES(s);

SELECT i, s FROM t ORDER BY i;

 i |                         s
---+-------------------------------------------------
 1 | NULL
 2 | "A"
 3 | {"attributename":"B"}
 4 | [{"subobject":{"c":"C"},"subarray":[{"d":"D"}, "E"]}]
```

Amati bagaimana LOWER_ATTRIBUTE_NAMES berfungsi.

- Nilai NULL dan nilai SUPER sk `"A"` alar seperti tidak berubah.
- Dalam objek SUPER, semua nama atribut diubah menjadi huruf kecil, tetapi nilai atribut seperti `"B"` tetap tidak berubah.
- LOWER_ATTRIBUTE_NAMES berlaku secara rekursif ke objek SUPER apa pun yang bersarang di dalam array SUPER atau di dalam objek lain.

Menggunakan LOWER_ATTRIBUTE_NAMES pada objek SUPER dengan nama atribut duplikat

Jika objek SUPER berisi atribut yang namanya hanya berbeda dalam kasus mereka, LOWER_ATTRIBUTE_NAMES akan menimbulkan kesalahan. Pertimbangkan contoh berikut.

```
SELECT LOWER_ATTRIBUTE_NAMES(JSON_PARSE('{"A": "A", "a": "a"}'));

error:   Invalid input
code:    8001
context: SUPER value has duplicate attributes after case conversion.
```

Fungsi OBJEK

Membuat objek tipe data SUPER.

Sintaksis

```
OBJECT ( [ key1, value1 ], [ key2, value2 ...] )
```

Argumen

kunci1, kunci2

Ekspresi yang mengevaluasi string tipe VARCHAR.

nilai1, nilai2

Ekspresi tipe data Amazon Redshift apa pun kecuali tipe datetime, karena Amazon Redshift tidak mentransmisikan tipe datetime ke tipe data SUPER. Untuk informasi selengkapnya tentang jenis datetime, lihatJenis tanggal-waktu.

`value`ekspresi dalam objek tidak harus dari tipe data yang sama.

Jenis pengembalian

SUPER

Contoh

```
-- Creates an empty object.
select object();

object
```

```
--------
{}
(1 row)

-- Creates objects with different keys and values.
select object('a', 1, 'b', true, 'c', 3.14);

object
---------------------------
{"a":1,"b":true,"c":3.14}
(1 row)

select object('a', object('aa', 1), 'b', array(2,3), 'c', json_parse('{}'));

object
---------------------------------
{"a":{"aa":1},"b":[2,3],"c":{}}
(1 row)

-- Creates objects using columns from a table.
create table bar (k varchar, v super);
insert into bar values ('k1', json_parse('[1]')), ('k2', json_parse('{}'));
select object(k, v) from bar;

object
------------
{"k1":[1]}
{"k2":{}}
(2 rows)

-- Errors out because DATE type values can't be converted to SUPER type.
select object('k', '2008-12-31'::date);

ERROR:  OBJECT could not convert type date to super
```

Fungsi OBJECT_TRANSFORM

Mengubah objek SUPER.

Sintaksis

```
OBJECT_TRANSFORM(
  input
  [KEEP path1, ...]
```

```
  [SET
    path1, value1,
    ...,  ...
  ]
)
```

Argumen

masukan

Ekspresi yang diselesaikan ke objek tipe SUPER.

SIMPAN

Semua nilai jalur yang ditentukan dalam klausa ini disimpan dan dibawa ke objek output.

Klausul ini bersifat opsional.

jalur1, jalur 2,...

Literal string konstan, dalam format komponen jalur dengan tanda kutip ganda yang dibatasi oleh titik. Misalnya, `'"a"."b"."c"'` adalah nilai jalur yang valid. Ini berlaku untuk parameter path di klausa KEEP dan SET.

MENGATUR

jalur dan pasangan nilai untuk memodifikasi jalur keluar atau menambahkan jalur baru, dan mengatur nilai jalur itu di objek keluaran.

Klausul ini bersifat opsional.

nilai1, nilai 2,...

Ekspresi yang menyelesaikan ke nilai tipe SUPER. Perhatikan bahwa tipe numerik, teks, dan Boolean dapat diselesaikan menjadi SUPER.

Jenis pengembalian

SUPER

Catatan penggunaan

OBJECT_TRANSFORM mengembalikan objek tipe SUPER yang berisi nilai jalur dari input yang ditentukan di KEEP dan jalur dan pasangan nilai yang ditentukan dalam SET.

Jika kedua KEEP dan SET kosong, OBJECT_TRANSFORM mengembalikan input.

Jika input bukan objek tipe SUPER, OBJECT_TRANSFORM mengembalikan input, terlepas dari nilai KEEP atau SET.

Contoh

Contoh berikut mengubah objek SUPER menjadi objek SUPER lainnya.

```
CREATE TABLE employees (
    col_person SUPER
);

INSERT INTO employees
VALUES
    (
        json_parse('
            {
                "name": {
                    "first": "John",
                    "last": "Doe"
                },
                "age": 25,
                "ssn": "111-22-3333",
                "company": "Company Inc.",
                "country": "U.S."
            }
        ')
    ),
    (
        json_parse('
            {
                "name": {
                    "first": "Jane",
                    "last": "Appleseed"
                },
                "age": 34,
                "ssn": "444-55-7777",
                "company": "Organization Org.",
                "country": "Ukraine"
            }
        ')
    )
;

SELECT
```

```
    OBJECT_TRANSFORM(
        col_person
        KEEP
            '"name"."first"',
            '"age"',
            '"company"',
            '"country"'
        SET
            '"name"."first"', UPPER(col_person.name.first::TEXT),
            '"age"', col_person.age + 5,
            '"company"', 'Amazon'
    ) AS col_person_transformed
FROM employees;

--This result is formatted for ease of reading.
                  col_person_transformed
---------------------------------------------------------------
{
    "name": {
        "first": "JOHN"
    },
    "age": 30,
    "company": "Amazon",
    "country": "U.S."
}
{
    "name": {
        "first": "JANE"
    },
    "age": 39,
    "company": "Amazon",
    "country": "Ukraine"
}
```

Fungsi UPPER_ATTRIBUTE_NAMES

Mengonversi semua nama atribut yang berlaku dalam nilai SUPER menjadi huruf besar, menggunakan rutinitas konversi huruf yang sama seperti. Fungsi ATAS UPPER_ATTRIBUTE_NAMES mendukung karakter UTF-8 multibyte, hingga maksimal empat byte per karakter.

Untuk mengonversi nama atribut SUPER menjadi huruf kecil, gunakan. Fungsi LOWER_ATTRIBUTE_NAMES

Sintaksis

```
UPPER_ATTRIBUTE_NAMES( super_expression )
```

Argumen

super_ekspresi

Ekspresi SUPER.

Jenis pengembalian

SUPER

Contoh

Mengonversi nama atribut SUPER menjadi huruf besar

Contoh berikut menggunakan UPPER_ATTRIBUTE_NAMES untuk mengonversi nama atribut dari semua nilai SUPER dalam tabel.

```
-- Create a table and insert several SUPER values.
CREATE TABLE t (i INT, s SUPER);

INSERT INTO t VALUES
  (1, NULL),
  (2, 'a'::SUPER),
  (3, JSON_PARSE('{"AttributeName": "b"}')),
  (4, JSON_PARSE(
     '[{"Subobject": {"c": "c"},
       "Subarray": [{"d": "d"}, "e"]
      }]'));

-- Convert all attribute names to uppercase.
UPDATE t SET s = UPPER_ATTRIBUTE_NAMES(s);

SELECT i, s FROM t ORDER BY i;

 i |                         s
---+---------------------------------------------------
 1 | NULL
 2 | "a"
```

```
 3 | {"ATTRIBUTENAME":"B"}
 4 | [{"SUBOBJECT":{"C":"c"},"SUBARRAY":[{"D":"d"}, "e"]}]
```

Amati bagaimana UPPER_ATTRIBUTE_NAMES berfungsi.

- Nilai NULL dan nilai SUPER sk `"a"` alar seperti tidak berubah.
- Dalam objek SUPER, semua nama atribut diubah menjadi huruf besar, tetapi nilai atribut seperti `"b"` tetap tidak berubah.
- UPPER_ATTRIBUTE_NAMES berlaku secara rekursif ke objek SUPER apa pun yang bersarang di dalam array SUPER atau di dalam objek lain.

Menggunakan UPPER_ATTRIBUTE_NAMES pada objek SUPER dengan nama atribut duplikat

Jika objek SUPER berisi atribut yang namanya hanya berbeda dalam kasus mereka, UPPER_ATTRIBUTE_NAMES akan menimbulkan kesalahan. Pertimbangkan contoh berikut.

```
SELECT UPPER_ATTRIBUTE_NAMES(JSON_PARSE('{"A": "A", "a": "a"}'));

error:   Invalid input
code:    8001
context: SUPER value has duplicate attributes after case conversion.
```

Fungsi spasial

Hubungan antara objek geometri didasarkan pada Model Sembilan Persimpangan Dimensi yang Diperpanjang (). DE-9IM Model ini mendefinisikan predikat seperti sama, mengandung, dan menutupi. Untuk informasi lebih lanjut tentang definisi hubungan spasial, lihat DE-9IM di Wikipedia.

Untuk informasi selengkapnya tentang cara menggunakan data spasial dengan Amazon Redshift, lihatMengkueri data spasial di Amazon Redshift.

Amazon Redshift menyediakan fungsi spasial yang berfungsi dengan `GEOMETRY` dan tipe `GEOGRAPHY` data. Berikut ini mencantumkan fungsi yang mendukung tipe `GEOGRAPHY` data:

- ST_area
- ST_Asewkt
- ST_JSON AsGeo
- ST_EWKB AsHex

- ST_WKB AsHex
- ST_ AsText
- ST_Jarak
- ST_ GeogFromText
- ST_WKB GeogFrom
- ST_panjang
- ST_NPoin
- ST_Perimeter

Berikut ini mencantumkan rangkaian lengkap fungsi spasial yang didukung oleh Amazon Redshift.

Topik

- AddBbox
- DropbBox
- GeometryType
- H3_Batas
- H3_Pusat
- H3_ FromLongLat
- H3_ FromPoint
- H3_ IsValid
- H3_Polyfill
- Resolusi H3_
- H3_ ToChildren
- H3_ ToParent
- ST_ AddPoint
- ST_Angle
- ST_area
- ST_ AsBinary
- ST_AsewKB
- ST_Asewkt
- ST_JSON AsGeo

- ST_WKB AsHex
- ST_EWKB AsHex
- ST_ AsText
- ST_Azimut
- ST_Batas
- ST_Buffer
- ST_Centroid
- ST_Collect
- ST_berisi
- ST_ ContainsProperly
- ST_ ConvexHull
- ST_ CoveredBy
- ST_Cover
- ST_cross
- st_Dimensi
- ST_disjoint
- ST_Jarak
- ST_ DistanceSphere
- ST_Dwithin
- ST_ EndPoint
- ST_Amplop
- ST_sama
- ST_ ExteriorRing
- ST_Force2D
- ST_Force3d
- ST_Force3DM
- ST_Force3DZ
- ST_Force4d
- ST_ GeoHash
- ST_ GeogFromText

- ST_WKB GeogFrom
- ST_Geometryn
- ST_ GeometryType
- ST_EWKB GeomFrom
- ST_EWKT GeomFrom
- ST_ GeomFromGeoHash
- ST_ GeomFromGeoJSON
- ST_ GeomFromGeoSquare
- ST_ GeomFromText
- ST_WKB GeomFrom
- ST_ GeoSquare
- ST_N InteriorRing
- ST_berpotongan
- ST_Persimpangan
- ST_CCW IsPolygon
- ST_CW IsPolygon
- ST_ IsClosed
- ST_ IsCollection
- ST_ IsEmpty
- ST_ IsRing
- ST_ IsSimple
- ST_ IsValid
- ST_panjang
- ST_ LengthSphere
- ST_panjang2d
- ST_ LineFromMultiPoint
- ST_ LineInterpolatePoint
- ST_M
- ST_ MakeEnvelope
- ST_ MakeLine

- ST_ MakePoint
- ST_ MakePolygon
- ST_ MemSize
- ST_Mmax
- St_mmin
- ST_multi
- ST_NDIM
- ST_NPoin
- ST_NRings
- ST_ NumGeometries
- ST_ NumInteriorRings
- ST_ NumPoints
- ST_Perimeter
- ST_perimeter2d
- Titik ST
- st_pointn
- ST_poin
- ST_poligon
- ST_ RemovePoint
- ST_terbalik
- ST_ SetPoint
- ST_SetsRid
- ST_Sederhanakan
- ST_SRID
- ST_ StartPoint
- ST_sentuhan
- ST_Transform
- ST_Union
- ST_dalam
- ST_X

- ST_xmax
- st_xmin
- ST_Y
- ST_ymax
- st_ymin
- ST_Z
- ST_Zmax
- St_zmin
- DukunganBBox

AddBbox

AddbBox mengembalikan salinan geometri input yang mendukung pengkodean dengan kotak pembatas yang telah dihitung sebelumnya. Untuk informasi selengkapnya tentang dukungan untuk kotak pembatas, lihatKotak pembatas.

Sintaksis

```
AddBBox(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan salinan geometri poligon input yang mendukung dikodekan dengan kotak pembatas.

```
SELECT ST_AsText(AddBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
 st_astext
----------
 POLYGON((0 0,1 0,0 1,0 0))
```

DropbBox

DropbBox mengembalikan salinan geometri input yang tidak mendukung pengkodean dengan kotak pembatas yang telah dihitung sebelumnya. Untuk informasi selengkapnya tentang dukungan untuk kotak pembatas, lihatKotak pembatas.

Sintaksis

```
DropBBox(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan salinan geometri poligon input yang tidak mendukung dikodekan dengan kotak pembatas.

```
SELECT ST_AsText(DropBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
 st_astext
----------
 POLYGON((0 0,1 0,0 1,0 0))
```

GeometryType

GeometryType mengembalikan subtipe geometri input sebagai string.

Sintaksis

```
GeometryType(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

VARCHARmewakili subtipe ge om.

Jika geom adalah null, maka null dikembalikan.

Nilai yang dikembalikan adalah sebagai berikut.

Nilai string yang dikembalikan untuk geometri 2D, 3DZ, 4D	Nilai string yang dikembalikan untuk geometri 3DM	Subtipe geometri
POINT	POINTM	Dikembalikan jika geom adalah subtipe POINT
LINESTRING	LINESTRINGM	Dikembalikan jika geom adalah subtipe LINESTRING
POLYGON	POLYGONM	Dikembalikan jika geom adalah subtipe POLYGON
MULTIPOINT	MULTIPOINTM	Dikembalikan jika geom adalah subtipe MULTIPOINT
MULTILINESTRING	MULTILINESTRINGM	Dikembalikan jika geom adalah subtipe MULTILINE STRING

Nilai string yang dikembalikan untuk geometri 2D, 3DZ, 4D	Nilai string yang dikembalikan untuk geometri 3DM	Subtipe geometri
MULTIPOLYGON	MULTIPOLYGONM	Dikembalikan jika geom adalah subtipe MULTIPOLYGON
GEOMETRYCOLLECTION	GEOMETRYCOLLECTIONM	Dikembalikan jika geom adalah subtipe GEOMETRYCOLLECTION

Contoh

SQL berikut mengonversi representasi teks terkenal (WKT) dari poligon dan mengembalikan GEOMETRY subtipe sebagai string.

```
SELECT GeometryType(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
geometrytype
-------------
 POLYGON
```

H3_Batas

H3_Boundary mengembalikan batas ID sel H3 dari indeks input. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_Boundary(index)
```

Argumen

indeks

Nilai tipe data BIGINT atau VARCHAR yang mewakili indeks sel H3. Atau, ekspresi yang mengevaluasi salah satu tipe data ini.

Jenis pengembalian

POLYGON— mewakili poligon dengan pengidentifikasi sistem referensi spasial (SRID) dari. 0

Jika indeks tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut input a VARCHAR yang mewakili indeks sel H3, dan mengembalikan POLYGON dengan SRID 0 yang mewakili batas input sel H3. Output dari H3_Boundary adalah input ke ST_Awewkt untuk ditampilkan dalam representasi teks terkenal yang diperluas (EWKT).

```
SELECT ST_AsEWKT(H3_Boundary('8025fffffffffff'));
```

```
 st_asewkt
-----------------------------------------------------------------------------------------------------
 POLYGON((87.7729641223953 52.6542030078627,86.7082098104731
 40.3127667561966,98.2285042557705 33.6210697806835,110.694610548823
 37.163896485796,116.212895637138 47.3094513028131,106.40349563788
 56.210610737585,87.7729641223953 52.6542030078627))
```

SQL berikut input a BIGINT yang mewakili indeks sel H3, dan mengembalikan POLYGON dengan SRID 0 yang mewakili batas input sel H3. Output dari H3_Boundary adalah input ke ST_Awewkt untuk ditampilkan dalam representasi teks terkenal yang diperluas (EWKT).

```
SELECT ST_AsEWKT(H3_Boundary(577129255373111295));
```

```
 st_asewkt
-----------------------------------------------------------------------------------------------------
POLYGON((87.7729641223953 52.6542030078627,86.7082098104731
 40.3127667561966,98.2285042557705 33.6210697806835,110.694610548823
 37.163896485796,116.212895637138 47.3094513028131,106.40349563788
 56.210610737585,87.7729641223953 52.6542030078627))
```

H3_Pusat

H3_Center mengembalikan sentroid dari ID sel H3 dari indeks input. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_Center(index)
```

Argumen

indeks

Nilai tipe data BIGINT atau VARCHAR yang mewakili indeks sel H3. Atau, ekspresi yang mengevaluasi salah satu tipe data ini.

Jenis pengembalian

POINT— mewakili sentroid sel H3 dengan pengidentifikasi sistem referensi spasial (SRID) dari. 0

Jika indeks tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut input a VARCHAR yang mewakili indeks sel H3, dan mengembalikan POINT dengan SRID 0 yang mewakili sentroid dari input sel H3.

```
SELECT H3_Center('8025fffffffffff');
```

```
 h3_center
--------------------------------------------
 010100000070707A550B605940AEE9D70B327E4640
```

SQL berikut input a BIGINT yang mewakili indeks sel H3, dan mengembalikan POINT dengan SRID 0 yang mewakili sentroid dari input sel H3.

```
SELECT H3_Center(577129255373111295);
```

```
 h3_center
--------------------------------------------
 010100000070707A550B605940AEE9D70B327E4640
```

SQL berikut input a VARCHAR yang mewakili indeks sel H3, dan mengembalikan POINT dengan SRID 0 yang mewakili sentroid dari input sel H3. Output dari H3_Center adalah input ke ST_Awewkt untuk ditampilkan dalam representasi teks terkenal yang diperluas (EWKT).

```
SELECT ST_AsEWKT(H3_Center('8075fffffffffff'));
```

```
 st_asewkt
-----------------------------------------
POINT(-5.24539029677733 2.30088211162675)
```

H3_ FromLongLat

H3_ FromLongLat mengembalikan ID sel H3 yang sesuai dari masukan bujur, lintang, dan resolusi. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_FromLongLat(longitude, latitude, resolution)
```

Argumen

bujur

Nilai tipe data DOUBLE PRECISION atau ekspresi yang mengevaluasi ke DOUBLE PRECISION tipe.

garis lintang

Nilai tipe data DOUBLE PRECISION atau ekspresi yang mengevaluasi ke DOUBLE PRECISION tipe.

resolusi

Nilai tipe data INTEGER atau ekspresi yang mengevaluasi ke INTEGER tipe. Nilai tersebut mewakili resolusi sistem grid H3. Nilai harus berupa bilangan bulat antara 0-15, inklusif. Dengan 0 menjadi yang paling kasar dan 15 menjadi yang terbaik.

Jenis pengembalian

BIGINT— mewakili ID sel H3.

Jika resolusi di luar batas, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan ID sel H3 dari bujur`0`, lintang`0`, dan resolusi`10`.

```
SELECT H3_FromLongLat(0, 0, 10);
```

```
 h3_fromlonglat
------------------
 623560421467684863
```

H3_ FromPoint

H3_ FromPoint mengembalikan ID sel H3 yang sesuai dari titik geometri input dan resolusi. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_FromPoint(geom, resolution)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe. Geom harus a`POINT`.

resolusi

Nilai tipe data `INTEGER` atau ekspresi yang mengevaluasi ke `INTEGER` tipe. Nilai tersebut mewakili resolusi sistem grid H3. Nilai harus berupa bilangan bulat antara 0-15, inklusif. Dengan `0` menjadi yang paling kasar dan 15 menjadi yang terbaik.

Jenis pengembalian

`BIGINT`— mewakili ID sel H3.

Jika geom bukan a`POINT`, maka kesalahan dikembalikan.

Jika resolusi di luar batas, maka kesalahan dikembalikan.

Jika geom kosong, maka NULL dikembalikan.

Contoh

SQL berikut mengembalikan ID sel H3 dari titik`0,0`, dan resolusi`10`.

```
SELECT H3_FromPoint(ST_GeomFromText('POINT(0 0)'), 10);
```

```
 h3_frompoint
--------------------
 623560421467684863
```

H3_ IsValid

H3_ IsValid mengembalikan true jika input mewakili ID sel H3, jika tidak salah. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_IsValid(index)
```

Argumen

indeks

Nilai tipe data `BIGINT` atau`VARCHAR`, atau ekspresi yang mengevaluasi salah satu tipe data ini.

Jenis pengembalian

`BOOLEAN`— benar jika input mewakili ID sel H3 yang valid, false sebaliknya.

Jika indeks adalah NULL, maka NULL dikembalikan.

Contoh

SQL berikut memasukkan VARCHAR yang mewakili ID sel H3, dan mengembalikan true.

```
SELECT H3_IsValid('8025fffffffffff');
```

```
 h3_isvalid
------------
 true
```

SQL berikut memasukkan BIGINT yang mewakili ID sel H3, dan mengembalikan true.

```
SELECT H3_IsValid(577129255373111295);
```

```
 h3_isvalid
------------
 true
```

SQL berikut memasukkan ID sel H3 yang tidak valid dan mengembalikan false.

```
SELECT H3_IsValid('');
```

```
 h3_isvalid
------------
 false
```

H3_Polyfill

H3_Polyfill mengembalikan ID sel H3 yang sesuai yang sesuai dengan segi enam dan pentagon yang terkandung dalam poligon input dari resolusi yang diberikan. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_Polyfill(geom, resolution)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Geom harus aPOLYGON.

resolusi

Nilai tipe data INTEGER atau ekspresi yang mengevaluasi ke INTEGER tipe. Nilai tersebut mewakili resolusi sistem grid H3. Nilai harus berupa bilangan bulat antara 0-15, inklusif. Dengan 0 menjadi yang paling kasar dan 15 menjadi yang terbaik.

Jenis pengembalian

SUPER— mewakili daftar ID sel H3.

Jika geom bukan aPOLYGON, maka kesalahan dikembalikan.

Jika resolusi di luar batas, maka kesalahan dikembalikan.

Jika geom kosong, maka NULL dikembalikan.

Contoh

SQL berikut mengembalikan array tipe data SUPER ID sel H3 dari poligon dan resolusi. 4

```
SELECT H3_Polyfill(ST_GeomFromText('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))'), 4);
```

```
 h3_polyfill
-------------------------------------------------------------------------------------------------
 [596538848238895103,596538805289222143,596538856828829695,596538813879156735,5965379205259516
```

Resolusi H3_

H3_Resolution mengembalikan resolusi ID sel H3 dari indeks input. Resolusi adalah bilangan bulat dari 0 (paling kasar) hingga 15 (terbaik). Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_Resolution(index)
```

Argumen

indeks

Nilai tipe data BIGINT atau VARCHAR yang mewakili indeks sel H3, atau ekspresi yang mengevaluasi salah satu tipe data ini.

Jenis pengembalian

`INTEGER`— mewakili resolusi input ID sel H3.

Jika indeks adalah NULL, maka NULL dikembalikan.

Jika indeks tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut memasukkan VARCHAR yang mewakili indeks sel H3, dan mengembalikan INTEGER yang mewakili resolusi input sel H3.

```
SELECT H3_Resolution('8025fffffffffff');
```

```
 h3_resolution
---------------
 0
```

SQL berikut memasukkan BIGINT yang mewakili indeks sel H3, dan mengembalikan INTEGER yang mewakili resolusi input sel H3.

```
SELECT H3_Resolution(614553222213795839);
```

```
 h3_resolution
---------------
 8
```

H3_ ToChildren

H3_ ToChildren mengembalikan daftar ID sel H3 anak pada resolusi tertentu untuk indeks H3 tertentu. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_ToChildren(index, resolution)
```

Argumen

indeks

Nilai tipe data BIGINT atau VARCHAR yang mewakili indeks sel H3, atau ekspresi yang mengevaluasi salah satu tipe data ini.

resolusi

Nilai tipe data INTEGER atau ekspresi yang mengevaluasi ke INTEGER tipe. Nilai tersebut mewakili resolusi ID sel anak. Nilai harus berupa bilangan bulat antara resolusi indeks input dan 15, inklusif.

Jenis pengembalian

SUPER— mewakili daftar ID sel H3.

Jika indeks atau resolusi adalah NULL, maka NULL dikembalikan.

Jika indeks tidak valid, maka kesalahan dikembalikan.

Jika resolusi tidak antara resolusi indeks dan 15, inklusif, maka kesalahan dikembalikan.

Jika ukuran output melebihi batas ukuran SUPER maksimum, maka kesalahan dikembalikan.

Contoh

SQL berikut memasukkan VARCHAR yang mewakili indeks sel H3, dan INTEGER yang mewakili resolusi yang diinginkan dari semua anak, dan mengembalikan larik SUPER yang berisi anak-anak pada resolusi 6.

```
SELECT H3_ToChildren('85283473fffffff', 6);
```

```
 h3_tochildren
---------------------------------------------------------------------------------------------
 [604189641121202175,604189641255419903,604189641389637631,604189641523855359,6041896416580730
```

SQL berikut memasukkan BIGINT yang mewakili indeks sel H3, dan INTEGER yang mewakili resolusi yang diinginkan dari semua anak, dan mengembalikan larik SUPER yang berisi anak-anak pada resolusi 6.

```
SELECT H3_ToChildren(599686042433355775, 6);
```

```
 h3_tochildren
---------------------------------------------------------------------------------------------
 [604189641121202175,604189641255419903,604189641389637631,604189641523855359,6041896416580730
```

Catatan: Perbedaan 7 atau kurang antara resolusi dan resolusi indeks aman.

Contoh berikut menunjukkan solusi untuk kueri yang melebihi batas ukuran SUPER. Ketika perbedaan resolusi antara indeks H3 input dan resolusi anak yang diinginkan terlalu besar (lebih besar dari 7). Prosedur ini memecahkan masalah dengan memperluas anak secara bertahap dalam langkah-langkah yang lebih kecil (maksimum 5 tingkat resolusi sekaligus) dan menyimpan hasil akhir dalam tabel yang dibuat pengguna.

```
CREATE OR REPLACE PROCEDURE generate_h3_children()
LANGUAGE plpgsql
AS $$
BEGIN
    -- Drop and create h3_children table that will contain the results
    DROP TABLE IF EXISTS h3_children;
    CREATE TABLE h3_children (
        h3_index BIGINT,
        child_res INTEGER,
        children SUPER
    );

    -- Create temporary table for steps
    DROP TABLE IF EXISTS h3_steps;
    CREATE TABLE h3_steps (
        h3_index BIGINT,
        current_res INTEGER,
        target_res INTEGER,
        h3_array SUPER
    );

    -- Initial insert into h3_steps
    INSERT INTO h3_steps
    SELECT h3_index, H3_Resolution(h3_index), child_res, ARRAY(h3_index)
    FROM h3_indexes; -- Insert from your table with h3_index and child_res as columns

    -- Loop until we reach target resolution
```

```
    -- We expect at most 3 iterations considering that we can start at resolution
    -- 0 and target/child resolution equal to 15 (0 -> 5 -> 10 -> 15)
    WHILE EXISTS (
        SELECT 1
        FROM h3_steps h
        GROUP BY h3_index, target_res
        HAVING MAX(current_res) < target_res
    ) LOOP
        -- Populate the h3_steps table with the tables that need to
        -- reach closer to the target res
        INSERT INTO h3_steps
        SELECT
            h.h3_index,
            LEAST(h.current_res + 5, h.target_res), -- Do not exceed target res
            h.target_res,
            -- Take the children of the child cell at resolution current_res of the
            -- h3_index
            H3_ToChildren(c.child::BIGINT, LEAST(h.current_res + 5, h.target_res))
        FROM h3_steps h, UNNEST(h.h3_array) AS c(child)
        WHERE h.current_res < h.target_res
        AND h.current_res = (SELECT MAX(current_res)
                             FROM h3_steps
                             WHERE h3_index = h.h3_index
        );
    END LOOP;

    -- Store final results
    INSERT INTO h3_children
    SELECT h3_index AS h3_index, target_res AS child_res, h3_array AS children
    FROM h3_steps
    WHERE current_res = target_res;
END;
$$;

-- Create the source table for H3_ToChildren queries
CREATE TABLE h3_indexes (
    h3_index BIGINT,
    child_res INTEGER,
    PRIMARY KEY (h3_index, child_res)
);
INSERT INTO h3_indexes (h3_index, child_res)
VALUES (x'8001fffffffffff'::BIGINT, 11);

-- Execute the procedure
```

```
CALL generate_h3_children();

-- View results
SELECT * FROM h3_children;
```

H3_ ToParent

H3_ ToParent mengembalikan ID sel H3 induk pada resolusi induk tertentu untuk indeks H3 tertentu. Untuk informasi tentang pengindeksan H3, lihat. H3

Sintaksis

```
H3_ToParent(index, resolution)
```

Argumen

indeks

Nilai tipe data BIGINT atau VARCHAR yang mewakili indeks sel H3, atau ekspresi yang mengevaluasi salah satu tipe data ini.

resolusi

Nilai tipe data INTEGER atau ekspresi yang mengevaluasi ke INTEGER tipe. Nilai tersebut mewakili resolusi ID sel induk. Nilai harus antara 0 dan resolusi indeks, inklusif.

Jenis pengembalian

BIGINT— mewakili ID sel H3 induk.

Jika indeks atau resolusi adalah NULL, maka NULL dikembalikan.

Jika indeks tidak valid, maka kesalahan dikembalikan.

Jika resolusi kurang dari 0 atau lebih besar dari resolusi indeks, maka kesalahan dikembalikan.

Contoh

SQL berikut memasukkan VARCHAR yang mewakili indeks sel H3, dan INTEGER yang mewakili resolusi yang diinginkan dari induk yang diinginkan, dan mengembalikan BIGINT yang mewakili induk pada resolusi 0 dari sel H3 input.

```
SELECT H3_ToParent('85283473fffffff', 0);
```

```
 h3_toparent
--------------------
 577199624117288959
```

SQL berikut memasukkan BIGINT yang mewakili indeks sel H3, dan INTEGER yang mewakili resolusi yang diinginkan dari induk yang diinginkan, dan mengembalikan BIGINT yang mewakili induk pada resolusi 0 dari sel H3 input.

```
SELECT H3_ToParent(646078419604526808, 8);
```

```
 h3_toparent
--------------------
 614553222213795839
```

ST_ AddPoint

ST_ AddPoint mengembalikan geometri linestring yang sama dengan geometri input dengan titik ditambahkan. Jika indeks disediakan, maka titik ditambahkan pada posisi indeks. Jika indeksnya -1 atau tidak disediakan, maka titik tersebut ditambahkan ke linestring.

Indeks ini berbasis nol. Pengidentifikasi sistem referensi spasial (SRID) dari hasilnya sama dengan geometri input.

Dimensi geometri yang dikembalikan sama dengan nilai geom1. Jika geom1 dan geom2 memiliki dimensi yang berbeda, geom2 diproyeksikan ke dimensi geom 1.

Sintaksis

```
ST_AddPoint(geom1, geom2)
```

```
ST_AddPoint(geom1, geom2, index)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus POINT Poinnya bisa menjadi titik kosong.

indeks

Nilai tipe data INTEGER yang mewakili posisi indeks berbasis nol.

Jenis pengembalian

GEOMETRY

Jika geom1, geom2, atau indeks adalah null, maka null dikembalikan.

Jika geom2 adalah titik kosong, maka salinan geom 1 dikembalikan.

Jika geom1 bukan aLINESTRING, maka kesalahan dikembalikan.

Jika geom2 bukan aPOINT, maka kesalahan dikembalikan.

Jika indeks berada di luar jangkauan, maka kesalahan dikembalikan. Nilai yang valid untuk posisi indeks adalah -1 atau nilai antara 0 danST_NumPoints(geom1).

Contoh

SQL berikut menambahkan titik ke linestring untuk membuatnya menjadi linestring tertutup.

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_AddPoint(g, ST_StartPoint(g))) FROM tmp;
```

```
 st_asewkt
-------------------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 5,0 5,0 0)
```

SQL berikut menambahkan titik ke posisi tertentu dalam linestring.

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_AddPoint(g, ST_SetSRID(ST_Point(5, 10), 4326), 3)) FROM tmp;
```

```
 st_asewkt
-----------------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 10,5 5,0 5)
```

ST_Angle

ST_Angle mengembalikan sudut dalam radian antara titik-titik yang diukur searah jarum jam sebagai berikut:

- Jika tiga titik dimasukkan, maka sudut yang dikembalikan P1-P2-P3 diukur seolah-olah sudut diperoleh dengan memutar dari P1 ke P3 sekitar P2 searah jarum jam.
- Jika empat titik dimasukkan, maka sudut searah jarum jam yang dikembalikan dibentuk oleh garis yang diarahkan P1-P2 dan P3-P4 dikembalikan. Jika input adalah kasus degenerasi (yaitu, P1 sama dengan P2, atau P3 sama dengan P4), maka null dikembalikan.

Nilai pengembalian dalam radian dan dalam kisaran [0, 2π).

ST_Angle beroperasi pada proyeksi 2D dari geometri input.

Sintaksis

```
ST_Angle(geom1, geom2, geom3)
```

```
ST_Angle(geom1, geom2, geom3, geom4)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus POINT

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus POINT

geom3

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus POINT

geom4

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus POINT

Jenis pengembalian

DOUBLE PRECISION.

Jika geom1 sama dengan geom2, atau geom2 sama dengan geom3, maka null dikembalikan.

Jika geom1, geom2, geom 3, atau geom4 adalah null, maka null dikembalikan.

Jika salah satu dari geom1, geom2, geom 3, atau geom4 adalah titik kosong, maka kesalahan dikembalikan.

Jika geom1, geom2, geom 3 , dan geom4 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan sudut yang dikonversi ke derajat tiga titik input.

```
SELECT ST_Angle(ST_Point(1,1), ST_Point(0,0), ST_Point(1,0)) / Pi() * 180.0 AS angle;
```

```
 angle
---------------
    45
```

SQL berikut mengembalikan sudut yang dikonversi ke derajat empat titik input.

```
SELECT ST_Angle(ST_Point(1,1), ST_Point(0,0), ST_Point(1,0), ST_Point(2,0)) / Pi() *
 180.0 AS angle;
```

```
 angle
---------------
   225
```

ST_area

Untuk geometri input, ST_Area mengembalikan area Cartesian dari proyeksi 2D. Satuan luas sama dengan satuan di mana koordinat geometri input dinyatakan. Untuk titik, linestring, multipoint, dan multilinestring, fungsi mengembalikan 0. Untuk koleksi geometri, ia mengembalikan jumlah area geometri dalam koleksi.

Untuk geografi masukan, ST_Area mengembalikan area geodesik dari proyeksi 2D dari geografi areal input yang dihitung pada spheroid yang ditentukan oleh SRID. Satuan panjangnya dalam meter persegi. Fungsi mengembalikan nol (0) untuk titik, multipoint, dan geografi linier. Ketika input adalah kumpulan geometri, fungsi mengembalikan jumlah area geografi areal dalam koleksi.

Sintaksis

```
ST_Area(geo)
```

Argumen

geografis

Nilai tipe data `GEOMETRY` atau`GEOGRAPHY`, atau ekspresi yang mengevaluasi ke `GEOGRAPHY` tipe `GEOMETRY` atau.

Jenis pengembalian

`DOUBLE PRECISION`

Jika geo adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan area Cartesian dari multipoligon.

```
SELECT ST_Area(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20 10,10
 0)))'));
```

```
 st_area
---------
     100
```

SQL berikut mengembalikan area poligon dalam geografi.

```
SELECT ST_Area(ST_GeogFromText('polygon((34 35, 28 30, 25 34, 34 35))'));
```

```
     st_area
------------------
 201824655743.383
```

SQL berikut mengembalikan nol untuk geografi linier.

```
SELECT ST_Area(ST_GeogFromText('multipoint(0 0, 1 1, -21.32 121.2)'));
```

```
 st_area
---------
       0
```

ST_ AsBinary

ST_ AsBinary mengembalikan representasi biner terkenal heksadesimal (WKB) dari geometri input. Untuk geometri 3DZ, 3DM, dan 4D, ST_ AsBinary menggunakan nilai standar Open Geospatial Consortium (OGC) untuk tipe geometri.

Sintaksis

```
ST_AsBinary(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

VARBYTE

Jika *geom* adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan representasi WKB heksadesimal dari poligon.

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_asbinary
--------------------------------
0103000000010000000500000000000000000000000000000000000000000000000000F03F00000
```

ST_AsewKB

ST_ASEWKB mengembalikan representasi biner terkenal yang diperluas (EWKB) dari geometri input. Untuk geometri 3DZ, 3DM, dan 4D, ST_ASEWKB menggunakan nilai standar Open Geospatial Consortium (OGC) untuk tipe geometri.

Sintaksis

```
ST_AsEWKB(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

VARBYTE

Jika *geom* adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan representasi EWKB heksadesimal dari poligon.

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_asewkb
--------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000F0
```

ST_Asewkt

ST_ASewkt mengembalikan representasi teks terkenal yang diperluas (EWKT) dari geometri masukan atau geografi. Untuk geometri 3DZ, 3DM, dan 4D, ST_ASewkt menambahkan Z, M, atau ZM ke nilai WKT untuk tipe geometri.

Sintaksis

```
ST_AsEWKT(geo)
```

```
ST_AsEWKT(geo, precision)
```

Argumen

geografis

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

presisi

Nilai tipe dataINTEGER. Untuk geometri, koordinat geo ditampilkan menggunakan presisi yang ditentukan 1—20. Jika presisi tidak ditentukan, defaultnya adalah 15. Untuk geografi, koordinat geo ditampilkan menggunakan presisi yang ditentukan. Jika presisi tidak ditentukan, defaultnya adalah 15.

Jenis pengembalian

VARCHAR

Jika geo adalah null, maka null dikembalikan.

Jika pres isi nol, maka null dikembalikan.

Jika hasilnya lebih besar dari 64-KBVARCHAR, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan representasi EWKT dari linestring.

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326));
```

```
st_asewkt
------------------------------
 SRID=4326;LINESTRING(3.14159265358979 -6.28318530717959,2.71828182845905
 -1.41421356237309)
```

SQL berikut mengembalikan representasi EWKT dari linestring. Koordinat geometri ditampilkan dengan enam digit presisi.

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326), 6);
```

```
st_asewkt
------------------------------
 SRID=4326;LINESTRING(3.14159 -6.28319,2.71828 -1.41421)
```

SQL berikut mengembalikan representasi EWKT dari geografi.

```
SELECT ST_AsEWKT(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
                 st_asewkt
---------------------------------------------
 SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

ST_JSON AsGeo

ST_ AsGeo JSON mengembalikan representasi GeoJSON dari geometri masukan atau geografi. Untuk informasi lebih lanjut tentang GeoJSON, lihat GeoJSON di Wikipedia.

Untuk geometri 3DZ dan 4D, geometri keluaran adalah proyeksi 3DZ dari input geometri 3DZ atau 4D. Artinya, z koordinat xy,, dan hadir dalam output. Untuk geometri 3DM, geometri keluaran adalah proyeksi 2D dari input geometri 3DM. Artinya, hanya x dan y koordinat hadir dalam output.

Untuk masukan geografi, ST_ AsGeo JSON mengembalikan representasi GeoJSON dari geografi masukan. Koordinat geografi ditampilkan menggunakan presisi yang ditentukan.

Sintaksis

```
ST_AsGeoJSON(geo)
```

```
ST_AsGeoJSON(geo, precision)
```

Argumen

geo

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

presisi

Nilai tipe dataINTEGER. Untuk geometri, koordinat geo ditampilkan menggunakan presisi yang ditentukan 1—20. Jika presisi tidak ditentukan, defaultnya adalah 15. Untuk geografi, koordinat geo ditampilkan menggunakan presisi yang ditentukan. Jika presisi tidak ditentukan, defaultnya adalah 15.

Jenis pengembalian

VARCHAR

Jika geo adalah null, maka null dikembalikan.

Jika pres isi nol, maka null dikembalikan.

Jika hasilnya lebih besar dari 64-KBVARCHAR, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan representasi GeoJSON dari linestring.

```
SELECT ST_AsGeoJSON(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)'));
```

```
st_asgeojson
-------------------------------------------------------------------------------------------------
 {"type":"LineString","coordinates":[[3.14159265358979,-6.28318530717959],
[2.71828182845905,-1.414213562373095]]}
```

SQL berikut mengembalikan representasi GeoJSON dari linestring. Koordinat geometri ditampilkan dengan enam digit presisi.

```
SELECT ST_AsGeoJSON(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)'), 6);
```

```
st_asgeojson
-------------------------------------------------------------------------------
  {"type":"LineString","coordinates":[[3.14159,-6.28319],[2.71828,-1.41421]]}
```

SQL berikut mengembalikan representasi GeoJSON dari suatu geografi.

```
SELECT ST_AsGeoJSON(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
                            st_asgeojson
-------------------------------------------------------------------------
 {"type":"LineString","coordinates":[[110,40],[2,3],[-10,80],[-7,9]]}
```

ST_WKB AsHex

ST_ AsHex WKB mengembalikan representasi biner terkenal heksadesimal (WKB) dari geometri masukan atau geografi menggunakan karakter heksadesimal ASCII (0—9, A—F). Untuk geometri atau geografi 3DZ, 3DM, dan 4D, ST_ AsHex WKB menggunakan nilai standar Open Geospatial Consortium (OGC) untuk jenis geometri atau geografi.

Sintaksis

```
ST_AsHexWKB(geo)
```

Argumen

geo

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

Jenis pengembalian

VARCHAR

Jika geo adalah null, maka null dikembalikan.

Jika hasilnya lebih besar dari 64-KBVARCHAR, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan representasi WKB heksadesimal dari poligon dalam geometri.

```
SELECT ST_AsHexWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_ashexwkb
--------------------------------
0103000000010000000500000000000000000000000000000000000000000000000000F03F00000
```

SQL berikut mengembalikan representasi WKB heksadesimal dari poligon dalam geografi.

```
SELECT ST_AsHexWKB(ST_GeogFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'));
```

```
st_ashexwkb
--------------------------------
0103000000010000000500000000000000000000000000000000000000000000000000F03F00000
```

ST_EWKB AsHex

ST_ AsHex EWKB mengembalikan representasi biner terkenal yang diperluas (EWKB) dari geometri masukan atau geografi menggunakan karakter heksadesimal ASCII (0—9, A—F). Untuk geometri

atau geografi 3DZ, 3DM, dan 4D, ST_ AsHex EWKB menggunakan nilai WKB yang diperluas PostGIS untuk jenis geometri atau geografi.

Sintaksis

```
ST_AsHexEWKB(geo)
```

Argumen

geo

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

Jenis pengembalian

VARCHAR

Jika geo adalah null, maka null dikembalikan.

Jika hasilnya lebih besar dari 64-KBVARCHAR, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan representasi EWKB heksadesimal dari poligon dalam geometri.

```
SELECT ST_AsHexEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_ashexewkb
--------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000F03F000000000000F03F000000000000F03F000000000000F03F0000000000000000000000000000000000000000000000F
```

SQL berikut mengembalikan representasi EWKB heksadesimal dari poligon dalam geografi.

```
SELECT ST_AsHexEWKB(ST_GeogFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'));
```

```
st_ashexewkb
--------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000F03F000000000000F03F000000000000F03F000000000000F03F0000000000000000000000000000000000000000000000F
```

ST_ AsText

ST_ AsText mengembalikan representasi teks terkenal (WKT) dari geometri masukan atau geografi. Untuk geometri atau geografi 3DZ, 3DM, dan 4D, ST_ASewkt menambahkan Z, M, atau ZM ke nilai WKT untuk jenis geometri atau geografi.

Sintaksis

```
ST_AsText(geo)
```

```
ST_AsText(geo, precision)
```

Argumen

geo

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

presisi

Nilai tipe dataINTEGER. Untuk geometri, koordinat geo ditampilkan menggunakan presisi yang ditentukan 1—20. Jika presisi tidak ditentukan, defaultnya adalah 15. Untuk geogpraphies, koordinat geo ditampilkan menggunakan pres isi yang ditentukan. Jika presisi tidak ditentukan, defaultnya adalah 15.

Jenis pengembalian

VARCHAR

Jika geo adalah null, maka null dikembalikan.

Jika pres isi nol, maka null dikembalikan.

Jika hasilnya lebih besar dari 64-KBVARCHAR, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan representasi WKT dari linestring.

```
SELECT ST_AsText(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326));
```

```
st_astext
-------------------------------
LINESTRING(3.14159265358979 -6.28318530717959,2.71828182845905 -1.41421356237309)
```

SQL berikut mengembalikan representasi WKT dari linestring. Koordinat geometri ditampilkan dengan enam digit presisi.

```
SELECT ST_AsText(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326), 6);
```

```
st_astext
---------------------------------------------
 LINESTRING(3.14159 -6.28319,2.71828 -1.41421)
```

SQL berikut mengembalikan representasi WKT dari geografi.

```
SELECT ST_AsText(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
             st_astext
-----------------------------------
 LINESTRING(110 40,2 3,-10 80,-7 9)
```

ST_Azimut

ST_Azimuth mengembalikan azimuth Cartesian berbasis utara menggunakan proyeksi 2D dari dua titik input.

Sintaksis

```
ST_Azimuth(point1, point2)
```

Argumen

poin1

POINTNilai tipe dataGEOMETRY. Pengidentifikasi sistem referensi spasial (SRID) dari point1 harus cocok dengan SRID dari point2.

poin2

POINTNilai tipe dataGEOMETRY. SRID point2 harus sesuai dengan SRID point1.

Jenis pengembalian

Angka yang merupakan sudut dalam radian tipe DOUBLE PRECISION data. Nilai berkisar dari 0 (inklusif) hingga 2 pi (eksklusif).

Jika point1 atau point2 adalah titik kosong, maka kesalahan dikembalikan.

Jika point1 atau point2 adalah null, maka null dikembalikan.

Jika point1 dan point2 sama, maka null dikembalikan.

Jika point1 atau point2 bukan titik, maka kesalahan dikembalikan.

Jika point1 dan point2 tidak memiliki nilai untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan azimuth dari titik input.

```
SELECT ST_Azimuth(ST_Point(1,2), ST_Point(5,6));
```

```
st_azimuth
-------------------
 0.7853981633974483
```

ST_Batas

ST_Boundary mengembalikan batas geometri input sebagai berikut:

- Jika geometri input kosong (yaitu, tidak mengandung titik) itu dikembalikan apa adanya.
- Jika geometri masukan adalah titik atau multipoint nonkosong, kumpulan geometri kosong dikembalikan.
- Jika input adalah linestring atau multilinestring, maka multipoint yang berisi semua titik pada batas dikembalikan. Multipoint mungkin kosong).

- Jika input adalah poligon yang tidak memiliki cincin interior, maka garis tertutup yang mewakili batasnya dikembalikan.
- Jika input adalah poligon yang memiliki cincin interior, atau multipoligon, maka multilinestring dikembalikan. Multilinestring berisi semua batas semua cincin dalam geometri areal sebagai garis tertutup.

Untuk menentukan kesetaraan titik, ST_Boundary beroperasi pada proyeksi 2D dari geometri input. Jika geometri input kosong, salinannya dikembalikan dalam dimensi yang sama dengan input. Untuk geometri 3DM dan 4D yang tidak kosong, `m` koordinatnya dijatuhkan. Dalam kasus khusus multilinestring 3DZ dan 4D, `z` koordinat titik batas multilinestring dihitung sebagai rata-rata nilai-z yang berbeda dari titik batas garis dengan proyeksi 2D yang sama.

Sintaksis

```
ST_Boundary(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`GEOMETRY`

Jika geom adalah null, maka null dikembalikan.

Jika geom adalah a`GEOMETRYCOLLECTION`, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan batas poligon input sebagai multilinestring.

```
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1
 2,2 1,1 1))')));
```

```
st_asewkt
```

```
--------------------
 MULTILINESTRING((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1))
```

ST_Buffer

ST_Buffer mengembalikan geometri 2D yang mewakili semua titik yang jaraknya dari geometri input yang diproyeksikan pada bidang XY-Cartesian kurang dari atau sama dengan jarak input.

Sintaksis

```
ST_Buffer(geom, distance)
```

```
ST_Buffer(geom, distance, number_of_segments_per_quarter_circle)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

jarak

Nilai tipe data DOUBLE PRECISION yang mewakili jarak (atau radius) buffer.

jumlah_segmen_perempat_lingkaran

Nilai tipe dataINTEGER. Nilai ini menentukan jumlah titik untuk mendekati seperempat lingkaran di sekitar setiap simpul geometri input. Nilai negatif default ke nol. Defaultnya adalah 8.

Jenis pengembalian

GEOMETRY

Fungsi ST_Buffer mengembalikan geometri dua dimensi (2D) di bidang XY-Cartesian.

Jika geom adalah aGEOMETRYCOLLECTION, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan buffer dari input linestring.

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('LINESTRING(1 2,5 2,5 8)'), 2));
```

```
               st_asewkt
POLYGON((-1 2,-0.96157056080646 2.39018064403226,-0.847759065022573
 2.76536686473018,-0.662939224605089 3.11114046603921,-0.414213562373093
 3.4142135623731,-0.111140466039201 3.66293922460509,0.234633135269824
 3.84775906502257,0.609819355967748 3.96157056080646,1 4,3 4,3 8,3.03842943919354
 8.39018064403226,3.15224093497743 8.76536686473018,3.33706077539491
 9.11114046603921,3.58578643762691 9.4142135623731,3.8888595339608
 9.66293922460509,4.23463313526982 9.84775906502257,4.60981935596775
 9.96157056080646,5 10,5.39018064403226 9.96157056080646,5.76536686473018
 9.84775906502257,6.11114046603921 9.66293922460509,6.4142135623731
 9.41421356237309,6.66293922460509 9.1111404660392,6.84775906502258
 8.76536686473017,6.96157056080646 8.39018064403225,7 8,7 2,6.96157056080646
 1.60981935596774,6.84775906502257 1.23463313526982,6.66293922460509
 0.888859533960796,6.41421356237309 0.585786437626905,6.1111404660392
 0.33706077539491,5.76536686473018 0.152240934977427,5.39018064403226
 0.0384294391935391,5 0,1 0,0.609819355967744 0.0384294391935391,0.234633135269821
 0.152240934977427,-0.111140466039204 0.337060775394909,-0.414213562373095
 0.585786437626905,-0.662939224605091 0.888859533960796,-0.847759065022574
 1.23463313526982,-0.961570560806461 1.60981935596774,-1 2))
```

SQL berikut mengembalikan buffer geometri titik input yang mendekati lingkaran. Karena perintah tidak menentukan jumlah segmen per seperempat lingkaran, fungsi menggunakan nilai default delapan segmen untuk memperkirakan seperempat lingkaran.

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('POINT(3 4)'), 2));
```

```
               st_asewkt
POLYGON((1 4,1.03842943919354 4.39018064403226,1.15224093497743
 4.76536686473018,1.33706077539491 5.11114046603921,1.58578643762691
 5.4142135623731,1.8888595339608 5.66293922460509,2.23463313526982
 5.84775906502257,2.60981935596775 5.96157056080646,3 6,3.39018064403226
 5.96157056080646,3.76536686473019 5.84775906502257,4.11114046603921
 5.66293922460509,4.4142135623731 5.41421356237309,4.66293922460509
 5.1111404660392,4.84775906502258 4.76536686473017,4.96157056080646 4.39018064403225,5
 4,4.96157056080646 3.60981935596774,4.84775906502257 3.23463313526982,4.66293922460509
 2.8888595339608,4.41421356237309 2.58578643762691,4.1111404660392
 2.33706077539491,3.76536686473018 2.15224093497743,3.39018064403226 2.03842943919354,3
 2,2.60981935596774 2.03842943919354,2.23463313526982 2.15224093497743,1.8888595339608
 2.33706077539491,1.58578643762691 2.58578643762691,1.33706077539491
 2.8888595339608,1.15224093497743 3.23463313526982,1.03842943919354 3.60981935596774,1
 4))
```

SQL berikut mengembalikan buffer geometri titik input yang mendekati lingkaran. Karena perintah menentukan 3 sebagai jumlah segmen per seperempat lingkaran, fungsi menggunakan tiga segmen untuk memperkirakan seperempat lingkaran.

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('POINT(3 4)'), 2, 3));
```

```
               st_asewkt
POLYGON((1 4,1.26794919243112 5,2 5.73205080756888,3 6,4
 5.73205080756888,4.73205080756888 5,5 4,4.73205080756888 3,4 2.26794919243112,3 2,2
 2.26794919243112,1.26794919243112 3,1 4))
```

ST_Centroid

ST_centroid mengembalikan titik yang mewakili centroid geometri sebagai berikut:

- Untuk `POINT` geometri, ia mengembalikan titik yang koordinatnya adalah rata-rata koordinat titik-titik dalam geometri.
- Untuk `LINESTRING` geometri, ia mengembalikan titik yang koordinatnya adalah rata-rata tertimbang dari titik tengah segmen geometri, di mana bobot adalah panjang segmen geometri.
- Untuk `POLYGON` geometri, ia mengembalikan titik yang koordinatnya adalah rata-rata tertimbang dari sentroid triangulasi geometri areal di mana bobotnya adalah area segitiga dalam triangulasi.
- Untuk koleksi geometri, ia mengembalikan rata-rata tertimbang dari sentroid geometri dimensi topologi maksimum dalam koleksi geometri.

Sintaksis

```
ST_Centroid(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`GEOMETRY`

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka null dikembalikan.

Contoh

SQL berikut mengembalikan titik pusat dari linestring input.

```
SELECT ST_AsEWKT(ST_Centroid(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9, -22
 -33)', 4326)))
```

```
                    st_asewkt
-----------------------------------------------------
 SRID=4326;POINT(15.6965103455214 27.0206782881905)
```

ST_Collect

ST_Collect memiliki dua varian. Satu menerima dua geometri, dan satu menerima ekspresi agregat.

Varian pertama ST_Collect menciptakan geometri dari geometri input. Urutan geometri input dipertahankan. Varian ini bekerja sebagai berikut:

- Jika kedua geometri input adalah titik, maka a `MULTIPOINT` dengan dua titik dikembalikan.
- Jika kedua geometri input adalah linestring, maka a `MULTILINESTRING` dengan dua linestring dikembalikan.
- Jika kedua geometri input adalah poligon, maka a `MULTIPOLYGON` dengan dua poligon dikembalikan.
- Jika tidak, a `GEOMETRYCOLLECTION` dengan dua geometri input dikembalikan.

Varian kedua dari ST_Collect menciptakan geometri dari geometri dalam kolom geometri. Tidak ada urutan pengembalian geometri yang ditentukan. Tentukan klausa WITHIN GROUP (ORDER BY...) untuk menentukan urutan geometri yang dikembalikan. Varian ini bekerja sebagai berikut:

- Jika semua baris non-null dalam ekspresi agregat masukan adalah titik, maka multipoint yang berisi semua titik dalam ekspresi agregat dikembalikan.
- Jika semua baris non-null dalam ekspresi agregat adalah linestring, maka multilinestring yang berisi semua linestring dalam ekspresi agregat dikembalikan.

- Jika semua baris non-null dalam ekspresi agregat adalah poligon, hasilnya adalah multipoligon yang berisi semua poligon dalam ekspresi agregat dikembalikan.
- Jika tidak, a yang GEOMETRYCOLLECTION berisi semua geometri dalam ekspresi agregat dikembalikan.

ST_Collect mengembalikan geometri dari dimensi yang sama dengan geometri input. Semua geometri input harus memiliki dimensi yang sama.

Sintaksis

```
ST_Collect(geom1, geom2)
```

```
ST_Collect(aggregate_expression)  [WITHIN GROUP (ORDER BY sort_expression1 [ASC | DESC]
 [, sort_expression2 [ASC | DESC] ...])]
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

ekspresi agregat

Kolom tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

[DALAM GRUP (URUTAN DENGAN sort_expression1 [ASC | DESC] [, sort_expres sion2 [ASC | DESC]...])]

Klausa opsional yang menentukan urutan dari nilai agregat. Klausa ORDER BY berisi daftar ekspresi pengurutan. Ekspresi pengurutan adalah ekspresi yang mirip dengan ekspresi pengurutan yang valid dalam daftar pilih kueri, seperti nama kolom. Anda dapat menentukan urutan naik (ASC) atau turun (DESC). Nilai default-nya ASC.

Jenis pengembalian

GEOMETRYdari subtipeMULTIPOINT,, MULTILINESTRINGMULTIPOLYGON, atauGEOMETRYCOLLECTION.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika semua baris agregate_expression adalah null, maka null dikembalikan.

Jika geom1 adalah null, maka salinan geom 2 dikembalikan. Demikian juga, jika geom2 adalah null, maka salinan geom 1 dikembalikan.

Jika geom1 dan geom2 memiliki nilai SRID yang berbeda, maka kesalahan dikembalikan.

Jika dua geometri dalam agregate_expression memiliki nilai SRID yang berbeda, maka kesalahan dikembalikan.

Jika geometri yang dikembalikan lebih besar dari ukuran maksimum a`GEOMETRY`, maka kesalahan dikembalikan.

Jika geom1 dan geom2 memiliki dimensi yang berbeda, maka kesalahan dikembalikan.

Jika dua geometri dalam agregate_expression memiliki dimensi yang berbeda, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan koleksi geometri yang berisi dua geometri input.

```
SELECT ST_AsText(ST_Collect(ST_GeomFromText('LINESTRING(0 0,1 1)'),
 ST_GeomFromText('POLYGON((10 10,20 10,10 20,10 10))')));
```

```
st_astext
-----------
 GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),POLYGON((10 10,20 10,10 20,10 10)))
```

SQL berikut mengumpulkan semua geometri dari tabel ke dalam koleksi geometri.

```
WITH tbl(g) AS (SELECT ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
```

```
SELECT NULL::geometry UNION ALL
SELECT ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326))
SELECT ST_AsEWKT(ST_Collect(g)) FROM tbl;
```

```
st_astext
-----------
 SRID=4326;GEOMETRYCOLLECTION(POINT(1 2),LINESTRING(0 0,10 0),MULTIPOINT((13 4),(8 5),
(4 4)),POLYGON((0 0,10 0,0 10,0 0)))
```

SQL berikut mengumpulkan semua geometri dalam tabel dikelompokkan oleh kolom id dan diurutkan oleh ID ini. Dalam contoh ini, geometri yang dihasilkan dikelompokkan berdasarkan ID sebagai berikut:

- id 1 — poin dalam multipoint.
- id 2 — linestring dalam multilinestring.
- id 3 — subtipe campuran dalam koleksi geometri.
- id 4 — poligon dalam multipoligon.
- id 5 — null dan hasilnya adalah null.

```
WITH tbl(id, g) AS (SELECT 1, ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT 1, ST_GeomFromText('POINT(4 5)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(10 0,20 -5)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5))', 4326) UNION
 ALL
SELECT 4, ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326) UNION ALL
SELECT 4, ST_GeomFromText('POLYGON((20 20,20 30,30 20,20 20))', 4326) UNION ALL
SELECT 1, NULL::geometry UNION ALL SELECT 2, NULL::geometry UNION ALL
SELECT 5, NULL::geometry UNION ALL SELECT 5, NULL::geometry)
SELECT id, ST_AsEWKT(ST_Collect(g)) FROM tbl GROUP BY id ORDER BY id;
```

```
 id |                                                st_asewkt

----
+-----------------------------------------------------------------------------------------
  1 | SRID=4326;MULTIPOINT((1 2),(4 5))
```

```
  2 | SRID=4326;MULTILINESTRING((0 0,10 0),(10 0,20 -5))
  3 | SRID=4326;GEOMETRYCOLLECTION(MULTIPOINT((13 4),(8 5),(4 4)),MULTILINESTRING((-1
 -1,-2 -2),(-3 -3,-5 -5)))
  4 | SRID=4326;MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((20 20,20 30,30 20,20 20)))
  5 |
```

SQL berikut mengumpulkan semua geometri dari tabel dalam koleksi geometri. Hasil diurutkan dalam urutan menurun berdasarkan`id`, dan kemudian secara leksikografis berdasarkan koordinat x minimum dan maksimumnya.

```
WITH tbl(id, g) AS (
SELECT 1, ST_GeomFromText('POINT(4 5)', 4326) UNION ALL
SELECT 1, ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(10 0,20 -5)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5))', 4326) UNION
 ALL
SELECT 4, ST_GeomFromText('POLYGON((20 20,20 30,30 20,20 20))', 4326) UNION ALL
SELECT 4, ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326) UNION ALL
SELECT 1, NULL::geometry UNION ALL SELECT 2, NULL::geometry UNION ALL
SELECT 5, NULL::geometry UNION ALL SELECT 5, NULL::geometry)
SELECT ST_AsEWKT(ST_Collect(g) WITHIN GROUP (ORDER BY id DESC, ST_XMin(g), ST_XMax(g)))
 FROM tbl;
```

```
                          st_asewkt

-----------------------------------------------------------------------------------------
 SRID=4326;GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0 0)),POLYGON((20 20,20 30,30
 20,20 20)),MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5)),MULTIPOINT((13 4),(8 5),(4
 4)),LINESTRING(0 0,10 0),LINESTRING(10 0,20 -5),POINT(1 2),POINT(4 5)
```

ST_berisi

ST_contains mengembalikan true jika proyeksi 2D dari geometri input pertama berisi proyeksi 2D dari geometri input kedua. Geometri `A` berisi geometri `B` jika setiap titik di dalam `B` adalah titik di`A`, dan interiornya memiliki persimpangan yang tidak kosong.

ST_contains (`A`,`B`) setara dengan ST_within (,). `B` `A`

Sintaksis

```
ST_Contains(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Nilai ini dibandingkan dengan geom1 untuk menentukan apakah itu terkandung dalam geom1.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama berisi poligon kedua.

```
SELECT ST_Contains(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_contains
-----------
 false
```

ST_ ContainsProperly

ST_ ContainsProperly mengembalikan true jika kedua geometri input tidak kosong, dan semua titik proyeksi 2D dari geometri kedua adalah titik interior dari proyeksi 2D dari geometri pertama.

Sintaksis

```
ST_ContainsProperly(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipe tidak bisaGEOMETRYCOLLECTION.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipe tidak bisaGEOMETRYCOLLECTION. Nilai ini dibandingkan dengan geom1 untuk menentukan apakah semua titiknya adalah titik interior geom1.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan nilai ST_Contains dan ST_ ContainsProperly di mana linestring input memotong interior dan batas poligon input (tetapi bukan eksteriornya). Poligon berisi linestring tetapi tidak berisi linestring dengan benar.

```
WITH tmp(g1, g2)
AS (SELECT ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0))'),
 ST_GeomFromText('LINESTRING(5 5,10 5,10 6,5 5)')) SELECT ST_Contains(g1, g2),
 ST_ContainsProperly(g1, g2)
FROM tmp;
```

```
 st_contains | st_containsproperly
```

```
------------+--------------------
 t          | f
```

ST_ ConvexHull

ST_ ConvexHull mengembalikan geometri yang mewakili lambung cembung dari titik-titik tidak kosong yang terkandung dalam geometri input.

Untuk input kosong, geometri yang dihasilkan sama dengan geometri input. Untuk semua input yang tidak kosong, fungsi beroperasi pada proyeksi 2D dari geometri input. Namun, dimensi geometri keluaran tergantung pada dimensi geometri input. Lebih khusus lagi, ketika geometri input adalah geometri 3DM atau 3D yang tidak kosong, `m` koordinat dijatuhkan. Artinya, dimensi geometri yang dikembalikan adalah 2D atau 3DZ, masing-masing. Jika input adalah geometri 2D atau 3DZ yang tidak kosong, geometri yang dihasilkan memiliki dimensi yang sama.

Sintaksis

```
ST_ConvexHull(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`GEOMETRY`

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom adalah null, maka null dikembalikan.

Nilai yang dikembalikan adalah sebagai berikut.

Jumlah titik pada lambung cembung	Subtipe geometri
0	Salinan geom dikembalikan.
1	Sebuah `POINT` subtipe dikembalikan.

Jumlah titik pada lambung cembung	Subtipe geometri
2	Sebuah `LINESTRING` subtipe dikembalikan. Dua titik dari linestring yang dikembalikan diurutkan secara leksikografis.
3 atau lebih	`POLYGON`Subtipe tanpa cincin interior dikembalikan. Poligon berorientasi searah jarum jam, dan titik pertama cincin eksterior adalah titik terkecil dari cincin secara leksikografis.

Contoh

SQL berikut mengembalikan representasi teks terkenal yang diperluas (EWKT) dari linestring. Dalam hal ini, lambung cembung yang dikembalikan adalah poligon.

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('LINESTRING(0 0,1 0,0 1,1 1,0.5 0.5)')))
 as output;
```

```
output
-------------
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

SQL berikut mengembalikan representasi EWKT dari linestring. Dalam hal ini, lambung cembung yang dikembalikan adalah garis.

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('LINESTRING(0 0,1 1,0.2 0.2,0.6 0.6,0.5
 0.5)'))) as output;
```

```
output
-------------
LINESTRING(0 0,1 1)
```

SQL berikut mengembalikan representasi EWKT dari multipoint. Dalam hal ini, lambung cembung yang dikembalikan adalah titik.

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('MULTIPOINT(0 0,0 0,0 0)'))) as output;
```

```
output
-------------
POINT(0 0)
```

ST_ CoveredBy

ST_ CoveredBy mengembalikan true jika proyeksi 2D dari geometri input pertama dicakup oleh proyeksi 2D dari geometri input kedua. Geometri A ditutupi oleh geometri B jika keduanya tidak kosong dan setiap titik di dalamnya A adalah titik diB.

ST_ CoveredBy (A,B) setara dengan ST_Covers (B,). A

Sintaksis

```
ST_CoveredBy(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Nilai ini dibandingkan dengan geom2 untuk menentukan apakah itu tercakup oleh geom2.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama ditutupi oleh poligon kedua.

```
SELECT ST_CoveredBy(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_coveredby
------------
 true
```

ST_Cover

ST_Covers mengembalikan true jika proyeksi 2D dari geometri input pertama mencakup proyeksi 2D dari geometri input kedua. Geometri A mencakup geometri B jika keduanya tidak kosong dan setiap titik di dalamnya B adalah titik dalamA.

ST_covers (A,B) setara dengan ST_ CoveredBy (B,). A

Sintaksis

```
ST_Covers(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Nilai ini dibandingkan dengan geom1 untuk menentukan apakah itu mencakup geom1.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama mencakup poligon kedua.

```
SELECT ST_Covers(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_covers
-----------
 false
```

ST_cross

ST_Crosses mengembalikan true jika proyeksi 2D dari dua geometri input saling bersilangan.

Sintaksis

```
ST_Crosses(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama melintasi multipoint kedua. Dalam contoh ini, multipoint memotong interior dan eksterior poligon, itulah sebabnya ST_Crosses mengembalikan nilai true.

```
SELECT ST_Crosses (ST_GeomFromText('polygon((0 0,10 0,10 10,0 10,0 0))'),
 ST_GeomFromText('multipoint(5 5,0 0,-1 -1)'));
```

```
st_crosses
-------------
 true
```

SQL berikut memeriksa apakah poligon pertama melintasi multipoint kedua. Dalam contoh ini, multipoint memotong bagian luar poligon tetapi bukan bagian dalamnya, itulah sebabnya ST_cross mengembalikan false.

```
SELECT ST_Crosses (ST_GeomFromText('polygon((0 0,10 0,10 10,0 10,0 0))'),
 ST_GeomFromText('multipoint(0 0,-1 -1)'));
```

```
st_crosses
-------------
 false
```

st_Dimensi

ST_Dimension mengembalikan dimensi inheren dari geometri input. Dimensi inher en adalah nilai dimensi subtipe yang didefinisikan dalam geometri.

Untuk input geometri 3DM, 3DZ, dan 4D, ST_Dimension mengembalikan hasil yang sama seperti untuk input geometri 2D.

Sintaksis

```
ST_Dimension(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

INTEGERmewakili dimensi inheren geom.

Jika geom adalah null, maka null dikembalikan.

Nilai yang dikembalikan adalah sebagai berikut.

Nilai yang dikembalikan	Subtipe geometri
0	Dikembalikan jika geom adalah subtipe POINT atau MULTIPOINT
1	Dikembalikan jika geom adalah MULTILINE STRING sub LINESTRING tipe atau.
2	Dikembalikan jika geom adalah subtipe POLYGON atau MULTIPOLYGON
0	Dikembalikan jika geom adalah subtipe kosong GEOMETRYCOLLECTION
Dimensi terbesar dari komponen koleksi	Dikembalikan jika geom adalah subtipe GEOMETRYCOLLECTION

Contoh

SQL berikut mengubah representasi teks terkenal (WKT) dari linestring empat titik ke objek GEOMETRY dan mengembalikan dimensi linestring.

```
SELECT ST_Dimension(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
```

```
st_dimension
-------------
1
```

ST_disjoint

ST_Disjoint mengembalikan true jika proyeksi 2D dari dua geometri input tidak memiliki titik yang sama.

Sintaksis

```
ST_Disjoint(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama terputus dari poligon kedua.

```
SELECT ST_Disjoint(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5
 2,2 2))'), ST_Point(4, 4));
```

```
st_disjoint
-----------
 true
```

ST_Jarak

Untuk geometri input, ST_distance mengembalikan jarak Euclidean minimum antara proyeksi 2D dari dua nilai geometri input.

Untuk geometri 3DM, 3DZ, 4D, ST_distance mengembalikan jarak Euclidean antara proyeksi 2D dari dua nilai geometri input.

Untuk masukan geografi, ST_distance mengembalikan jarak geodesik dari dua titik 2D. Satuan jarak dalam meter. Untuk geografi selain titik dan titik kosong, kesalahan dikembalikan.

Sintaksis

```
ST_Distance(geo1, geo2)
```

Argumen

geo1

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau. Tipe data geo1 harus sama dengan geo 2.

geo2

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau. Tipe data geo2 harus sama dengan geo 1.

Jenis pengembalian

DOUBLE PRECISIONdalam satuan yang sama dengan geometri masukan atau geografi.

Jika geo1 atau geo2 nol atau kosong, maka null dikembalikan.

Jika geo1 dan geo2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geo1 atau geo2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan jarak antara dua poligon.

```
SELECT ST_Distance(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 -3,-2 -1,0 -3,-1 -3))'));
```

```
  st_distance
-----------
1.4142135623731
```

SQL berikut mengembalikan jarak (dalam meter) antara Gerbang Brandenburg dan gedung Reichstag di Berlin menggunakan tipe data GEOGRAPHY.

```
SELECT ST_Distance(ST_GeogFromText('POINT(13.37761826722198 52.516411678282445)'),
 ST_GeogFromText('POINT(13.377950831464005 52.51705102546893)'));
```

```
   st_distance
------------------
 74.64129172609631
```

ST_ DistanceSphere

ST_ DistanceSphere mengembalikan jarak antara dua geometri titik yang terletak pada bola.

Sintaksis

```
ST_DistanceSphere(geom1, geom2)
```

```
ST_DistanceSphere(geom1, geom2, radius)
```

Argumen

geom1

Nilai titik dalam derajat tipe data yang GEOMETRY terletak pada bola. Koordinat pertama titik adalah nilai bujur. Koordinat kedua dari titik tersebut adalah nilai lintang. Untuk geometri 3DZ, 3DM, atau 4D, hanya dua koordinat pertama yang digunakan.

geom2

Nilai titik dalam derajat tipe data yang GEOMETRY terletak pada bola. Koordinat pertama titik adalah nilai bujur. Koordinat kedua dari titik tersebut adalah nilai lintang. Untuk geometri 3DZ, 3DM, atau 4D, hanya dua koordinat pertama yang digunakan.

jari-jari

Radius bola tipe dataDOUBLE PRECISION. Jika tidak ada radius yang disediakan, bola default ke Bumi dan radius dihitung dari representasi ellipsoid Sistem Geodetik Dunia (WGS) 84.

Jenis pengembalian

DOUBLE PRECISIONdalam satuan yang sama dengan radius. Jika tidak ada radius yang disediakan, jaraknya dalam meter.

Jika geom1 atau geom2 nol atau kosong, maka null dikembalikan.

Jika tidak ada radius yang disediakan, maka hasilnya dalam meter di sepanjang permukaan bumi.

Jika radius adalah angka negatif, maka kesalahan dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 bukan titik, maka kesalahan dikembalikan.

Contoh

Contoh berikut SQL menghitung jarak dalam kilometer antara dua titik di Bumi.

```
SELECT ROUND(ST_DistanceSphere(ST_Point(-122, 47), ST_Point(-122.1, 47.1))/ 1000, 0);
```

```
  round
-----------
 13
```

Contoh berikut SQL menghitung jarak dalam kilometer antara tiga lokasi bandara di Jerman: Berlin Tegel (TXL), Munich International (MUC), dan Frankfurt International (FRA).

```
WITH airports_raw(code,lon,lat) AS (
```

```
(SELECT 'MUC', 11.786111, 48.353889) UNION
(SELECT 'FRA', 8.570556, 50.033333) UNION
(SELECT 'TXL', 13.287778, 52.559722)),
airports1(code,location) AS (SELECT code, ST_Point(lon, lat) FROM airports_raw),
airports2(code,location) AS (SELECT * from airports1)
SELECT (airports1.code || ' <-> ' || airports2.code) AS airports,
round(ST_DistanceSphere(airports1.location, airports2.location) / 1000, 0) AS
 distance_in_km
FROM airports1, airports2 WHERE airports1.code < airports2.code ORDER BY 1;
```

```
  airports   | distance_in_km
-------------+----------------
 FRA <-> MUC |            299
 FRA <-> TXL |            432
 MUC <-> TXL |            480
```

ST_Dwithin

ST_DWithin mengembalikan true jika jarak Euclidean antara proyeksi 2D dari dua nilai geometri input tidak lebih besar dari nilai ambang.

Sintaksis

```
ST_DWithin(geom1, geom2, threshold)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

ambang batas

Nilai tipe dataDOUBLE PRECISION. Nilai ini ada dalam satuan argumen input.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika ambang batas negatif, maka kesalahan dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah jarak antara dua poligon berada dalam lima unit.

```
SELECT ST_DWithin(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'),5);
```

```
st_dwithin
-----------
 true
```

ST_ EndPoint

ST_ EndPoint mengembalikan titik terakhir dari linestring input. Nilai pengidentifikasi sistem referensi spasial (SRID) dari hasilnya sama dengan geometri input. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_EndPoint(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka null dikembalikan.

Jika geom bukan a`LINESTRING`, maka null dikembalikan.

Contoh

SQL berikut mengembalikan representasi teks terkenal yang diperluas (EWKT) dari empat titik `LINESTRING` ke `GEOMETRY` objek dan mengembalikan titik akhir dari linestring.

```
SELECT ST_AsEWKT(ST_EndPoint(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0
 5)',4326)));
```

```
st_asewkt
-------------
 SRID=4326;POINT(0 5)
```

ST_Amplop

ST_ENVELOPE mengembalikan kotak pembatas minimum dari geometri input, sebagai berikut:

- Jika geometri input kosong, geometri yang dikembalikan adalah salinan dari geometri input.
- Jika kotak pembatas minimum dari geometri input merosot ke suatu titik, geometri yang dikembalikan adalah titik.
- Jika kotak pembatas minimum dari geometri input adalah satu dimensi, garis dua titik dikembalikan.
- Jika tidak ada yang sebelumnya benar, fungsi mengembalikan poligon berorientasi searah jarum jam yang simpulnya adalah sudut kotak pembatas minimum.

Pengidentifikasi sistem referensi spasial (SRID) dari geometri yang dikembalikan sama dengan geometri input.

Untuk semua input yang tidak kosong, fungsi beroperasi pada proyeksi 2D dari geometri input.

Sintaksis

```
ST_Envelope(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengubah representasi teks terkenal (WKT) dari empat titik LINESTRING ke GEOMETRY objek dan mengembalikan poligon yang simpul yang sudutnya adalah kotak pembatas minimum.

```
SELECT ST_AsText(ST_Envelope(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0
 10,0 0)),LINESTRING(20 10,20 0,10 0))')));
```

```
    st_astext
-----------------------------------
  POLYGON((0 0,0 10,20 10,20 0,0 0))
```

ST_sama

ST_Equals mengembalikan true jika proyeksi 2D dari geometri input sama secara geometris. Geometri dianggap sama secara geometris jika mereka memiliki himpunan titik yang sama dan interiornya memiliki persimpangan yang tidak kosong.

Sintaksis

```
ST_Equals(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Nilai ini dibandingkan dengan geom1 untuk menentukan apakah sama dengan geom1.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka kesalahan dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah kedua poligon secara geometris sama.

```
SELECT ST_Equals(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_equals
-----------
 false
```

SQL berikut memeriksa apakah dua linestring sama secara geometris.

```
SELECT ST_Equals(ST_GeomFromText('LINESTRING(1 0,10 0)'), ST_GeomFromText('LINESTRING(1
 0,5 0,10 0)'));
```

```
st_equals
-----------
 true
```

ST_ ExteriorRing

ST_ ExteriorRing mengembalikan garis tertutup yang mewakili cincin eksterior dari poligon input. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_ExteriorRing(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRYdari subtipeLINESTRING.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom adalah null, maka null dikembalikan.

Jika geom bukan poligon, maka null dikembalikan.

Jika geom kosong, maka poligon kosong dikembalikan.

Contoh

SQL berikut mengembalikan cincin eksterior poligon sebagai linestring tertutup.

```
SELECT ST_AsEWKT(ST_ExteriorRing(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17
 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12
 14,15 14,13 11,12 14))')));
```

```
st_asewkt
-----------
 LINESTRING(7 9,8 7,11 6,15 8,16 6,17 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9)
```

ST_Force2D

ST_force2D mengembalikan geometri 2D dari geometri input. Untuk geometri 2D, salinan input dikembalikan. Untuk geometri 3DZ, 3DM, dan 4D, ST_Force2D memproyeksikan geometri ke bidang XY-Cartesian. Titik kosong dalam geometri input tetap titik kosong dalam geometri keluaran.

Sintaksis

```
ST_Force2D(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka geometri kosong dikembalikan.

Contoh

SQL berikut mengembalikan geometri 2D dari geometri 3DZ.

```
SELECT ST_AsEWKT(ST_Force2D(ST_GeomFromText('MULTIPOINT Z(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT((0 1),EMPTY,(2 3),(5 6))
```

ST_Force3d

ST_force3d adalah alias untuk ST_Force3dz. Untuk informasi selengkapnya, lihat ST_Force3DZ.

ST_Force3DM

ST_force3DM mengembalikan geometri 3DM dari geometri input. Untuk geometri 2D, `m` koordinat titik-titik tidak kosong dalam geometri keluaran semuanya diatur ke. `0` Untuk geometri 3DM, salinan geometri input dikembalikan. Untuk geometri 3DZ, geometri diproyeksikan ke bidang XY-Cartesian, dan `m` koordinat titik-titik tidak kosong dalam geometri keluaran semuanya diatur ke. `0` Untuk geometri 4D, geometri diproyeksikan ke ruang XYM-Cartesian. Titik kosong dalam geometri input tetap titik kosong dalam geometri keluaran.

Sintaksis

```
ST_Force3DM(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`GEOMETRY`.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka geometri kosong dikembalikan.

Contoh

SQL berikut mengembalikan geometri 3DM dari geometri 3DZ.

```
SELECT ST_AsEWKT(ST_Force3DM(ST_GeomFromText('MULTIPOINT Z(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
```

```
  MULTIPOINT M ((0 1 0),EMPTY,(2 3 0),(5 6 0))
```

ST_Force3DZ

ST_force3DZ mengembalikan geometri 3DZ dari geometri input. Untuk geometri 2D, z koordinat titik-titik tidak kosong dalam geometri keluaran semuanya diatur ke. 0 Untuk geometri 3DM, geometri diproyeksikan pada bidang XY-Cartesian, dan z koordinat titik-titik tidak kosong dalam geometri keluaran semuanya diatur ke. 0 Untuk geometri 3DZ, salinan geometri input dikembalikan. Untuk geometri 4D, geometri diproyeksikan ke ruang XYZ-Cartesian. Titik kosong dalam geometri input tetap titik kosong dalam geometri keluaran.

Sintaksis

```
ST_Force3DZ(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka geometri kosong dikembalikan.

Contoh

SQL berikut mengembalikan geometri 3DZ dari geometri 3DM.

```
SELECT ST_AsEWKT(ST_Force3DZ(ST_GeomFromText('MULTIPOINT M(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT Z ((0 1 0),EMPTY,(2 3 0),(5 6 0))
```

ST_Force4d

ST_force4d mengembalikan geometri 4D dari geometri input. Untuk geometri 2D, `m` koordin `z` at dan titik tidak kosong dalam geometri keluaran semuanya diatur ke. `0` Untuk geometri 3DM, `z` koordinat titik-titik tidak kosong dalam geometri keluaran semuanya diatur ke. `0` Untuk geometri 3DZ, `m` koordinat titik-titik tidak kosong dalam geometri keluaran semuanya diatur ke. `0` Untuk geometri 4D, salinan geometri input dikembalikan. Titik kosong dalam geometri input tetap titik kosong dalam geometri keluaran.

Sintaksis

```
ST_Force4D(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`GEOMETRY`.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka geometri kosong dikembalikan.

Contoh

SQL berikut mengembalikan geometri 4D dari geometri 3DM.

```
SELECT ST_AsEWKT(ST_Force4D(ST_GeomFromText('MULTIPOINT M(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT ZM ((0 1 0 2),EMPTY,(2 3 0 4),(5 6 0 7))
```

ST_ GeoHash

ST_ GeoHash mengembalikan geohash representasi titik input dengan presisi yang ditentukan. Nilai presisi default adalah 20. Untuk informasi lebih lanjut tentang definisi geohash, lihat Geoh ash di Wikipedia.

Sintaksis

```
ST_GeoHash(geom)
```

```
ST_GeoHash(geom, precision)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

presisi

Nilai tipe dataINTEGER. Defaultnya adalah 20.

Jenis pengembalian

GEOMETRY

Fungsi mengembalikan geohash representasi titik input.

Jika titik input kosong, fungsi mengembalikan null.

Jika geometri input bukan titik, fungsi mengembalikan kesalahan.

Contoh

SQL berikut mengembalikan representasi geohash dari titik input.

```
SELECT ST_GeoHash(ST_GeomFromText('POINT(45 -45)'), 25) AS geohash;
```

```
         geohash
-------------------------
 m0000000000000000000gzz
```

SQL berikut mengembalikan null karena titik input kosong.

```
SELECT ST_GeoHash(ST_GeomFromText('POINT EMPTY'), 10) IS NULL AS result;
```

```
 result
---------
 true
```

ST_ GeogFromText

ST_ GeogFromText membangun objek geografi dari teks terkenal (WKT) atau teks terkenal diperpanjang (EWKT) representasi dari geografi masukan.

Sintaksis

```
ST_GeogFromText(wkt_string)
```

Argumen

wkt_string

Nilai tipe data VARCHAR yang merupakan representasi WKT atau EWKT dari suatu geografi.

Jenis pengembalian

GEOGRAPHY

Jika nilai SRID diatur ke nilai yang disediakan dalam input. Jika SRID tidak disediakan, itu disetel ke4326.

Jika wkt_string adalah null, maka null dikembalikan.

Jika wkt_string tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun poligon dari objek geografi dengan nilai SRID.

```
SELECT ST_AsEWKT(ST_GeogFromText('SRID=4324;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))'));
```

```
  st_asewkt
----------------------------------------------
 SRID=4324;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))
```

SQL berikut membangun poligon dari objek geografi. Nilai SRID disetel ke4326.

```
SELECT ST_AsEWKT(ST_GeogFromText('POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))'));
```

```
 st_asewkt
----------------------------------------------
 SRID=4326;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))
```

ST_WKB GeogFrom

ST_ GeogFrom WKB membangun objek geografi dari representasi biner terkenal heksadesimal (WKB) dari geografi masukan.

Sintaksis

```
ST_GeogFromWKB(wkb_string)
```

Argumen

wkb_string

Nilai tipe data VARCHAR yang merupakan representasi WKB heksadesimal dari suatu geografi.

Jenis pengembalian

GEOGRAPHY

Jika nilai SRID disediakan, itu diatur ke nilai yang disediakan. Jika SRID tidak disediakan, itu disetel ke4326.

Jika wkb_string adalah null, maka null dikembalikan.

Jika wkb_string tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun geografi dari nilai WKB heksadesimal.

```
SELECT
 ST_AsEWKT(ST_GeogFromWKB('010300000001000000050000000000000000000000000000000000000000000000000000000000000000
```

```
 st_asewkt
-------------------------------------------
 SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_Geometryn

ST_Geometryn mengembalikan geometri yang ditunjukkan oleh indeks input geometri input, sebagai berikut:

- Jika input adalah titik, linestring, atau poligon, maka geometri dikembalikan seperti jika indeks sama dengan satu (1), dan nol jika indeks selain satu (1).
- Jika input adalah kumpulan multipoint, multilinestring, multipoligon, atau geometri, maka titik, linestring, poligon, atau koleksi geometri dikembalikan seperti yang ditunjukkan oleh indeks input.

Indeks ini berbasis satu. Pengidentifikasi sistem referensi spasial (SRID) dari hasilnya sama dengan geometri input. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_GeometryN(geom, index)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

indeks

Nilai tipe data INTEGER yang mewakili posisi indeks berbasis satu.

Jenis pengembalian

GEOMETRY

Jika geom atau indeks adalah null, maka null dikembalikan.

Jika indeks berada di luar jangkauan, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan geometri dalam koleksi geometri.

```
WITH tmp1(idx) AS (SELECT 1 UNION SELECT 2),
tmp2(g) AS (SELECT ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0
 0)),LINESTRING(20 10,20 0,10 0))'))
SELECT idx, ST_AsEWKT(ST_GeometryN(g, idx)) FROM tmp1, tmp2 ORDER BY idx;
```

```
 idx |          st_asewkt
-----+-----------------------------
   1 | POLYGON((0 0,10 0,0 10,0 0))
   2 | LINESTRING(20 10,20 0,10 0)
```

ST_ GeometryType

ST_ GeometryType mengembalikan subtipe geometri input sebagai string.

Untuk input geometri 3DM, 3DZ, dan 4D, ST_ GeometryType mengembalikan hasil yang sama seperti untuk input geometri 2D.

Sintaksis

```
ST_GeometryType(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

VARCHARmewakili subtipe ge om.

Jika geom adalah null, maka null dikembalikan.

Nilai yang dikembalikan adalah sebagai berikut.

Nilai string yang dikembalikan	Subtipe geometri
ST_Point	Dikembalikan jika geom adalah subtipe POINT
ST_LineString	Dikembalikan jika geom adalah subtipe LINESTRING
ST_Polygon	Dikembalikan jika geom adalah subtipe POLYGON
ST_MultiPoint	Dikembalikan jika geom adalah subtipe MULTIPOINT
ST_MultiLineString	Dikembalikan jika geom adalah subtipe MULTILINESTRING
ST_MultiPolygon	Dikembalikan jika geom adalah subtipe MULTIPOLYGON
ST_GeometryCollection	Dikembalikan jika geom adalah subtipe GEOMETRYCOLLECTION

Contoh

SQL berikut mengembalikan subtipe geometri linestring input.

```
SELECT ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
```

```
st_geometrytype
-------------
 ST_LineString
```

ST_EWKB GeomFrom

ST_ GeomFrom EWKB membangun objek geometri dari representasi biner terkenal yang diperluas (EWKB) dari geometri input.

ST_ GeomFrom EWKB menerima geometri 3DZ, 3DM, dan 4D yang ditulis dalam format heksadesimal WKB dan EWKB.

Sintaksis

```
ST_GeomFromEWKB(ewkb_string)
```

Argumen

ewkb_string

Nilai tipe data VARCHAR atau VARBYTE yang merupakan representasi EWKB heksadesimal dari geometri.

Jenis pengembalian

GEOMETRY

Jika ewkb_string adalah null, maka null dikembalikan.

Jika ewkb_string tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun poligon dari nilai EWKB dan mengembalikan representasi EWKT dari poligon.

```
SELECT
 ST_AsEWKT(ST_GeomFromEWKB('0103000020E6100000010000000500000000000000000000000000000000000000000
```

```
 st_asewkt
-------------------------------
 SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_EWKT GeomFrom

ST_ GeomFrom EWKT membangun objek geometri dari representasi teks terkenal diperpanjang (EWKT) dari geometri input.

ST_ GeomFrom EWKT menerima 3DZ, 3DM, dan 4D di mana tipe geometri diawali dengan Z, M, atau ZM, masing-masing.

Sintaksis

```
ST_GeomFromEWKT(ewkt_string)
```

Argumen

ewkt_string

Nilai tipe data VARCHAR atau ekspresi yang mengevaluasi ke VARCHAR tipe, yaitu representasi EWKT dari geometri.

Anda dapat menggunakan kata kunci WKT EMPTY untuk menunjuk titik kosong, multipoint dengan titik kosong, atau koleksi geometri dengan titik kosong. Contoh berikut menciptakan titik kosong.

```
ST_GeomFromEWKT('SRID=4326;POINT EMPTY');
```

Jenis pengembalian

GEOMETRY

Jika ewkt_string adalah null, maka null dikembalikan.

Jika ewkt_string tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun multilinestring dari nilai EWKT dan mengembalikan geometri. Ini juga mengembalikan hasil ST_ASewkt dari geometri.

```
SELECT ST_GeomFromEWKT('SRID=4326;MULTILINESTRING((1 0,1 0),(2 0,3 0),(4 0,5 0,6 0))')
 as geom, ST_AsEWKT(geom);
```

```
                                                    geom

                                          |                         st_asewkt

-------------------------------------------------------------------------------------------------------
+-----------------------------------------------------------
 0105000020E6100000030000000102000000020000000000000000F03F0000000000000000000000000000F03F00
 | SRID=4326;MULTILINESTRING((1 0,1 0),(2 0,3 0),(4 0,5 0,6 0))
```

ST_ GeomFromGeoHash

ST_ membangun GeomFromGeoHash objek geometri dari representasi geohash dari geometri input. ST_ GeomFromGeoHash mengembalikan geometri dua dimensi (2D) dengan pengidentifikasi referensi spasial (SRID) nol (0). Untuk informasi lebih lanjut tentang format geohash, lihat Geoh ash di Wikipedia.

Sintaksis

```
ST_GeomFromGeoHash(geohash_string)
```

```
ST_GeomFromGeoHash(geohash_string, precision)
```

Argumen

geohash_string

Nilai tipe data VARCHAR atau ekspresi yang mengevaluasi ke VARCHAR tipe, yaitu representasi geohash dari geometri.

presisi

Nilai tipe data INTEGER yang mewakili ketepatan geohash. Nilai adalah jumlah karakter geohash yang akan digunakan sebagai presisi. Jika nilainya tidak ditentukan, kurang dari nol, atau lebih besar dari panjang geohash_string. Maka panjang geohash_string digunakan.

Jenis pengembalian

GEOMETRY

Jika geohash_string adalah null, maka null dikembalikan.

Jika geohash_string tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan poligon dengan presisi tinggi.

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0'));
```

```
 st_asewkt
-----------------------
 POLYGON((-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646,-115.172816
 36.114646,-115.172816 36.114646))
```

SQL berikut mengembalikan titik dengan presisi tinggi.

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz00'));
```

```
 st_asewkt
-----------------------
 POINT(-115.172816 36.114646)
```

SQL berikut mengembalikan poligon dengan presisi rendah.

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qq'));
```

```
 st_asewkt
----------------------
 POLYGON((-115.3125 35.15625,-115.3125 36.5625,-113.90625 36.5625,-113.90625
 35.15625,-115.3125 35.15625))
```

SQL berikut mengembalikan poligon dengan presisi 3.

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 3));
```

```
 st_asewkt
----------------------
 POLYGON((-115.3125 35.15625,-115.3125 36.5625,-113.90625 36.5625,-113.90625
 35.15625,-115.3125 35.15625))
```

ST_ GeomFromGeoJSON

ST_ GeomFromGeoJSON membangun objek geometri dari representasi GeoJSON dari geometri input. Untuk informasi lebih lanjut tentang format GeoJSON, lihat GeoJSON di Wikipedia.

Jika setidaknya ada satu titik dengan tiga atau lebih koordinat, geometri yang dihasilkan adalah 3DZ, di mana komponen Z adalah nol untuk titik-titik yang hanya memiliki dua koordinat. Jika semua titik dalam input GeoJSON berisi dua koordinat atau kosong, ST_ GeomFromGeoJSON mengembalikan geometri 2D. Geometri yang dikembalikan selalu memiliki pengenal referensi spasial (SRID) 4326.

Sintaksis

```
ST_GeomFromGeoJSON(geojson_string)
```

Argumen

geojson_string

Nilai tipe data VARCHAR atauSUPER, atau ekspresi yang mengevaluasi ke VARCHAR tipe, yaitu representasi geoJSON dari geometri.

Jenis pengembalian

GEOMETRY

Jika geojson_string adalah null, maka null dikembalikan.

Jika geojson_string tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan geometri 2D yang diwakili dalam masukan GeoJSON.

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"Point","coordinates":[1,2]}'));
```

```
 st_asewkt
-----------------------
 SRID=4326;POINT(1 2)
```

SQL berikut mengembalikan geometri 3DZ yang diwakili dalam masukan GeoJSON.

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"LineString","coordinates":[[1,2,3],
[4,5,6],[7,8,9]]}'));
```

```
 st_asewkt
----------------------------------------
 SRID=4326;LINESTRING Z (1 2 3,4 5 6,7 8 9)
```

SQL berikut mengembalikan geometri 3DZ ketika hanya satu titik memiliki tiga koordinat sementara semua titik lainnya memiliki dua koordinat dalam masukan GeoJSON.

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"Polygon","coordinates":[[[0, 0],[0, 1,
 8],[1, 0],[0, 0]]]}'));
```

```
 st_asewkt
```

```
-----------------------------------------------
 SRID=4326;POLYGON Z ((0 0 0,0 1 8,1 0 0,0 0 0))
```

ST_ GeomFromGeoSquare

ST_ GeomFromGeoSquare mengembalikan geometri yang mencakup area yang diwakili oleh nilai masukan geosquare. Geometri yang dikembalikan selalu dua dimensi. Untuk menghitung nilai geosquare, lihat. ST_ GeoSquare

Sintaksis

```
ST_GeomFromGeoSquare(geosquare)
```

```
ST_GeomFromGeoSquare(geosquare, max_depth)
```

Argumen

geosquare

Nilai tipe data BIGINT atau ekspresi yang mengevaluasi ke BIGINT tipe yang merupakan nilai geosquare yang menggambarkan urutan subdivisi yang dibuat pada domain awal untuk mencapai kuadrat yang diinginkan. Nilai ini dihitung olehST_ GeoSquare.

kedalaman maksimum

Nilai tipe data INTEGER yang mewakili jumlah maksimum subdivisi domain yang dibuat pada domain awal. Nilai harus lebih besar dari atau sama dengan1.

Jenis pengembalian

GEOMETRY

Jika geosquare tidak valid, fungsi mengembalikan kesalahan.

Jika input max_deep tidak dalam jangkauan, fungsi mengembalikan kesalahan.

Contoh

SQL berikut mengembalikan geometri dari nilai geosquare.

```
SELECT ST_AsText(ST_GeomFromGeoSquare(797852));
```

```
 st_astext
-----------------------------------------------------------------------------------------------------------
 POLYGON((13.359375 52.3828125,13.359375 52.734375,13.7109375 52.734375,13.7109375
 52.3828125,13.359375 52.3828125))
```

SQL berikut mengembalikan geometri dari nilai geosquare dan kedalaman maksimum. 3

```
SELECT ST_AsText(ST_GeomFromGeoSquare(797852, 3));
```

```
 st_astext
-------------------------------------
 POLYGON((0 45,0 90,45 90,45 45,0 45))
```

SQL berikut pertama-tama menghitung nilai geosquare untuk Seattle dengan menentukan koordinat x sebagai garis bujur dan koordinat y sebagai garis lintang (-122.3, 47.6). Kemudian mengembalikan poligon untuk geosquare. Meskipun output adalah geometri dua dimensi, ini dapat digunakan untuk menghitung data spasial dalam hal bujur dan lintang.

```
SELECT ST_AsText(ST_GeomFromGeoSquare(ST_GeoSquare(ST_Point(-122.3, 47.6))));
```

```
 st_astext
-----------------------------------------------------------------------------------------------------------
POLYGON((-122.335167014971 47.6080129947513,-122.335167014971
 47.6080130785704,-122.335166931152 47.6080130785704,-122.335166931152
 47.6080129947513,-122.335167014971 47.6080129947513))
```

ST_ GeomFromText

ST_ GeomFromText membangun objek geometri dari representasi teks terkenal (WKT) dari geometri input.

ST_ GeomFromText menerima 3DZ, 3DM, dan 4D di mana tipe geometri diawali dengan Z, M, atau ZM, masing-masing.

Sintaksis

```
ST_GeomFromText(wkt_string)
```

```
ST_GeomFromText(wkt_string, srid)
```

Argumen

wkt_string

Nilai tipe data VARCHAR yang merupakan representasi WKT dari geometri.

Anda dapat menggunakan kata kunci WKT EMPTY untuk menunjuk titik kosong, multipoint dengan titik kosong, atau koleksi geometri dengan titik kosong. Contoh berikut membuat multipoint dengan satu titik kosong dan satu titik tidak kosong.

```
ST_GeomFromEWKT('MULTIPOINT(1 0,EMPTY)');
```

menjijikkan

Nilai tipe data INTEGER yang merupakan pengenal referensi spasial (SRID). Jika nilai SRID disediakan, geometri yang dikembalikan memiliki nilai SRID ini. Jika tidak, nilai SRID dari geometri yang dikembalikan diatur ke nol (0).

Jenis pengembalian

GEOMETRY

Jika wkt_string atau srid adalah null, maka null dikembalikan.

Jika srid negatif, maka null dikembalikan.

Jika wkt_string tidak valid, maka kesalahan dikembalikan.

Jika srid tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun objek geometri dari representasi WKT dan nilai SRID.

```
SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326);
```

```
st_geomfromtext
-------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000000000000000000000000000000000F
```

ST_WKB GeomFrom

ST_ GeomFrom WKB membangun objek geometri dari representasi biner terkenal heksadesimal (WKB) dari geometri input.

ST_ GeomFrom WKB menerima geometri 3DZ, 3DM, dan 4D yang ditulis dalam format heksadesimal WKB.

Sintaksis

```
ST_GeomFromWKB(wkb_string)
```

```
ST_GeomFromWKB(wkb_string, srid)
```

Argumen

wkb_string

Nilai tipe data VARCHAR atau VARBYTE yang merupakan representasi WKB heksadesimal dari geometri.

menjijikkan

Nilai tipe data INTEGER yang merupakan pengenal referensi spasial (SRID). Jika nilai SRID disediakan, geometri yang dikembalikan memiliki nilai SRID ini. Jika tidak, nilai SRID dari geometri yang dikembalikan diatur ke 0.

Jenis pengembalian

GEOMETRY

Jika wkb_string atau srid adalah null, maka null dikembalikan.

Jika srid negatif, maka null dikembalikan.

Jika wkb_string tidak valid, maka kesalahan dikembalikan.

Jika srid tidak valid, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun poligon dari nilai WKB dan mengembalikan representasi WKT dari poligon.

```
SELECT
 ST_AsText(ST_GeomFromWKB('010300000001000000050000000000000000000000000000000000000000000000000000000000000
```

```
 st_astext
-------------------------------
 POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_ GeoSquare

ST_ GeoSquare secara rekursif membagi domain ([-180, 180], [-90, 90]) menjadi wilayah persegi yang sama yang disebut ge osquares ke kedalaman tertentu. Subdivisi didasarkan pada lokasi titik yang disediakan. Salah satu geosquares yang berisi titik dibagi lagi di setiap langkah hingga mencapai kedalaman maksimum. Pemilihan geosquare ini stabil, yaitu hasil fungsi hanya bergantung pada argumen input. Fungsi mengembalikan nilai unik yang mengidentifikasi geosquare akhir di mana titik tersebut berada.

ST_ GeoSquare menerima TITIK di mana koordinat x mewakili bujur, dan koordinat y mewakili garis lintang. Garis bujur dan lintang masing-masing terbatas pada [-180, 180] dan [-90, 90]. Output ST_ GeoSquare dapat digunakan sebagai input ke ST_ GeomFromGeoSquare fungsi.

Ada 360° di sekitar busur lingkar ekuator Bumi yang dibagi menjadi dua belahan bumi (Timur dan Barat), masing-masing dengan 180° garis longitudinal (Meridian) dari 0° Meridian. Secara konvensi, garis bujur timur adalah koordinat “+” (positif) ketika diproyeksikan ke sumbu x pada bidang Cartesian dan garis bujur barat adalah koordinat “-” (negatif) ketika diproyeksikan ke sumbu x pada bidang Cartesian. Ada 90° garis lintang utara dan selatan lingkar ekuator 0° Bumi, masing-masing sejajar dengan keliling ekuator 0° Bumi. Secara konvensi, garis lintang utara memotong sumbu y “+” (positif) ketika diproyeksikan ke bidang Cartesian, dan garis lintang selatan memotong sumbu y “-” (negatif) ketika diproyeksikan ke bidang Cartesian. Kisi bola yang dibentuk oleh persimpangan garis longitudinal dan garis lintang diubah menjadi grid yang diproyeksikan ke bidang Cartesian dengan koordinat x positif dan negatif standar dan koordinat y positif dan negatif pada bidang Cartesian.

Tujuan ST_ GeoSquare adalah untuk menandai atau menandai titik penutupan dengan nilai kode yang sama. Poin yang terletak di geosquare yang sama menerima nilai kode yang sama. Geosquare digunakan untuk mengkodekan koordinat geografis (lintang dan bujur) menjadi bilangan bulat. Wilayah yang lebih besar dibagi menjadi grid untuk menggambarkan area pada peta dengan resolusi yang bervariasi. Geosquare dapat digunakan untuk pengindeksan spasial, binning spasial, pencarian jarak, pencarian lokasi, dan membuat pengidentifikasi tempat yang unik. ST_ GeoHashFungsi ini mengikuti proses serupa membagi wilayah menjadi grid, tetapi memiliki pengkodean yang berbeda.

Sintaksis

```
ST_GeoSquare(geom)
```

```
ST_GeoSquare(geom, max_depth)
```

Argumen

geom

Nilai POINT dari tipe data GEOMETRY atau ekspresi yang mengevaluasi ke subtipe POINT. Koordinat x (bujur) titik harus berada dalam kisaran: -180 —180. Koordinat y (lintang) titik harus berada dalam kisaran: -90 —90.

kedalaman maksimum

Nilai tipe dataINTEGER. Jumlah maksimum kali domain yang berisi titik dibagi lagi secara rekursif. Nilai harus berupa bilangan bulat dari 1 — 32. Default-nya adalah 32. Jumlah akhir yang sebenarnya dari subdivisi kurang dari atau sama dengan max _deep yang ditentukan.

Jenis pengembalian

BIGINT

Fungsi mengembalikan nilai unik yang mengidentifikasi geosquare akhir di mana titik input berada.

Jika geom input bukan titik, fungsi mengembalikan kesalahan.

Jika titik input kosong, nilai pengembalian bukanlah input yang valid ke ST_ GeomFromGeoSquare fungsi. Gunakan ST_ IsEmpty fungsi untuk mencegah panggilan ke ST_ GeoSquare dengan titik kosong.

Jika titik input tidak dalam jangkauan, fungsi mengembalikan kesalahan.

Jika input max_deep berada di luar jangkauan, fungsi mengembalikan kesalahan.

Contoh

SQL berikut mengembalikan geosquare dari titik input.

```
SELECT ST_GeoSquare(ST_Point(13.5, 52.5));
```

```
  st_geosquare
-----------------------
 -4410772491521635895
```

SQL berikut mengembalikan geosquare dari titik input dengan kedalaman maksimum. 10

```
SELECT ST_GeoSquare(ST_Point(13.5, 52.5), 10);
```

```
 st_geosquare
--------------
 797852
```

ST_N InteriorRing

ST_ InteriorRing N mengembalikan garis tertutup yang sesuai dengan cincin interior poligon input pada posisi indeks. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_InteriorRingN(geom, index)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

indeks

Nilai tipe data INTEGER yang mewakili posisi cincin indeks berbasis satu.

Jenis pengembalian

`GEOMETRY`dari subtipe`LINESTRING`.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom atau indeks adalah null, maka null dikembalikan.

Jika indeks berada di luar jangkauan, maka null dikembalikan.

Jika geom bukan poligon, maka null dikembalikan.

Jika geom adalah poligon kosong, maka null dikembalikan.

Contoh

SQL berikut mengembalikan cincin kedua dari poligon sebagai linestring tertutup.

```
SELECT ST_AsEWKT(ST_InteriorRingN(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17
 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12
 14,15 14,13 11,12 14))'),2));
```

```
st_asewkt
-----------
 LINESTRING(12 14,15 14,13 11,12 14)
```

ST_berpotongan

ST_intersects mengembalikan nilai true jika proyeksi 2D dari dua geometri input memiliki setidaknya satu titik yang sama.

Sintaksis

```
ST_Intersects(geom1, geom2)
```

Argumen

geom1

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama memotong poligon kedua.

```
SELECT ST_Intersects(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5
 2,2 2))'), ST_GeomFromText('MULTIPOINT((4 4),(6 6))'));
```

```
st_intersects
-------------
 true
```

ST_Persimpangan

ST_Intersection mengembalikan geometri yang mewakili persimpangan titik-set dari dua geometri. Artinya, ia mengembalikan bagian dari dua geometri input yang dibagi di antara mereka.

Sintaksis

```
ST_Intersection(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY

Jika geom1 dan geom2 tidak berbagi ruang apa pun (mereka terpisah), maka geometri kosong dikembalikan.

Jika geom1 atau geom2 kosong, maka geometri kosong dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Jika geom1 atau geom2 bukan geometri dua dimensi (2D), maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan geometri non-kosong yang mewakili persimpangan dua geometri input.

```
SELECT ST_AsEWKT(ST_Intersection(ST_GeomFromText('polygon((0 0,100 100,0 200,0 0))'),
 ST_GeomFromText('polygon((0 0,10 0,0 10,0 0))')));
```

```
        st_asewkt
-------------------------
 POLYGON((0 0,0 10,5 5,0 0))
```

SQL berikut mengembalikan geometri kosong ketika melewati geometri masukan terpisah (tidak berpotongan).

```
SELECT ST_AsEWKT(ST_Intersection(ST_GeomFromText('linestring(0 100,0 0)'),
 ST_GeomFromText('polygon((1 0,10 0,1 10,1 0))')));
```

```
    st_asewkt
------------------
 LINESTRING EMPTY
```

ST_CCW IsPolygon

ST_ IsPolygon CCW mengembalikan true jika proyeksi 2D dari poligon input atau multipoligon berlawanan arah jarum jam. Jika geometri input adalah titik, linestring, multipoint, atau multilinestring, maka true dikembalikan. Untuk koleksi geometri, ST_ IsPolygon CCW mengembalikan true jika semua geometri dalam koleksi berlawanan arah jarum jam.

Sintaksis

```
ST_IsPolygonCCW(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`BOOLEAN`

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon berlawanan arah jarum jam.

```
SELECT ST_IsPolygonCCW(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17 7,17 10,18
 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12 14,15 14,13
 11,12 14))'));
```

```
 st_ispolygonccw
----------
 true
```

ST_CW IsPolygon

ST_ IsPolygon CW mengembalikan true jika proyeksi 2D dari poligon input atau multipoligon searah jarum jam. Jika geometri input adalah titik, linestring, multipoint, atau multilinestring, maka true dikembalikan. Untuk koleksi geometri, ST_ IsPolygon CW mengembalikan true jika semua geometri dalam koleksi searah jarum jam.

Sintaksis

```
ST_IsPolygonCW(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon searah jarum jam.

```
SELECT ST_IsPolygonCW(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17 7,17 10,18
 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12 14,15 14,13
 11,12 14))'));
```

```
 st_ispolygonccw
----------
 true
```

ST_ IsClosed

ST_ IsClosed mengembalikan true jika proyeksi 2D dari geometri input ditutup. Aturan berikut menentukan geometri tertutup:

- Geometri input adalah titik atau multipoint.
- Geometri input adalah linestring, dan titik awal dan akhir dari linestring bertepatan.
- Geometri input adalah multilinestring yang tidak kosong dan semua linestring ditutup.
- Geometri input adalah poligon yang tidak kosong, semua cincin poligon tidak kosong, dan titik awal dan akhir dari semua cincinnya bertepatan.
- Geometri input adalah multipoligon yang tidak kosong dan semua poligonnya tertutup.
- Geometri input adalah kumpulan geometri yang tidak kosong dan semua komponennya ditutup.

Sintaksis

```
ST_IsClosed(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah titik kosong, maka false dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon ditutup.

```
SELECT ST_IsClosed(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_isclosed
-----------
 true
```

ST_ IsCollection

ST_ IsCollection mengembalikan true jika geometri input adalah salah satu subtipe berikut:GEOMETRYCOLLECTION,, MULTIPOINTMULTILINESTRING, atau. MULTIPOLYGON

Sintaksis

```
ST_IsCollection(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon adalah koleksi.

```
SELECT ST_IsCollection(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_iscollection
-----------
 false
```

ST_ IsEmpty

ST_ IsEmpty mengembalikan true jika geometri input kosong. Geometri tidak kosong jika berisi setidaknya satu titik yang tidak kosong.

ST_ IsEmpty mengembalikan true jika geometri input memiliki setidaknya satu titik tidak kosong.

Sintaksis

```
ST_IsEmpty(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon yang ditentukan kosong.

```
SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_isempty
-----------
 false
```

ST_ IsRing

ST_ IsRing mengembalikan true jika linestring input adalah cincin. Linestring adalah cincin jika tertutup dan sederhana.

Sintaksis

```
ST_IsRing(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Geometri harus aLINESTRING.

Jenis pengembalian

BOOLEAN

Jika geom bukan aLINESTRING, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah linestring yang ditentukan adalah cincin.

```
SELECT ST_IsRing(ST_GeomFromText('linestring(0 0, 1 1, 1 2, 0 0)'));
```

```
st_isring
-----------
 true
```

ST_ IsSimple

ST_ IsSimple mengembalikan true jika proyeksi 2D dari geometri input sederhana. Untuk informasi lebih lanjut tentang definisi geometri sederhana, lihatKesederhanaan geometris.

Sintaksis

```
ST_IsSimple(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah linestring yang ditentukan sederhana. Dalam contoh ini, ini tidak sederhana karena memiliki persimpangan diri.

```
SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(0 0,10 0,5 5,5 -5)'));
```

```
 st_issimple
-----------
 false
```

ST_ IsValid

ST_ IsValid mengembalikan true jika proyeksi 2D dari geometri input valid. Untuk informasi selengkapnya tentang definisi geometri yang valid, lihatValiditas geometris.

Sintaksis

```
ST_IsValid(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon yang ditentukan valid. Dalam contoh ini, poligon tidak valid karena bagian dalam poligon tidak hanya terhubung.

```
SELECT ST_IsValid(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(5 0,10 5,5 10,0
 5,5 0))'));
```

```
 st_isvalid
-----------
 false
```

ST_panjang

Untuk geometri linier, ST_length mengembalikan panjang Cartesian dari proyeksi 2D. Satuan panjang sama dengan satuan di mana koordinat geometri input dinyatakan. Fungsi mengembalikan nol (0) untuk titik, multipoint, dan geometri areal. Ketika input adalah kumpulan geometri, fungsi mengembalikan jumlah panjang geometri dalam koleksi.

Untuk geografi, ST_length mengembalikan panjang geodesik dari proyeksi 2D dari geografi linier input yang dihitung pada spheroid yang ditentukan oleh SRID. Satuan panjangnya dalam meter. Fungsi mengembalikan nol (0) untuk titik, multipoint, dan geografi areal. Ketika input adalah kumpulan geometri, fungsi mengembalikan jumlah panjang geografi dalam koleksi.

Sintaksis

```
ST_Length(geo)
```

Argumen

geografis

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

Jenis pengembalian

DOUBLE PRECISION

Jika geo adalah null, maka null dikembalikan.

Jika nilai SRID tidak ditemukan, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan panjang Cartesian dari multilinestring.

```
SELECT ST_Length(ST_GeomFromText('MULTILINESTRING((0 0,10 0,0 10),(10 0,20 0,20
 10))'));
```

```
st_length
--------------------------------
  44.142135623731
```

SQL berikut mengembalikan panjang linestring dalam geografi.

```
SELECT ST_Length(ST_GeogFromText('SRID=4326;LINESTRING(5 0,6 0,4 0)'));
```

```
 st_length
------------------
 333958.472379804
```

SQL berikut mengembalikan panjang titik dalam geografi.

```
SELECT ST_Length(ST_GeogFromText('SRID=4326;POINT(4 5)'));
```

```
 st_length
-----------
 0
```

ST_ LengthSphere

ST_ LengthSphere mengembalikan panjang geometri linier dalam meter. Untuk geometri titik, multititik, dan areal, ST_ LengthSphere mengembalikan 0. Untuk koleksi geometri, ST_ LengthSphere mengembalikan panjang total geometri linier dalam koleksi dalam meter.

ST_ men LengthSphere afsirkan koordinat setiap titik geometri input sebagai bujur dan lintang dalam derajat. Untuk geometri 3DZ, 3DM, atau 4D, hanya dua koordinat pertama yang digunakan.

Sintaksis

```
ST_LengthSphere(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONpanjang dalam meter. Perhitungan panjang didasarkan pada model bola Bumi yang radiusnya adalah radius rata-rata Bumi dari Sistem Geodesi Dunia (WGS) 84 model ellipsoidal Bumi.

Jika geom adalah null, maka null dikembalikan.

Contoh

Contoh berikut SQL menghitung panjang linestring dalam meter.

```
SELECT ST_LengthSphere(ST_GeomFromText('LINESTRING(10 10,45 45)'));
```

```
 st_lengthsphere
------------------
 5127736.08292556
```

ST_panjang2d

ST_Length2d adalah alias untuk ST_length. Untuk informasi selengkapnya, lihat ST_panjang.

ST_ LineFromMultiPoint

ST_ LineFromMultiPoint mengembalikan linestring dari geometri multipoint input. Urutan poin dipertahankan. Pengidentifikasi sistem referensi spasial (SRID) dari geometri yang dikembalikan sama dengan geometri input. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_LineFromMultiPoint(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus MULTIPOINT

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka kosong LINESTRING dikembalikan.

Jika geom berisi titik kosong, maka titik-titik kosong ini diabaikan.

Jika geom bukan aMULTIPOINT, maka kesalahan dikembalikan.

Contoh

SQL berikut membuat linestring dari multipoint.

```
SELECT ST_AsEWKT(ST_LineFromMultiPoint(ST_GeomFromText('MULTIPOINT(0 0,10 0,10 10,5 5,0
 5)',4326)));
```

```
 st_asewkt
-------------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 5,0 5)
```

ST_ LineInterpolatePoint

ST_ LineInterpolatePoint mengembalikan titik sepanjang garis pada jarak pecahan dari awal garis.

Untuk menentukan kesetaraan titik, ST_ LineInterpolatePoint beroperasi pada proyeksi 2D dari geometri input. Jika geometri input kosong, salinannya dikembalikan dalam dimensi yang sama

dengan input. Untuk geometri 3DZ, 3DM, dan 4D, `m` koordinat `z` or adalah rata-rata `m` koordinat `z` atau dari segmen tempat titik berada.

Sintaksis

```
ST_LineInterpolatePoint(geom, fraction)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe. Subtipenya adalah`LINESTRING`.

fraksi

Nilai tipe data `DOUBLE PRECISION` yang mewakili posisi titik di sepanjang linestring untuk garis. Nilainya adalah pecahan dalam kisaran 0—1, inklusif.

Jenis pengembalian

`GEOMETRY`dari subtipe`POINT`.

Jika geom atau fraksi adalah null, maka null dikembalikan.

Jika geom kosong, maka titik kosong dikembalikan.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika pec ahan berada di luar jangkauan, maka kesalahan dikembalikan.

Jika geom bukan linestring, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan titik setengah sepanjang linestring.

```
SELECT ST_AsEWKT(ST_LineInterpolatePoint(ST_GeomFromText('LINESTRING(0 0, 5 5, 7 7, 10
 10)'), 0.50));
```

```
st_asewkt
-----------
 POINT(5 5)
```

SQL berikut mengembalikan titik 90 persen dari sepanjang linestring.

```
SELECT ST_AsEWKT(ST_LineInterpolatePoint(ST_GeomFromText('LINESTRING(0 0, 5 5, 7 7, 10
 10)'), 0.90));
```

```
st_asewkt
-----------
 POINT(9 9)
```

ST_M

ST_M mengembalikan `m` koordinat titik input.

Sintaksis

```
ST_M(point)
```

Argumen

titik

`POINT`Nilai tipe data`GEOMETRY`.

Jenis pengembalian

`DOUBLE PRECISION`nilai `m` koordinat.

Jika titik nol, maka null dikembalikan.

Jika titik adalah titik 2D atau 3DZ, maka null dikembalikan.

Jika titik adalah titik kosong, maka null dikembalikan.

Jika titik bukan a`POINT`, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan m koordinat titik dalam geometri 3DM.

```
SELECT ST_M(ST_GeomFromEWKT('POINT M (1 2 3)'));
```

```
st_m
-----------
 3
```

SQL berikut mengembalikan m koordinat titik dalam geometri 4D.

```
SELECT ST_M(ST_GeomFromEWKT('POINT ZM (1 2 3 4)'));
```

```
st_m
-----------
 4
```

ST_ MakeEnvelope

ST_ MakeEnvelope mengembalikan geometri sebagai berikut:

- Jika koordinat input menentukan titik, maka geometri yang dikembalikan adalah titik.
- Jika koordinat input menentukan garis, maka geometri yang dikembalikan adalah linestring.
- Jika tidak, geometri yang dikembalikan adalah poligon, di mana koordinat input menentukan sudut kiri bawah dan kanan atas kotak.

Jika disediakan, nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan diatur ke nilai SRID input.

Sintaksis

```
ST_MakeEnvelope(xmin, ymin, xmax, ymax)
```

```
ST_MakeEnvelope(xmin, ymin, xmax, ymax, srid)
```

Argumen

xmin

Nilai tipe dataDOUBLE PRECISION. Nilai ini adalah koordinat pertama dari sudut kiri bawah kotak.

ymin

Nilai tipe dataDOUBLE PRECISION. Nilai ini adalah koordinat kedua dari sudut kiri bawah kotak.

xmax

Nilai tipe dataDOUBLE PRECISION. Nilai ini adalah koordinat pertama dari sudut kanan atas kotak.

ymax

Nilai tipe dataDOUBLE PRECISION. Nilai ini adalah koordinat kedua dari sudut kanan atas kotak.

menjijikkan

Nilai tipe data INTEGER yang mewakili pengidentifikasi sistem referensi spasial (SRID). Jika nilai SRID tidak disediakan, maka itu diatur ke nol.

Jenis pengembalian

GEOMETRYdari subtipePOINT,LINESTRING, atauPOLYGON.

SRID dari geometri yang dikembalikan disetel ke srid atau nol jika tidak srid disetel.

Jika xmin, ymin, xmax, ymax, atau srid adalah null, maka null dikembalikan.

Jika srid negatif, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan poligon yang mewakili amplop yang ditentukan oleh empat nilai koordinat masukan.

```
SELECT ST_AsEWKT(ST_MakeEnvelope(2,4,5,7));
```

```
 st_astext
---------------
```

```
 POLYGON((2 4,2 7,5 7,5 4,2 4))
```

SQL berikut mengembalikan poligon yang mewakili amplop yang ditentukan oleh empat nilai koordinat masukan dan nilai SRID.

```
SELECT ST_AsEWKT(ST_MakeEnvelope(2,4,5,7,4326));
```

```
 st_astext
---------------------------------
 SRID=4326;POLYGON((2 4,2 7,5 7,5 4,2 4))
```

ST_ MakeLine

ST_ membuat MakeLine linestring dari geometri input.

Dimensi geometri yang dikembalikan sama dengan geometri input. Kedua geometri input harus memiliki dimensi yang sama.

Sintaksis

```
ST_MakeLine(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipe harusPOINT,LINESTRING, atauMULTIPOINT.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipe harusPOINT,LINESTRING, atauMULTIPOINT.

Jenis pengembalian

GEOMETRYdari subtipeLINESTRING.

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 adalah titik kosong atau berisi titik kosong, maka titik-titik kosong ini diabaikan.

Jika geom1 dan geom2 kosong, maka kosong dikembalikan. LINESTRING

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom1 dan geom2 memiliki nilai SRID yang berbeda, maka kesalahan dikembalikan.

Jika geom1 atau geom2 bukanPOINT,, atau LINESTRINGMULTIPOINT, maka kesalahan dikembalikan.

Jika geom1 dan geom2 memiliki dimensi yang berbeda, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun linestring dari dua linestring input.

```
SELECT ST_MakeLine(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'), ST_GeomFromText('LINESTRING(88.29 39.07,88.42 39.26,88.27
 39.31,88.29 39.07)'));
```

```
st_makeline
-----------
 0102000000080000000C3F5285C8F52534052B81E85EB113D407B14AE47E15A5340C3F5285C8F423D40E17A14AE4751
```

ST_ MakePoint

ST_ MakePoint mengembalikan geometri titik yang nilai koordinatnya adalah nilai input.

Sintaksis

```
ST_MakePoint(x, y)
```

```
ST_MakePoint(x, y, z)
```

```
ST_MakePoint(x, y, z, m)
```

Argumen

x

Nilai tipe data yang DOUBLE PRECISION mewakili koordinat pertama.

y

Nilai tipe data yang DOUBLE PRECISION mewakili koordinat kedua.

z

Nilai tipe data yang DOUBLE PRECISION mewakili koordinat ketiga.

m

Nilai tipe data yang DOUBLE PRECISION mewakili koordinat keempat.

Jenis pengembalian

GEOMETRYdari subtipePOINT.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan diatur ke 0.

Jika x, y, z, atau m adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan jenis sub GEOMETRY tipe POINT dengan koordinat yang disediakan.

```
SELECT ST_AsText(ST_MakePoint(1,3));
```

```
st_astext
-----------
 POINT(1 3)
```

SQL berikut mengembalikan jenis sub GEOMETRY tipe POINT dengan koordinat yang disediakan.

```
SELECT ST_AsEWKT(ST_MakePoint(1, 2, 3));
```

```
st_asewkt
----------------
```

```
 POINT Z (1 2 3)
```

SQL berikut mengembalikan jenis sub GEOMETRY tipe POINT dengan koordinat yang disediakan.

```
SELECT ST_AsEWKT(ST_MakePoint(1, 2, 3, 4));
```

```
st_asewkt
-------------------
 POINT ZM (1 2 3 4)
```

ST_ MakePolygon

ST_ MakePolygon memiliki dua varian yang mengembalikan poligon. Satu mengambil satu geometri, dan yang lain mengambil dua geometri.

- Input dari varian pertama adalah linestring yang mendefinisikan cincin luar dari poligon keluaran.
- Input dari varian kedua adalah linestring dan multilinestring. Keduanya kosong atau tertutup.

  Batas cincin eksterior poligon keluaran adalah garis input, dan batas-batas cincin interior poligon adalah garis pada multilinestring input. Jika linestring input kosong, poligon kosong dikembalikan. Garis kosong di multilinestring diabaikan. Pengidentifikasi sistem referensi spasial (SRID) dari geometri yang dihasilkan adalah SRID umum dari dua geometri input.

Dimensi geometri yang dikembalikan sama dengan geometri input. Cincin eksterior dan cincin interior harus memiliki dimensi yang sama.

Sintaksis

```
ST_MakePolygon(geom1)
```

```
ST_MakePolygon(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING Nilai linestring harus ditutup atau kosong.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus MULTILINESTRING

Jenis pengembalian

GEOMETRYdari subtipePOLYGON.

Pengidentifikasi sistem referensi spasial (SRID) dari geometri yang dikembalikan sama dengan SRID input.

Jika geom1, atau geom2 adalah null, maka null dikembalikan.

Jika geom1 bukan linestring, maka kesalahan dikembalikan.

Jika geom2 bukan multilinestring, maka kesalahan dikembalikan.

Jika geom1 tidak ditutup, maka kesalahan dikembalikan.

Jika geom1 adalah satu titik atau tidak ditutup, maka kesalahan dikembalikan.

Jika geom2 berisi setidaknya satu linestring yang memiliki satu titik atau tidak ditutup, maka kesalahan dikembalikan.

Jika geom1 dan geom2 memiliki nilai SRID yang berbeda, maka kesalahan dikembalikan.

Jika geom1 dan geom2 memiliki dimensi yang berbeda, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan poligon dari input linestring.

```
SELECT ST_AsText(ST_MakePolygon(ST_GeomFromText('LINESTRING(77.29 29.07,77.42
 29.26,77.27 29.31,77.29 29.07)')));
```

```
 st_astext
---------------
POLYGON((77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07))
```

SQL berikut membuat poligon dari linestring tertutup dan multilinestring tertutup. Linestring digunakan untuk cincin eksterior poligon. Garis pada multilinestring digunakan untuk cincin interior poligon.

```
SELECT ST_AsEWKT(ST_MakePolygon(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,0 10,0 0)'),
 ST_GeomFromText('MULTILINESTRING((1 1,1 2,2 1,1 1),(3 3,3 4,4 3,3 3))')));
```

```
 st_astext
-----------------------------------
POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1),(3 3,3 4,4 3,3 3))
```

ST_ MemSize

ST_ MemSize mengembalikan jumlah ruang memori (dalam byte) yang digunakan oleh geometri input. Ukuran ini tergantung pada representasi internal Amazon Redshift dari geometri dan dengan demikian dapat berubah jika representasi internal berubah. Anda dapat menggunakan ukuran ini sebagai indikasi ukuran relatif objek geometri di Amazon Redshift.

Sintaksis

```
ST_MemSize(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

INTEGERmewakili dimensi inheren geom.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan ukuran memori dari koleksi geometri.

```
SELECT ST_MemSize(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0
 0)),LINESTRING(20 10,20 0,10 0))')::varchar + ' bytes';
```

```
 ?column?
```

```
-----------
 172 bytes
```

ST_Mmax

ST_Mmax mengembalikan `m` koordinat maksimum dari geometri input.

Sintaksis

```
ST_MMax(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`DOUBLE PRECISION`nilai `m` koordinat maksimum.

Jika `geom` kosong, maka null dikembalikan.

Jika `geom` adalah null, maka null dikembalikan.

Jika `geom` adalah geometri 2D atau 3DZ, maka null dikembalikan.

Contoh

SQL berikut mengembalikan `m` koordinat terbesar dari linestring dalam geometri 3DM.

```
SELECT ST_MMax(ST_GeomFromEWKT('LINESTRING M (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_mmax
-----------
  8
```

SQL berikut mengembalikan `m` koordinat terbesar dari linestring dalam geometri 4D.

```
SELECT ST_MMax(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_mmax
-----------
  11
```

St_mmin

St_mmin mengembalikan `m` koordinat minimum dari geometri input.

Sintaksis

```
ST_MMin(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`DOUBLE PRECISION`nilai `m` koordinat minimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Jika geom adalah geometri 2D atau 3DZ, maka null dikembalikan.

Contoh

SQL berikut mengembalikan `m` koordinat terkecil dari linestring dalam geometri 3DM.

```
SELECT ST_MMin(ST_GeomFromEWKT('LINESTRING M (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_mmin
-----------
  2
```

SQL berikut mengembalikan `m` koordinat terkecil dari linestring dalam geometri 4D.

```
SELECT ST_MMin(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_mmin
-----------
  3
```

ST_multi

ST_Multi mengonversi geometri ke multitipe yang sesuai. Jika geometri input sudah merupakan multitipe atau kumpulan geometri, salinannya dikembalikan. Jika geometri input adalah titik, linestring, atau poligon, maka multipoint, multilinestring, atau multipoligon, masing-masing, yang berisi geometri input dikembalikan.

Sintaksis

```
ST_Multi(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRYdengan subtipeMULTIPOINT,, MULTILINESTRINGMULTIPOLYGON, atauGEOMETRYCOLLECTION.

Pengidentifikasi sistem referensi spasial (SRID) dari geometri yang dikembalikan sama dengan geometri input.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan multipoint dari masukan multipoint.

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('MULTIPOINT((1 2),(3 4))', 4326)));
```

```
    st_asewkt
-----------------------------------
  SRID=4326;MULTIPOINT((1 2),(3 4))
```

SQL berikut mengembalikan multipoint dari titik input.

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('POINT(1 2)', 4326)));
```

```
    st_asewkt
-----------------------------------
  SRID=4326;MULTIPOINT((1 2))
```

SQL berikut mengembalikan koleksi geometri dari koleksi geometri input.

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('GEOMETRYCOLLECTION(POINT(1 2),MULTIPOINT((1
 2),(3 4)))', 4326)));
```

```
    st_asewkt
-----------------------------------
  SRID=4326;GEOMETRYCOLLECTION(POINT(1 2),MULTIPOINT((1 2),(3 4)))
```

ST_NDIM

ST_NDIMS mengembalikan dimensi koordinat geometri. ST_NDIMS tidak mempertimbangkan dimensi topologi geometri. Sebaliknya, ia mengembalikan nilai konstan tergantung pada dimensi geometri.

Sintaksis

```
ST_NDims(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

`INTEGER`mewakili dimensi inheren geom.

Jika geom adalah null, maka null dikembalikan.

Nilai yang dikembalikan adalah sebagai berikut.

Nilai yang dikembalikan	Dimensi geometri input
2	2D
3	3DZ atau 3DM
4	4 D

Contoh

SQL berikut mengembalikan jumlah dimensi dari linestring 2D.

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING(0 0,1 1,2 2,0 0)'));
```

```
st_ndims
-------------
 2
```

SQL berikut mengembalikan jumlah dimensi dari linestring 3DZ.

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING Z(0 0 3,1 1 3,2 2 3,0 0 3)'));
```

```
st_ndims
-------------
 3
```

SQL berikut mengembalikan jumlah dimensi linestring 3DM.

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING M(0 0 4,1 1 4,2 2 4,0 0 4)'));
```

```
st_ndims
-------------
 3
```

SQL berikut mengembalikan jumlah dimensi dari linestring 4D.

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING ZM(0 0 3 4,1 1 3 4,2 2 3 4,0 0 3 4)'));
```

```
st_ndims
-------------
 4
```

ST_NPoin

ST_NPoints mengembalikan jumlah titik tidak kosong dalam geometri masukan atau geografi.

Sintaksis

```
ST_NPoints(geo)
```

Argumen

geo

Nilai tipe data GEOMETRY atauGEOGRAPHY, atau ekspresi yang mengevaluasi ke GEOGRAPHY tipe GEOMETRY atau.

Jenis pengembalian

INTEGER

Jika geo adalah titik kosong, maka 0 dikembalikan.

Jika geo adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan jumlah titik dalam linestring.

```
SELECT ST_NPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_npoints
-------------
 4
```

SQL berikut mengembalikan jumlah titik dalam linestring dalam geografi.

```
SELECT ST_NPoints(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
st_npoints
-------------
 4
```

ST_NRings

ST_NRings mengembalikan jumlah cincin dalam geometri input.

Sintaksis

```
ST_NRings(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

INTEGER

Jika geom adalah null, maka null dikembalikan.

Nilai yang dikembalikan adalah sebagai berikut.

Nilai yang dikembalikan	Subtipe geometri
0	Dikembalikan jika geom adalah sub POINT tipeLINESTRING ,,MULTIPOINT , atau MULTILINESTRING
Jumlah cincin.	Dikembalikan jika geom adalah subtipe POLYGON atau MULTIPOLYGON
Jumlah cincin di semua komponen	Dikembalikan jika geom adalah subtipe GEOMETRYCOLLECTION

Contoh

SQL berikut mengembalikan jumlah cincin dalam multipoligon.

```
SELECT ST_NRings(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((0 0,-10 0,0
 -10,0 0)))'));
```

```
 st_nrings
-------------
 2
```

ST_ NumGeometries

ST_ NumGeometries mengembalikan jumlah geometri dalam geometri input.

Sintaksis

```
ST_NumGeometries(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

`INTEGER`mewakili jumlah geometri dalam ge om.

Jika geom adalah null, maka null dikembalikan.

Jika geom adalah geometri kosong tunggal, maka `0` dikembalikan.

Jika geom adalah geometri tunggal yang tidak kosong, maka 1 dikembalikan.

Jika geom adalah a `GEOMETRYCOLLECTION` atau `MULTI` subtipe, maka jumlah geometri dikembalikan.

Contoh

SQL berikut mengembalikan jumlah geometri dalam input multilinestring.

```
SELECT ST_NumGeometries(ST_GeomFromText('MULTILINESTRING((0 0,1 0,0 5),(3 4,13 26))'));
```

```
st_numgeometries
-------------
 2
```

ST_ NumInteriorRings

ST_ NumInteriorRings mengembalikan jumlah cincin dalam geometri poligon input.

Sintaksis

```
ST_NumInteriorRings(geom)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

Jenis pengembalian

`INTEGER`

Jika geom adalah null, maka null dikembalikan.

Jika geom bukan poligon, maka null dikembalikan.

Contoh

SQL berikut mengembalikan jumlah cincin interior dalam poligon input.

```
SELECT ST_NumInteriorRings(ST_GeomFromText('POLYGON((0 0,100 0,100 100,0 100,0 0),(1
 1,1 5,5 1,1 1),(7 7,7 8,8 7,7 7))'));
```

```
 st_numinteriorrings
-------------
 2
```

ST_ NumPoints

ST_ NumPoints mengembalikan jumlah titik dalam geometri input.

Sintaksis

```
ST_NumPoints(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

INTEGER

Jika geom adalah null, maka null dikembalikan.

Jika geom bukan subtipeLINESTRING, maka null dikembalikan.

Contoh

SQL berikut mengembalikan jumlah titik dalam linestring input.

```
SELECT ST_NumPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
```

```
st_numpoints
-------------
4
```

SQL berikut mengembalikan null karena input geom bukan sub `LINESTRING` tipe.

```
SELECT ST_NumPoints(ST_GeomFromText('MULTIPOINT(1 2,3 4)'));
```

```
st_numpoints
-------------
```

ST_Perimeter

Untuk geometri areal input, ST_perimeter mengembalikan perimeter Cartesian (panjang batas) dari proyeksi 2D. Unit perimeter sama dengan unit di mana koordinat geometri input dinyatakan. Fungsi mengembalikan nol (0) untuk titik, multipoint, dan geometri linier. Ketika input adalah kumpulan geometri, fungsi mengembalikan jumlah perimeter geometri dalam koleksi.

Untuk geografi input, ST_perimeter mengembalikan perimeter geodesik (panjang batas) dari proyeksi 2D dari geografi areal input yang dihitung pada spheroid yang ditentukan oleh SRID. Satuan perimeter dalam meter. Fungsi mengembalikan nol (0) untuk titik, multipoint, dan geografi linier. Ketika input adalah kumpulan geometri, fungsi mengembalikan jumlah perimeter geografi dalam koleksi.

Sintaksis

```
ST_Perimeter(geo)
```

Argumen

geo

Nilai tipe data `GEOMETRY` atau`GEOGRAPHY`, atau ekspresi yang mengevaluasi ke `GEOGRAPHY` tipe `GEOMETRY` atau.

Jenis pengembalian

`DOUBLE PRECISION`

Jika geo adalah null, maka null dikembalikan.

Jika nilai SRID tidak ditemukan, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan perimeter Cartesian dari multipoligon.

```
SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20
 10,10 0)))'));
```

```
 st_perimeter
------------------------------
    68.2842712474619
```

SQL berikut mengembalikan perimeter Cartesian dari multipoligon.

```
SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20
 10,10 0)))'));
```

```
 st_perimeter
------------------------------
    68.2842712474619
```

SQL berikut mengembalikan perimeter poligon dalam geografi.

```
SELECT ST_Perimeter(ST_GeogFromText('SRID=4326;POLYGON((0 0,1 0,0 1,0 0))'));
```

```
 st_perimeter
------------------
 378790.428393693
```

SQL berikut mengembalikan perimeter linestring dalam geografi.

```
SELECT ST_Perimeter(ST_GeogFromText('SRID=4326;LINESTRING(5 0,10 0)'));
```

```
 st_perimeter
--------------
 0
```

ST_perimeter2d

ST_perimeter2d adalah alias untuk ST_perimeter. Untuk informasi selengkapnya, lihat ST_Perimeter.

Titik ST

ST_point mengembalikan geometri titik dari nilai koordinat input.

Sintaksis

```
ST_Point(x, y)
```

Argumen

x

Nilai tipe data DOUBLE PRECISION yang mewakili koordinat pertama.

y

Nilai tipe data DOUBLE PRECISION yang mewakili koordinat kedua.

Jenis pengembalian

GEOMETRYdari subtipePOINT.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan diatur ke 0.

Jika x atau y adalah null, maka null dikembalikan.

Contoh

SQL berikut membangun geometri titik dari koordinat input.

```
SELECT ST_AsText(ST_Point(5.0, 7.0));
```

```
st_astext
-------------
POINT(5 7)
```

st_pointn

ST_pointn mengembalikan titik dalam linestring seperti yang ditentukan oleh nilai indeks. Nilai indeks negatif dihitung mundur dari akhir linestring, sehingga -1 adalah titik terakhir.

Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_PointN(geom, index)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING

indeks

Nilai tipe data INTEGER yang mewakili indeks titik dalam linestring.

Jenis pengembalian

GEOMETRYdari subtipePOINT.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan diatur ke 0.

Jika geom atau indeks adalah null, maka null dikembalikan.

Jika indeks berada di luar jangkauan, maka null dikembalikan.

Jika geom kosong, maka null dikembalikan.

Jika geom bukan aLINESTRING, maka null dikembalikan.

Contoh

SQL berikut mengembalikan representasi teks terkenal yang diperluas (EWKT) dari enam titik LINESTRING ke GEOMETRY objek dan mengembalikan titik pada indeks 5 dari linestring.

```
SELECT ST_AsEWKT(ST_PointN(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5,0
 0)',4326), 5));
```

```
st_asewkt
-------------
 SRID=4326;POINT(0 5)
```

ST_poin

ST_points mengembalikan geometri multipoint yang berisi semua titik tidak kosong dalam geometri input. ST_points tidak menghapus titik yang diduplikasi dalam input, termasuk titik awal dan akhir geometri cincin.

Sintaksis

```
ST_Points(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRYdari subtipeMULTIPOINT.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan sama dengan geom.

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka multipoint kosong dikembalikan.

Contoh

Contoh SQL berikut membangun geometri multipoint dari geometri input. Hasilnya adalah geometri multipoint yang berisi titik-titik tidak kosong dalam geometri input.

```
SELECT ST_AsEWKT(ST_Points(ST_SetSRID(ST_GeomFromText('LINESTRING(1 0,2 0,3 0)'),
 4326)));
```

```
st_asewkt
-------------
SRID=4326;MULTIPOINT((1 0),(2 0),(3 0))
```

```
SELECT ST_AsEWKT(ST_Points(ST_SetSRID(ST_GeomFromText('MULTIPOLYGON(((0 0,1 0,0 1,0
 0)))'), 4326)));
```

```
st_asewkt
-------------
SRID=4326;MULTIPOINT((0 0),(1 0),(0 1),(0 0))
```

ST_poligon

ST_Polygon mengembalikan geometri poligon yang cincin luarnya adalah garis input dengan nilai yang dimasukkan untuk pengenal sistem referensi spasial (SRID).

Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_Polygon(linestring, srid)
```

Argumen

linestring

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipe harus LINESTRING mewakili linestring. Nilai linestring harus ditutup.

menjijikkan

Nilai tipe data `INTEGER` yang mewakili SRID.

Jenis pengembalian

`GEOMETRY`dari subtipe`POLYGON`.

Nilai SRID dari geometri yang dikembalikan diatur ke s rid.

Jika linestring atau srid adalah null, maka null dikembalikan.

Jika linestring bukan linestring, maka kesalahan dikembalikan.

Jika linestring tidak ditutup, maka kesalahan dikembalikan.

Jika srid negatif, maka kesalahan dikembalikan.

Contoh

SQL berikut membangun poligon dengan nilai SRID.

```
SELECT ST_AsEWKT(ST_Polygon(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'),4356));
```

```
st_asewkt
-------------
 SRID=4356;POLYGON((77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07))
```

ST_ RemovePoint

ST_ RemovePoint mengembalikan geometri linestring yang memiliki titik geometri input pada posisi indeks dihapus.

Indeks ini berbasis nol. Pengidentifikasi sistem referensi spasial (SRID) dari hasilnya sama dengan geometri input. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_RemovePoint(geom, index)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING

indeks

Nilai tipe data INTEGER yang mewakili posisi indeks berbasis nol.

Jenis pengembalian

GEOMETRY

Jika geom atau indeks adalah null, maka null dikembalikan.

Jika geom bukan subtipeLINESTRING, maka kesalahan dikembalikan.

Jika indeks berada di luar jangkauan, maka kesalahan dikembalikan. Nilai yang valid untuk posisi indeks adalah antara 0 dan ST_NumPoints(geom) minus 1.

Contoh

SQL berikut menghapus titik terakhir dalam linestring.

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_RemovePoint(g, ST_NumPoints(g) - 1)) FROM tmp;
```

```
   st_asewkt
----------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 5)
```

ST_terbalik

ST_reverse membalikkan urutan simpul untuk geometri linier dan areal. Untuk geometri titik atau multipoint, salinan geometri asli dikembalikan. Untuk koleksi geometri, ST_reverse membalikkan urutan simpul untuk masing-masing geometri dalam koleksi.

Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_Reverse(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY

Pengidentifikasi sistem referensi spasial (SRID) dari geometri yang dikembalikan sama dengan geometri input.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut membalikkan urutan titik dalam linestring.

```
SELECT ST_AsEWKT(ST_Reverse(ST_GeomFromText('LINESTRING(1 0,2 0,3 0,4 0)', 4326)));
```

```
    st_asewkt
----------------------------------
  SRID=4326;LINESTRING(4 0,3 0,2 0,1 0)
```

ST_ SetPoint

ST_ SetPoint mengembalikan linestring dengan koordinat yang diperbarui sehubungan dengan posisi linestring input seperti yang ditentukan oleh indeks. Koordinat baru adalah koordinat titik input.

Dimensi geometri yang dikembalikan sama dengan nilai geom1. Jika geom1 dan geom2 memiliki dimensi yang berbeda, geom2 diproyeksikan ke dimensi geom 1.

Sintaksis

```
ST_SetPoint(geom1, index, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING

indeks

Nilai tipe data INTEGER yang mewakili posisi indeks. A 0 mengacu pada titik pertama dari garis linestring dari kiri, 1 mengacu pada titik kedua, dan seterusnya. Indeks bisa menjadi nilai negatif. A -1 mengacu pada titik pertama dari linestring dari kanan, -2 mengacu pada titik kedua dari linestring dari kanan, dan seterusnya.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus POINT

Jenis pengembalian

GEOMETRY

Jika geom2 adalah titik kosong, maka geom 1 dikembalikan.

Jika geom1, geom2, atau indeks adalah null, maka null dikembalikan.

Jika geom1 bukan linestring, maka kesalahan dikembalikan.

Jika indeks tidak berada dalam rentang indeks yang valid, maka kesalahan dikembalikan.

Jika geom2 bukan titik, maka kesalahan dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan linestring baru di mana titik kedua dari linestring input diatur ke titik yang ditentukan.

```
SELECT ST_AsText(ST_SetPoint(ST_GeomFromText('LINESTRING(1 2, 3 2, 5 2, 1 2)'), 2,
 ST_GeomFromText('POINT(7 9)')));
```

```
st_astext
-------------
 LINESTRING(1 2,3 2,7 9,1 2)
```

Contoh SQL berikut mengembalikan linestring baru di mana titik ketiga dari kanan (indeks negatif) dari linestring diatur ke titik yang ditentukan.

```
SELECT ST_AsText(ST_SetPoint(ST_GeomFromText('LINESTRING(1 2, 3 2, 5 2, 1 2)'), -3,
 ST_GeomFromText('POINT(7 9)')));
```

```
st_astext
-------------
 LINESTRING(1 2,7 9,5 2,1 2)
```

ST_SetsRid

ST_SetsRid mengembalikan geometri yang sama dengan geometri input, kecuali diperbarui dengan input nilai untuk pengenal sistem referensi spasial (SRID).

Sintaksis

```
ST_SetSRID(geom, srid)
```

Argumen

geom

Nilai tipe data `GEOMETRY` atau ekspresi yang mengevaluasi ke `GEOMETRY` tipe.

menjijikkan

Nilai tipe data `INTEGER` yang mewakili SRID.

Jenis pengembalian

`GEOMETRY`

Nilai SRID dari geometri yang dikembalikan diatur ke s rid.

Jika geom atau srid adalah null, maka null dikembalikan.

Jika srid negatif, maka kesalahan dikembalikan.

Contoh

SQL berikut menetapkan nilai SRID dari linestring.

```
SELECT ST_AsEWKT(ST_SetSRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'),50));
```

```
st_asewkt
-------------
 SRID=50;LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)
```

ST_Sederhanakan

ST_Simplify mengembalikan salinan geometri input yang disederhanakan menggunakan Ramer-Douglas-Peucker algoritma dengan toleransi yang diberikan. Topologi geometri input mungkin tidak dipertahankan. Untuk informasi lebih lanjut tentang algoritma, lihat Algoritma Ramer—Douglas—Peucker di Wikipedia.

Ketika ST_Simplify menghitung jarak untuk menyederhanakan geometri, ST_Simplify beroperasi pada proyeksi 2D dari geometri input.

Sintaksis

```
ST_Simplify(geom, tolerance)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

toleransi

Nilai tipe data DOUBLE PRECISION yang mewakili tingkat toleransi Ramer-Douglas-Peucker algoritma. Jika toleransi adalah angka negatif, maka nol digunakan.

Jenis pengembalian

GEOMETRY.

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Dimensi geometri yang dikembalikan sama dengan geometri input.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut menyederhanakan linestring input menggunakan toleransi jarak Euclidean 1 dengan algoritma. Ramer-Douglas-Peucker Satuan jarak sama dengan koordinat geometri.

```
SELECT ST_AsEWKT(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,1 2,1 1,2 2,2 1)'), 1));
```

```
 st_asewkt
-----------
LINESTRING(0 0,1 2,2 1)
```

ST_SRID

ST_SRID mengembalikan pengenal sistem referensi spasial (SRID) dari geometri input. Untuk informasi lebih lanjut tentang SRID, lihatMengkueri data spasial di Amazon Redshift.

Sintaksis

```
ST_SRID(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

INTEGERmewakili nilai SRID dari ge om.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan nilai SRID dari linestring yang disetel ke SRID. 4326

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)',4326));
```

```
st_srid
-------------
 4326
```

SQL berikut mengembalikan nilai SRID dari linestring yang tidak disetel saat dibangun. Ini menghasilkan 0 nilai SRID.

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_srid
-------------
 0
```

ST_ StartPoint

ST_ StartPoint mengembalikan titik pertama dari linestring input. Nilai pengidentifikasi sistem referensi spasial (SRID) dari hasilnya sama dengan geometri input. Dimensi geometri yang dikembalikan sama dengan geometri input.

Sintaksis

```
ST_StartPoint(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Subtipenya harus LINESTRING

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Jika geom kosong, maka null dikembalikan.

Jika geom bukan aLINESTRING, maka null dikembalikan.

Contoh

SQL berikut mengembalikan representasi teks terkenal yang diperluas (EWKT) dari empat titik LINESTRING ke GEOMETRY objek dan mengembalikan titik awal dari linestring.

```
SELECT ST_AsEWKT(ST_StartPoint(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0
 5)',4326)));
```

```
st_asewkt
-------------
 SRID=4326;POINT(0 0)
```

ST_sentuhan

ST_Touches mengembalikan true jika proyeksi 2D dari dua geometri input bersentuhan. Kedua geometri bersentuhan jika tidak kosong, berpotongan, dan tidak memiliki titik interior yang sama.

Sintaksis

```
ST_Touches(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon menyentuh linestring.

```
SELECT ST_Touches(ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))'),
 ST_GeomFromText('LINESTRING(20 10,20 0,10 0)'));
```

```
 st_touches
-------------
 t
```

ST_Transform

ST_transform mengembalikan geometri baru dengan koordinat yang diubah dalam sistem referensi spasial yang ditentukan oleh input pengidentifikasi sistem referensi spasial (SRID).

Sintaksis

```
ST_Transform(geom, srid)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

menjijikkan

Nilai tipe data INTEGER yang mewakili SRID.

Jenis pengembalian

GEOMETRY.

Nilai SRID dari geometri yang dikembalikan diatur ke s rid.

Jika geom atau srid adalah null, maka null dikembalikan.

Jika nilai SRID yang terkait dengan geom input tidak ada, maka kesalahan dikembalikan.

Jika srid tidak ada, maka kesalahan dikembalikan.

Contoh

SQL berikut mengubah SRID dari koleksi geometri kosong.

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('GEOMETRYCOLLECTION EMPTY', 3857),
 4326));
```

```
            st_asewkt
-----------------------------------
 SRID=4326;GEOMETRYCOLLECTION EMPTY
```

SQL berikut mengubah SRID dari linestring.

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9,
 -22 -33)', 4326), 26918));
```

```
    st_asewkt
-----------------------------------------------------------------------------------------
 SRID=26918;LINESTRING(73106.6977300955 15556182.9688576,14347201.5059964
 1545178.32934967,1515090.41262989 9522193.25115316,10491250.83295
 2575457.28410878,5672303.72135968 -5233682.61176205)
```

SQL berikut mengubah SRID dari poligon.

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('POLYGON Z ((-10 10 -7, -65 10 -6, -10 64
 -5, -10 10 -7), (-11 11 5, -11 12 6, -12 11 7, -11 11 5))', 6989), 6317));
```

```
                                        st_asewkt
-------------------------------------------------------------------------------------------------
 SRID=6317;POLYGON Z ((6186430.2771091 -1090834.57212608
 1100247.33216237,2654831.67853801 -5693304.90741276 1100247.50581055,2760987.41750022
 -486836.575101877 5709710.44137268,6186430.2771091 -1090834.57212608
 1100247.33216237),(6146675.25029258 -1194792.63532103 1209007.1115113,6125027.87562215
 -1190584.81194058 1317403.77865723,6124888.99555252 -1301885.3455052
 1209007.49312929,6146675.25029258 -1194792.63532103 1209007.1115113))
```

ST_Union

ST_Union mengembalikan geometri yang mewakili penyatuan dua geometri. Artinya, ia menggabungkan geometri input untuk menghasilkan geometri yang dihasilkan tanpa tumpang tindih.

Sintaksis

```
ST_Union(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

GEOMETRY

Nilai pengenal sistem referensi spasial (SRID) dari geometri yang dikembalikan adalah nilai SRID dari geometri input.

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 atau geom2 kosong, maka geometri kosong dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai yang sama untuk pengenal sistem referensi spasial (SRID), maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, linestring, atau multilinestring, maka kesalahan dikembalikan.

Jika geom1 atau geom2 bukan geometri dua dimensi (2D), maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan geometri non-kosong yang mewakili penyatuan dua geometri input.

```
SELECT ST_AsEWKT(ST_Union(ST_GeomFromText('POLYGON((0 0,100 100,0 200,0 0))'),
 ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))')));
```

```
        st_asewkt
-------------------------
 POLYGON((0 0,0 200,100 100,5 5,10 0,0 0))
```

ST_dalam

ST_within mengembalikan true jika proyeksi 2D dari geometri input pertama berada dalam proyeksi 2D dari geometri input kedua.

Misalnya, geometri A berada dalam geometri B jika setiap titik di dalam A adalah titik di dalam B dan interiornya memiliki persimpangan yang tidak kosong.

ST_within (A,B) setara dengan ST_contains (,). B A

Sintaksis

```
ST_Within(geom1, geom2)
```

Argumen

geom1

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe. Nilai ini dibandingkan dengan geom2 untuk menentukan apakah itu dalam geom2.

geom2

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 dan geom2 tidak memiliki nilai pengenal sistem referensi spasial (SRID) yang sama, maka kesalahan dikembalikan.

Jika geom1 atau geom2 adalah kumpulan geometri, maka kesalahan dikembalikan.

Contoh

SQL berikut memeriksa apakah poligon pertama berada dalam poligon kedua.

```
SELECT ST_Within(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_within
-----------
 true
```

ST_X

ST_X mengembalikan koordinat pertama dari titik input.

Sintaksis

```
ST_X(point)
```

Argumen

titik

POINTNilai tipe dataGEOMETRY.

Jenis pengembalian

DOUBLE PRECISIONnilai koordinat pertama.

Jika titik nol, maka null dikembalikan.

Jika titik adalah titik kosong, maka null dikembalikan.

Jika titik bukan aPOINT, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan koordinat pertama dari suatu titik.

```
SELECT ST_X(ST_Point(1,2));
```

```
st_x
-----------
 1.0
```

ST_xmax

ST_Xmax mengembalikan koordinat pertama maksimum dari geometri input.

Sintaksis

```
ST_XMax(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONnilai koordinat pertama maksimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan koordinat pertama terbesar dari linestring.

```
SELECT ST_XMax(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_xmax
-----------
 77.42
```

st_xmin

St_xmin mengembalikan koordinat pertama minimum dari geometri input.

Sintaksis

```
ST_XMin(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONnilai koordinat pertama minimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan koordinat pertama terkecil dari linestring.

```
SELECT ST_XMin(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_xmin
-----------
 77.27
```

ST_Y

ST_Y mengembalikan koordinat kedua dari titik input.

Sintaksis

```
ST_Y(point)
```

Argumen

titik

`POINT`Nilai tipe data`GEOMETRY`.

Jenis pengembalian

`DOUBLE PRECISION`nilai koordinat kedua.

Jika titik nol, maka null dikembalikan.

Jika titik adalah titik kosong, maka null dikembalikan.

Jika titik bukan a`POINT`, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan koordinat kedua dari suatu titik.

```
SELECT ST_Y(ST_Point(1,2));
```

```
st_y
-----------
 2.0
```

ST_ymax

St_ymax mengembalikan koordinat kedua maksimum dari geometri input.

Sintaksis

```
ST_YMax(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONnilai koordinat kedua maksimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan koordinat kedua terbesar dari linestring.

```
SELECT ST_YMax(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_ymax
-----------
 29.31
```

st_ymin

St_ymin mengembalikan koordinat kedua minimum dari geometri input.

Sintaksis

```
ST_YMin(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONnilai koordinat kedua minimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan koordinat kedua terkecil dari linestring.

```
SELECT ST_YMin(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_ymin
-----------
 29.07
```

ST_Z

ST_Z mengembalikan z koordinat titik input.

Sintaksis

```
ST_Z(point)
```

Argumen

titik

POINTNilai tipe dataGEOMETRY.

Jenis pengembalian

`DOUBLE PRECISION`nilai `m` koordinat.

Jika titik nol, maka null dikembalikan.

Jika titik adalah titik 2D atau 3DM, maka null dikembalikan.

Jika titik adalah titik kosong, maka null dikembalikan.

Jika titik bukan a`POINT`, maka kesalahan dikembalikan.

Contoh

SQL berikut mengembalikan z koordinat titik dalam geometri 3DZ.

```
SELECT ST_Z(ST_GeomFromEWKT('POINT Z (1 2 3)'));
```

```
st_z
-----------
 3
```

SQL berikut mengembalikan z koordinat titik dalam geometri 4D.

```
SELECT ST_Z(ST_GeomFromEWKT('POINT ZM (1 2 3 4)'));
```

```
st_z
-----------
 3
```

ST_Zmax

ST_Zmax mengembalikan z koordinat maksimum dari geometri input.

Sintaksis

```
ST_ZMax(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONnilai z koordinat maksimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Jika geom adalah geometri 2D atau 3DM, maka null dikembalikan.

Contoh

SQL berikut mengembalikan z koordinat terbesar dari linestring dalam geometri 3DZ.

```
SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING Z (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_zmax
-----------
  8
```

SQL berikut mengembalikan z koordinat terbesar dari linestring dalam geometri 4D.

```
SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_zmax
-----------
  10
```

St_zmin

ST_Zmin mengembalikan z koordinat minimum dari geometri input.

Sintaksis

```
ST_ZMin(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

DOUBLE PRECISIONnilai z koordinat minimum.

Jika geom kosong, maka null dikembalikan.

Jika geom adalah null, maka null dikembalikan.

Jika geom adalah geometri 2D atau 3DM, maka null dikembalikan.

Contoh

SQL berikut mengembalikan z koordinat terkecil dari linestring dalam geometri 3DZ.

```
SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING Z (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_zmin
-----------
  2
```

SQL berikut mengembalikan z koordinat terkecil dari linestring dalam geometri 4D.

```
SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_zmin
-----------
```

```
2
```

DukunganBBox

supportsBbox mengembalikan true jika geometri input mendukung pengkodean dengan kotak pembatas yang telah dihitung sebelumnya. Untuk informasi selengkapnya tentang dukungan untuk kotak pembatas, lihatKotak pembatas.

Sintaksis

```
SupportsBBox(geom)
```

Argumen

geom

Nilai tipe data GEOMETRY atau ekspresi yang mengevaluasi ke GEOMETRY tipe.

Jenis pengembalian

BOOLEAN

Jika geom adalah null, maka null dikembalikan.

Contoh

SQL berikut mengembalikan true karena geometri titik input mendukung dikodekan dengan kotak pembatas.

```
SELECT SupportsBBox(AddBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
supportsbbox
--------------
t
```

SQL berikut mengembalikan false karena geometri titik input tidak mendukung dikodekan dengan kotak pembatas.

```
SELECT SupportsBBox(DropBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
supportsbbox
--------------
f
```

Fungsi string

Topik

- || (Penggabungan) operator
- Fungsi ASCII
- Fungsi BPCHARCMP
- Fungsi BTRIM
- Fungsi BTTEXT_PATTERN_CMP
- Fungsi CHAR_LENGTH
- Fungsi CHARACTER_LENGTH
- Fungsi CHARINDEX
- Fungsi CHR
- Fungsi COLLATE
- Fungsi CONCAT
- Fungsi CRC32
- Fungsi PERBEDAAN
- Fungsi INITCAP
- Fungsi KIRI dan KANAN
- Fungsi LEN
- Fungsi LENGTH
- Fungsi LOWER
- Fungsi LPAD dan RPAD
- Fungsi LTRIM
- Fungsi OCTETINDEX

- Fungsi OCTET_LENGTH
- Fungsi POSISI
- Fungsi QUOTE_IDENT
- Fungsi QUOTE_LITERAL
- Fungsi REGEXP_COUNT
- Fungsi REGEXP_INSTR
- Fungsi REGEXP_REPLACE
- Fungsi REGEXP_SUBSTR
- Fungsi REPEAT
- Fungsi REPLACE
- Fungsi REPLIKA
- Fungsi REVERSE
- Fungsi RTRIM
- Fungsi SOUNDEX
- Fungsi SPLIT_PART
- Fungsi STRPOS
- Fungsi STRTOL
- Fungsi SUBSTRING
- Fungsi TEXTLEN
- Fungsi TRANSLATE
- Fungsi TRIM
- Fungsi ATAS

Fungsi string memproses dan memanipulasi string karakter atau ekspresi yang mengevaluasi string karakter. Ketika argumen string dalam fungsi ini adalah nilai literal, itu harus ditutup dalam tanda kutip tunggal. Jenis data yang didukung termasuk CHAR dan VARCHAR.

Bagian berikut menyediakan nama fungsi, sintaks, dan deskripsi untuk fungsi yang didukung. Semua offset ke dalam string adalah berbasis satu.

Fungsi hanya node pemimpin yang tidak digunakan lagi

Fungsi string berikut tidak digunakan lagi karena hanya berjalan pada node pemimpin. Untuk informasi selengkapnya, lihat Node—hanya fungsi utama

- DAPAT_BYTE
- SET_BIT
- SET_BYTE
- TO_ASCII

|| (Penggabungan) operator

Menggabungkan dua ekspresi di kedua sisi `||` simbol dan mengembalikan ekspresi gabungan.

Mirip denganFungsi CONCAT.

> **Note**
>
> Jika salah satu atau kedua ekspresi adalah null, hasil penggabungan adalah. `NULL`

Sintaksis

```
expression1 || expression2
```

Argumen

Ekspresi1

Sebuah `CHAR` string, `VARCHAR` string, ekspresi biner, atau ekspresi yang mengevaluasi salah satu jenis ini.

ekspressi2

Sebuah `CHAR` string, `VARCHAR` string, ekspresi biner, atau ekspresi yang mengevaluasi salah satu jenis ini.

Jenis pengembalian

Jenis pengembalian string sama dengan jenis argumen input. Misalnya, menggabungkan dua string tipe `VARCHAR` mengembalikan string tipe. `VARCHAR`

Contoh

Contoh berikut menggunakan tabel USERS dan VENUE dari database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menggabungkan bidang FIRSTNAME dan LASTNAME dari tabel USERS dalam database sampel, gunakan contoh berikut.

```
SELECT (firstname || ' ' || lastname) as fullname
FROM users
ORDER BY 1
LIMIT 10;

+-----------------+
|    fullname     |
+-----------------+
| Aaron Banks     |
| Aaron Booth     |
| Aaron Browning  |
| Aaron Burnett   |
| Aaron Casey     |
| Aaron Cash      |
| Aaron Castro    |
| Aaron Dickerson |
| Aaron Dixon     |
| Aaron Dotson    |
+-----------------+
```

Untuk menggabungkan kolom yang mungkin berisi nol, gunakan ekspresi. Fungsi NVL dan COALESCE Contoh berikut menggunakan NVL untuk mengembalikan `0` kapan pun `NULL` ditemui.

```
SELECT (venuename || ' seats ' || NVL(venueseats, 0)) as seating
FROM venue
WHERE venuestate = 'NV' or venuestate = 'NC'
ORDER BY 1
LIMIT 10;

+-------------------------------------+
|               seating               |
+-------------------------------------+
| Ballys Hotel seats 0                |
| Bank of America Stadium seats 73298 |
| Bellagio Hotel seats 0              |
```

```
| Caesars Palace seats 0              |
| Harrahs Hotel seats 0               |
| Hilton Hotel seats 0                |
| Luxor Hotel seats 0                 |
| Mandalay Bay Hotel seats 0          |
| Mirage Hotel seats 0                |
| New York New York seats 0           |
+-------------------------------------+
```

Fungsi ASCII

Fungsi ASCII mengembalikan kode ASCII, atau titik kode Unicode, dari karakter pertama dalam string yang Anda tentukan. Fungsi kembali `0` jika string kosong. Ini kembali `NULL` jika string adalah null.

Sintaksis

```
ASCII('string')
```

Pendapat

tali

Sebuah `CHAR` string atau `VARCHAR` string.

Jenis pengembalian

INTEGER

Contoh

Untuk kembali`NULL`, gunakan contoh berikut. Fungsi NULLIF kembali `NULL` jika kedua argumennya sama, jadi argumen input untuk fungsi ASCII adalah. `NULL` Untuk informasi selengkapnya, lihat Fungsi NULLIF.

```
SELECT ASCII(NULLIF('',''));

+-------+
| ascii |
+-------+
|  NULL |
+-------+
```

Untuk mengembalikan kode ASCII 0, gunakan contoh berikut.

```
SELECT ASCII('');

+-------+
| ascii |
+-------+
|     0 |
+-------+
```

Untuk mengembalikan kode ASCII 97 untuk huruf pertama dari kata amazon, gunakan contoh berikut.

```
SELECT ASCII('amazon');

+-------+
| ascii |
+-------+
|    97 |
+-------+
```

Untuk mengembalikan kode ASCII 65 untuk huruf pertama dari kata Amazon, gunakan contoh berikut.

```
SELECT ASCII('Amazon');

+-------+
| ascii |
+-------+
|    65 |
+-------+
```

Fungsi BPCHARCMP

Membandingkan nilai dua string dan mengembalikan bilangan bulat. Jika string identik, fungsi kembali`0`. Jika string pertama lebih besar menurut abjad, fungsi kembali`1`. Jika string kedua lebih besar, fungsi kembali`-1`.

Untuk karakter multibyte, perbandingan didasarkan pada pengkodean byte.

Sinonim dari[Fungsi BTTEXT_PATTERN_CMP](#).

Sintaksis

```
BPCHARCMP(string1, string2)
```

Argumen

string1

Sebuah CHAR string atau VARCHAR string.

senar2

Sebuah CHAR string atau VARCHAR string.

Jenis pengembalian

INTEGER

Contoh

Contoh berikut menggunakan tabel USERS dari database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menentukan apakah nama depan pengguna menurut abjad lebih besar dari nama belakang pengguna untuk sepuluh entri pertama dalam tabel USERS, gunakan contoh berikut. Untuk entri di mana string untuk FIRSTNAME lebih lambat menurut abjad daripada string untuk LASTNAME, fungsi kembali. 1 Jika LASTNAME menurut abjad lebih lambat dari FIRSTNAME, fungsi kembali. -1

```
SELECT userid, firstname, lastname, BPCHARCMP(firstname, lastname)
FROM users
ORDER BY 1, 2, 3, 4
LIMIT 10;

+--------+-----------+-----------+-----------+
| userid | firstname | lastname  | bpcharcmp |
+--------+-----------+-----------+-----------+
|      1 | Rafael    | Taylor    |        -1 |
|      2 | Vladimir  | Humphrey  |         1 |
|      3 | Lars      | Ratliff   |        -1 |
|      4 | Barry     | Roy       |        -1 |
|      5 | Reagan    | Hodge     |         1 |
|      6 | Victor    | Hernandez |         1 |
```

```
|      7 | Tamekah   | Juarez    |         1 |
|      8 | Colton    | Roy       |        -1 |
|      9 | Mufutau   | Watkins   |        -1 |
|     10 | Naida     | Calderon  |         1 |
+--------+-----------+-----------+-----------+
```

Untuk mengembalikan semua entri dalam tabel USERS tempat fungsi kembali0, gunakan contoh berikut. Fungsi kembali 0 ketika FIRSTNAME identik dengan LASTNAME.

```
SELECT userid, firstname, lastname,
BPCHARCMP(firstname, lastname)
FROM users
WHERE BPCHARCMP(firstname, lastname)=0
ORDER BY 1, 2, 3, 4;

+--------+-----------+----------+-----------+
| userid | firstname | lastname | bpcharcmp |
+--------+-----------+----------+-----------+
|     62 | Chase     | Chase    |         0 |
|   4008 | Whitney   | Whitney  |         0 |
|  12516 | Graham    | Graham   |         0 |
|  13570 | Harper    | Harper   |         0 |
|  16712 | Cooper    | Cooper   |         0 |
|  18359 | Chase     | Chase    |         0 |
|  27530 | Bradley   | Bradley  |         0 |
|  31204 | Harding   | Harding  |         0 |
+--------+-----------+----------+-----------+
```

Fungsi BTRIM

Fungsi BTRIM memangkas string dengan menghapus kosong di depan dan di belakang atau dengan menghapus karakter depan dan belakang yang cocok dengan string yang ditentukan opsional.

Sintaksis

```
BTRIM(string [, trim_chars ] )
```

Argumen

tali

String masukan VARCHAR yang akan dipangkas.

trim_chars

String VARCHAR berisi karakter yang akan dicocokkan.

Jenis pengembalian

Fungsi BTRIM mengembalikan string VARCHAR.

Contoh

Contoh berikut memangkas bagian depan dan belakang kosong dari string: `'  abc  '`

```
select '     abc    ' as untrim, btrim('     abc    ') as trim;

untrim    | trim
----------+------
   abc    | abc
```

Contoh berikut menghapus string depan dan belakang dari `'xyz'` string`'xyzaxyzbxyzcxyz'`. Kemunculan di depan dan di belakang `'xyz'` dihapus, tetapi kejadian yang internal di dalam string tidak dihapus.

```
select 'xyzaxyzbxyzcxyz' as untrim,
btrim('xyzaxyzbxyzcxyz', 'xyz') as trim;

     untrim      |   trim
-----------------+-----------
 xyzaxyzbxyzcxyz | axyzbxyzc
```

Contoh berikut menghapus bagian depan dan belakang dari string `'setuphistorycassettes'` yang cocok dengan salah satu karakter dalam daftar trim_ chars. `'tes'` Setiap`t`,`e`, atau `s` yang terjadi sebelum karakter lain yang tidak ada dalam daftar trim_chars di awal atau akhir string input dihapus.

```
SELECT btrim('setuphistorycassettes', 'tes');

     btrim
-----------------
 uphistoryca
```

Fungsi BTTEXT_PATTERN_CMP

Sinonim untuk fungsi BPCHARCMP.

Lihat Fungsi BPCHARCMP untuk detail.

Fungsi CHAR_LENGTH

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi CHARACTER_LENGTH

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi CHARINDEX

Mengembalikan lokasi substring yang ditentukan dalam string.

Lihat Fungsi POSISI dan Fungsi STRPOS untuk fungsi serupa.

Sintaksis

```
CHARINDEX( substring, string )
```

Argumen

substring

Substring untuk dicari di dalam string.

senar

String atau kolom yang akan dicari.

Jenis pengembalian

INTEGER

Fungsi CHARINDEX mengembalikan yang `INTEGER` sesuai dengan posisi substring (berbasis satu, bukan berbasis nol). Posisi didasarkan pada jumlah karakter, bukan byte, sehingga

karakter multi-byte dihitung sebagai karakter tunggal. CHARINDEX kembali `0` jika substring tidak ditemukan di dalam string.

Contoh

Untuk mengembalikan posisi string `fish` dalam kata`dog`, gunakan contoh berikut.

```
SELECT CHARINDEX('fish', 'dog');

+-----------+
| charindex |
+-----------+
|         0 |
+-----------+
```

Untuk mengembalikan posisi string `fish` dalam kata`dogfish`, gunakan contoh berikut.

```
SELECT CHARINDEX('fish', 'dogfish');

+-----------+
| charindex |
+-----------+
|         4 |
+-----------+
```

Contoh berikut menggunakan tabel SALES dari database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan jumlah transaksi penjualan yang berbeda dengan komisi lebih dari 999,00 dari tabel PENJUALAN, gunakan contoh berikut. Perintah ini menghitung komisi lebih besar dari 999,00 dengan memeriksa apakah desimal lebih dari 4 tempat dari awal nilai komisi.

```
SELECT DISTINCT CHARINDEX('.', commission), COUNT (CHARINDEX('.', commission))
FROM sales
WHERE CHARINDEX('.', commission) > 4
GROUP BY CHARINDEX('.', commission)
ORDER BY 1,2;

+-----------+-------+
| charindex | count |
+-----------+-------+
```

```
|         5 |   629 |
+-----------+-------+
```

Fungsi CHR

Fungsi CHR mengembalikan karakter yang cocok dengan nilai titik kode ASCII yang ditentukan oleh parameter input.

Sintaksis

```
CHR(number)
```

Pendapat

jumlah

Parameter input adalah `INTEGER` yang mewakili nilai titik kode ASCII.

Jenis pengembalian

CHAR

Fungsi CHR mengembalikan `CHAR` string jika karakter ASCII cocok dengan nilai input. Jika nomor input tidak memiliki kecocokan ASCII, fungsi kembali`NULL`.

Contoh

Untuk mengembalikan karakter yang sesuai dengan titik kode ASCII 0, gunakan contoh berikut. Perhatikan bahwa fungsi CHR kembali `NULL` untuk input`0`.

```
SELECT CHR(0);

+-----+
| chr |
+-----+
|     |
+-----+
```

Untuk mengembalikan karakter yang sesuai dengan titik kode ASCII 65, gunakan contoh berikut.

```
SELECT CHR(65);
```

```
+-----+
| chr |
+-----+
| A   |
+-----+
```

Untuk mengembalikan nama peristiwa berbeda yang dimulai dengan huruf besar A (titik kode ASCII 65), gunakan contoh berikut. Contoh berikut menggunakan tabel EVENT dari database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

```
SELECT DISTINCT eventname FROM event
WHERE SUBSTRING(eventname, 1, 1)=CHR(65) LIMIT 5;

+----------------------+
|       eventname      |
+----------------------+
| A Catered Affair     |
| As You Like It       |
| A Man For All Seasons |
| Alan Jackson         |
| Armando Manzanero    |
+----------------------+
```

Fungsi COLLATE

Fungsi COLLATE mengesampingkan kolasi kolom string atau ekspresi.

Untuk informasi tentang cara membuat tabel menggunakan kolasi database, lihatCREATE TABLE.

Untuk informasi tentang cara membuat database menggunakan pengumpulan basis data, lihatBUAT BASIS DATA.

Sintaksis

```
COLLATE( string, 'case_sensitive' | 'cs' | 'case_insensitive' | 'ci');
```

Argumen

senar

Kolom string atau ekspresi yang ingin Anda ganti.

'case_sensitive' | 'cs' | 'case_in sensitive' | 'ci'

Konstanta string dari nama kolasi. Amazon Redshift hanya mendukung nilai berikut untuk parameter ini:

- peka kasus
- cs
- kasus_tidak sensitif
- le

case_sensitive dan cs dapat dipertukarkan dan menghasilkan hasil yang sama. Demikian pula, case_insensitive dan ci dapat dipertukarkan dan menghasilkan hasil yang sama.

Jenis pengembalian

Fungsi COLLATE mengembalikanVARCHAR,CHAR, atau SUPER tergantung pada jenis ekspresi masukan pertama. Fungsi ini hanya mengubah kolasi argumen input pertama dan tidak akan mengubah nilai outputnya.

Contoh

Untuk membuat tabel T dan mendefinisikan col1 dalam tabel T sebagai`case_sensitive`, gunakan contoh berikut.

```
CREATE TABLE T ( col1 Varchar(20) COLLATE case_sensitive );

INSERT INTO T VALUES ('john'),('JOHN');
```

Saat Anda menjalankan kueri pertama, Amazon Redshift hanya kembali`john`. Setelah fungsi COLLATE berjalan pada col1, kolasi menjadi. `case_insensitive` Kueri kedua mengembalikan keduanya `john` dan`JOHN`.

```
SELECT * FROM T WHERE col1 = 'john';

+------+
| col1 |
+------+
| john |
+------+

SELECT * FROM T WHERE COLLATE(col1, 'case_insensitive') = 'john';
```

```
+------+
| col1 |
+------+
| john |
| JOHN |
+------+
```

Untuk membuat tabel A dan mendefinisikan col1 dalam tabel A sebagai`case_insensitive`, gunakan contoh berikut.

```
CREATE TABLE A ( col1 Varchar(20) COLLATE case_insensitive );

INSERT INTO A VALUES ('john'),('JOHN');
```

Saat Anda menjalankan kueri pertama, Amazon Redshift mengembalikan keduanya `john` dan`JOHN`. Setelah fungsi COLLATE berjalan pada col1, kolasi menjadi. `case_sensitive` Kueri kedua hanya kembali`john`.

```
SELECT * FROM A WHERE col1 = 'john';

+------+
| col1 |
+------+
| john |
| JOHN |
+------+

SELECT * FROM A WHERE COLLATE(col1, 'case_sensitive') = 'john';

+------+
| col1 |
+------+
| john |
+------+
```

Fungsi CONCAT

Fungsi CONCAT menggabungkan dua ekspresi dan mengembalikan ekspresi yang dihasilkan. Untuk menggabungkan lebih dari dua ekspresi, gunakan fungsi CONCAT bersarang. Operator gabungan (`||`) antara dua ekspresi menghasilkan hasil yang sama dengan fungsi CONCAT.

Sintaksis

```
CONCAT ( expression1, expression2 )
```

Argumen

ekspresi1, ekspresi2

Kedua argumen dapat berupa string karakter panjang tetap, string karakter panjang variabel, ekspresi biner, atau ekspresi yang mengevaluasi salah satu input ini.

Jenis pengembalian

CONCAT mengembalikan ekspresi. Tipe data ekspresi adalah tipe yang sama dengan argumen input.

Jika ekspresi input memiliki tipe yang berbeda, Amazon Redshift mencoba mengetik secara implisit cast salah satu ekspresi. Jika nilai tidak dapat diberikan, kesalahan dikembalikan.

Catatan penggunaan

- Untuk fungsi CONCAT dan operator penggabungan, jika salah satu atau kedua ekspresi adalah null, hasil penggabungan adalah null.

Contoh

Contoh berikut menggabungkan dua literal karakter:

```
SELECT CONCAT('December 25, ', '2008');

concat
-------------------
December 25, 2008
(1 row)
```

Kueri berikut, menggunakan || operator alih-alih CONCAT, menghasilkan hasil yang sama:

```
SELECT 'December 25, '||'2008';

?column?
```

```
-------------------
December 25, 2008
(1 row)
```

Contoh berikut menggunakan fungsi CONCAT bersarang di dalam fungsi CONCAT lain untuk menggabungkan tiga string karakter:

```
SELECT CONCAT('Thursday, ', CONCAT('December 25, ', '2008'));

concat
----------------------------
Thursday, December 25, 2008
(1 row)
```

Untuk menggabungkan kolom yang mungkin berisi NULL, gunakanFungsi NVL dan COALESCE, yang mengembalikan nilai yang diberikan saat menemukan NULL. Contoh berikut menggunakan NVL untuk mengembalikan 0 setiap kali NULL ditemui.

```
SELECT CONCAT(venuename, CONCAT(' seats ', NVL(venueseats, 0))) AS seating
FROM venue WHERE venuestate = 'NV' OR venuestate = 'NC'
ORDER BY 1
LIMIT 5;

seating
-----------------------------------
Ballys Hotel seats 0
Bank of America Stadium seats 73298
Bellagio Hotel seats 0
Caesars Palace seats 0
Harrahs Hotel seats 0
(5 rows)
```

Kueri berikut menggabungkan nilai CITY dan STATE dari tabel VENUE:

```
SELECT CONCAT(venuecity, venuestate)
FROM venue
WHERE venueseats > 75000
ORDER BY venueseats;

concat
-------------------
DenverCO
```

```
Kansas CityMO
East RutherfordNJ
LandoverMD
(4 rows)
```

Kueri berikut menggunakan fungsi CONCAT bersarang. Kueri menggabungkan nilai CITY dan STATE dari tabel VENUE tetapi membatasi string yang dihasilkan dengan koma dan spasi:

```
SELECT CONCAT(CONCAT(venuecity,', '),venuestate)
FROM venue
WHERE venueseats > 75000
ORDER BY venueseats;

concat
---------------------
Denver, CO
Kansas City, MO
East Rutherford, NJ
Landover, MD
(4 rows)
```

Contoh berikut menggabungkan dua ekspresi biner. Dim `abc` ana nilai biner (dengan representasi heksadesimal`616263`) dan `def` merupakan nilai biner (dengan representasi heksadesimal`646566`). Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari nilai biner.

```
SELECT CONCAT('abc'::VARBYTE, 'def'::VARBYTE);

concat
------------------
616263646566
```

Fungsi CRC32

CRC32 adalah fungsi yang digunakan untuk deteksi kesalahan. Fungsi ini menggunakan algoritma CRC32 untuk mendeteksi perubahan antara sumber dan data target. Fungsi CRC32 mengubah string panjang variabel menjadi string 8 karakter yang merupakan representasi teks dari nilai heksadesimal dari urutan biner 32 bit. Untuk mendeteksi perubahan antara sumber dan data target, gunakan fungsi CRC32 pada data sumber dan simpan output. Kemudian, gunakan fungsi CRC32 pada data target dan bandingkan output itu dengan output dari data sumber. Outputnya akan sama jika data tidak dimodifikasi, dan output akan berbeda jika data dimodifikasi.

Sintaksis

```
CRC32(string)
```

Argumen

senar

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

Jenis pengembalian

Fungsi CRC32 mengembalikan string 8-karakter yang merupakan representasi teks dari nilai heksadesimal dari urutan biner 32-bit. Fungsi Amazon Redshift CRC32 didasarkan pada polinomial. CRC-32C

Contoh

Untuk menunjukkan nilai 8-bit untuk stringAmazon Redshift.

```
SELECT CRC32('Amazon Redshift');

+----------+
|  crc32   |
+----------+
| f2726906 |
+----------+
```

Fungsi PERBEDAAN

Fungsi DIFFERENCE membandingkan kode Soundex Amerika dari dua string. Fungsi mengembalikan an INTEGER untuk menunjukkan jumlah karakter yang cocok antara kode Soundex.

Kode Soundex adalah string yang panjangnya empat karakter. Kode Soundex mewakili bagaimana sebuah kata terdengar daripada bagaimana ejaannya. Misalnya, Smith dan Smyth memiliki kode Soundex yang sama.

Sintaksis

```
DIFFERENCE(string1, string2)
```

Argumen

string1

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

senar2

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

Jenis pengembalian

INTEGER

Fungsi DIFFERENCE mengembalikan INTEGER nilai dari 0—4 yang menghitung jumlah karakter yang cocok dalam kode Soundex Amerika dari dua string. Kode Soundex memiliki 4 karakter, sehingga fungsi DIFFERENCE kembali 4 ketika semua 4 karakter dari nilai kode American Soundex string sama. DIFFER 0 ENCE kembali jika salah satu dari dua string kosong. Fungsi kembali 1 jika tidak ada string yang berisi karakter yang valid. Fungsi DIFFERENCE hanya mengubah karakter ASCII huruf kecil atau huruf besar alfabet bahasa Inggris, termasuk a—z dan A—Z. DIFFERENCE mengabaikan karakter lain.

Contoh

Untuk membandingkan nilai Soundex dari string % dan@, gunakan contoh berikut. Fungsi kembali 1 karena tidak ada string yang berisi karakter yang valid.

```
SELECT DIFFERENCE('%', '@');

+------------+
| difference |
+------------+
|          1 |
+------------+
```

Untuk membandingkan nilai Soundex Amazon dan string kosong, gunakan contoh berikut. Fungsi kembali 0 karena salah satu dari dua string kosong.

```
SELECT DIFFERENCE('Amazon', '');
```

```
+------------+
| difference |
+------------+
|          0 |
+------------+
```

Untuk membandingkan nilai Soundex dari string `Amazon` dan`Ama`, gunakan contoh berikut. Fungsi kembali 2 karena 2 karakter dari nilai Soundex string adalah sama.

```
SELECT DIFFERENCE('Amazon', 'Ama');

+------------+
| difference |
+------------+
|          2 |
+------------+
```

Untuk membandingkan nilai Soundex dari string `Amazon` dan`+-*/%Amazon`, gunakan contoh berikut. Fungsi kembali 4 karena semua 4 karakter dari nilai Soundex string adalah sama. Perhatikan bahwa fungsi mengabaikan karakter yang tidak valid `+-*/%` di string kedua.

```
SELECT DIFFERENCE('Amazon', '+-*/%Amazon');

+------------+
| difference |
+------------+
|          4 |
+------------+
```

Untuk membandingkan nilai Soundex dari string `AC/DC` dan`Ay See Dee See`, gunakan contoh berikut. Fungsi kembali 4 karena semua 4 karakter dari nilai Soundex string adalah sama.

```
SELECT DIFFERENCE('AC/DC', 'Ay See Dee See');

+------------+
| difference |
+------------+
|          4 |
+------------+
```

Fungsi INITCAP

Mengkapitalisasi huruf pertama dari setiap kata dalam string tertentu. INITCAP mendukung karakter UTF-8 multibyte, hingga maksimal empat byte per karakter.

Sintaksis

```
INITCAP(string)
```

Pendapat

senar

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

Jenis pengembalian

VARCHAR

Catatan penggunaan

Fungsi INITCAP membuat huruf pertama dari setiap kata dalam string huruf besar, dan huruf berikutnya dibuat (atau kiri) huruf kecil. Oleh karena itu, penting untuk memahami karakter mana (selain karakter spasi) yang berfungsi sebagai pemisah kata. Karakter pemisah kata adalah karakter non-alfanumerik, termasuk tanda baca, simbol, dan karakter kontrol. Semua karakter berikut adalah pemisah kata:

```
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
```

Tab, karakter baris baru, umpan formulir, umpan baris, dan pengembalian kereta juga merupakan pemisah kata.

Contoh

Contoh berikut menggunakan data dari tabel KATEGORI dan PENGGUNA dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengkapitalisasi inisial setiap kata di kolom CATDESC, gunakan contoh berikut.

```
SELECT catid, catdesc, INITCAP(catdesc)
FROM category
ORDER BY 1, 2, 3;

+-------+--------------------------------------------
+-----------------------------------------+
| catid |                  catdesc                   |                  initcap
          |
+-------+--------------------------------------------
+-----------------------------------------+
|     1 | Major League Baseball                      | Major League Baseball
          |
|     2 | National Hockey League                     | National Hockey League
          |
|     3 | National Football League                   | National Football League
          |
|     4 | National Basketball Association            | National Basketball Association
          |
|     5 | Major League Soccer                        | Major League Soccer
          |
|     6 | Musical theatre                            | Musical Theatre
          |
|     7 | All non-musical theatre                    | All Non-Musical Theatre
          |
|     8 | All opera and light opera                  | All Opera And Light Opera
          |
|     9 | All rock and pop music concerts            | All Rock And Pop Music Concerts
          |
|    10 | All jazz singers and bands                 | All Jazz Singers And Bands
          |
|    11 | All symphony, concerto, and choir concerts | All Symphony, Concerto, And
 Choir Concerts |
+-------+--------------------------------------------
+-----------------------------------------+
```

Untuk menunjukkan bahwa fungsi INITCAP tidak mempertahankan karakter huruf besar ketika mereka tidak memulai kata, gunakan contoh berikut. Misalnya, string `MLB` menjadi`Mlb`.

```
SELECT INITCAP(catname)
FROM category
ORDER BY catname;

+-----------+
```

```
|  initcap  |
+-----------+
| Classical |
| Jazz      |
| Mlb       |
| Mls       |
| Musicals  |
| Nba       |
| Nfl       |
| Nhl       |
| Opera     |
| Plays     |
| Pop       |
+-----------+
```

Untuk menunjukkan bahwa karakter non-alfanumerik selain spasi berfungsi sebagai pemisah kata, gunakan contoh berikut. Beberapa huruf di setiap string akan dikapitalisasi.

```
SELECT email, INITCAP(email)
FROM users
ORDER BY userid DESC LIMIT 5;

+-----------------------------------+-----------------------------------+
|               email               |              initcap              |
+-----------------------------------+-----------------------------------+
| urna.Ut@egetdictumplacerat.edu    | Urna.Ut@Egetdictumplacerat.Edu    |
| nibh.enim@egestas.ca              | Nibh.Enim@Egestas.Ca              |
| in@Donecat.ca                     | In@Donecat.Ca                     |
| sodales@blanditviverraDonec.ca    | Sodales@Blanditviverradonec.Ca    |
| sociis.natoque.penatibus@vitae.org | Sociis.Natoque.Penatibus@Vitae.Org |
+-----------------------------------+-----------------------------------+
```

Fungsi KIRI dan KANAN

Fungsi-fungsi ini mengembalikan jumlah karakter paling kiri atau paling kanan yang ditentukan dari string karakter.

Jumlahnya didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal.

Sintaksis

```
LEFT( string,  integer )
```

```
RIGHT( string,  integer )
```

Argumen

senar

Sebuah CHAR string, VARCHAR string, atau ekspresi apa pun yang mengevaluasi ke VARCHAR string CHAR atau.

bilangan bulat

Integer positif.

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut menggunakan data dari tabel EVENT dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan 5 karakter paling kiri dan 5 paling kanan dari nama peristiwa yang memiliki ID peristiwa antara 1000 dan 1005, gunakan contoh berikut.

```
SELECT eventid, eventname,
LEFT(eventname,5) AS left_5,
RIGHT(eventname,5) AS right_5
FROM event
WHERE eventid BETWEEN 1000 AND 1005
ORDER BY 1;

+---------+----------------+--------+---------+
| eventid |   eventname    | left_5 | right_5 |
+---------+----------------+--------+---------+
|    1000 | Gypsy          | Gypsy  | Gypsy   |
|    1001 | Chicago        | Chica  | icago   |
|    1002 | The King and I | The K  | and I   |
|    1003 | Pal Joey       | Pal J  | Joey    |
|    1004 | Grease         | Greas  | rease   |
|    1005 | Chicago        | Chica  | icago   |
```

```
+---------+----------------+--------+---------+
```

Fungsi LEN

Mengembalikan panjang string yang ditentukan sebagai jumlah karakter.

Sintaksis

LEN adalah sinonim dariFungsi LENGTH,, Fungsi CHAR_LENGTHFungsi CHARACTER_LENGTH, danFungsi TEXTLEN.

```
LEN(expression)
```

Pendapat

ekspresi

Sebuah CHAR string, VARCHAR string, VARBYTE ekspresi, atau ekspresi yang secara implisit mengevaluasi keCHAR,VARCHAR, atau VARBYTE tipe.

Jenis pengembalian

INTEGER

Fungsi LEN mengembalikan integer yang menunjukkan jumlah karakter dalam string input.

Jika string input adalah string karakter, fungsi LEN mengembalikan jumlah karakter aktual dalam string multi-byte, bukan jumlah byte. Misalnya, VARCHAR(12) kolom diperlukan untuk menyimpan tiga karakter Mandarin empat byte. Fungsi LEN akan kembali 3 untuk string yang sama. Untuk mendapatkan panjang string dalam byte, gunakan OCTET_LENGTH fungsi.

Catatan penggunaan

Jika ekspresi adalah CHAR string, spasi akhir tidak dihitung.

Jika ekspresi adalah VARCHAR string, spasi akhir dihitung.

Contoh

Untuk mengembalikan jumlah byte dan jumlah karakter dalam stringfrançais, gunakan contoh berikut.

```
SELECT OCTET_LENGTH('français'),
LEN('français');

+--------------+-----+
| octet_length | len |
+--------------+-----+
|            9 |   8 |
+--------------+-----+
```

Untuk mengembalikan jumlah byte dan jumlah karakter dalam string `français` tanpa menggunakan fungsi OCTET_LENGTH, gunakan contoh berikut. Untuk informasi selengkapnya, lihat Fungsi CAST.

```
SELECT LEN(CAST('français' AS VARBYTE)) as bytes, LEN('français');

+-------+-----+
| bytes | len |
+-------+-----+
|     9 |   8 |
+-------+-----+
```

Untuk mengembalikan jumlah karakter dalam string tanpa spasi akhir, `cat cat` dengan tiga spasi di belakang, `cat` dengan tiga spasi akhir yang ditampilkan sebagai CHAR panjang 6, dan `cat` dengan tiga spasi akhir dicetak sebagai VARCHAR panjang 6, gunakan contoh berikut. Perhatikan bahwa fungsi tidak menghitung spasi akhir untuk CHAR string, tetapi itu menghitung spasi akhir untuk VARCHAR string.

```
SELECT LEN('cat'), LEN('cat   '), LEN(CAST('cat   ' AS CHAR(6))) AS len_char,
 LEN(CAST('cat   ' AS VARCHAR(6))) AS len_varchar;

+-----+-----+----------+-------------+
| len | len | len_char | len_varchar |
+-----+-----+----------+-------------+
|   3 |   6 |        3 |           6 |
+-----+-----+----------+-------------+
```

Contoh berikut menggunakan data dari tabel VENUE dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan 10 nama venue terpanjang dalam tabel VENUE, gunakan contoh berikut.

```
SELECT venuename, LEN(venuename)
```

```
FROM venue
ORDER BY 2 DESC, 1
LIMIT 10;

+----------------------------------------+-----+
|               venuename                | len |
+----------------------------------------+-----+
| Saratoga Springs Performing Arts Center |  39 |
| Lincoln Center for the Performing Arts  |  38 |
| Nassau Veterans Memorial Coliseum       |  33 |
| Jacksonville Municipal Stadium          |  30 |
| Rangers BallPark in Arlington           |  29 |
| University of Phoenix Stadium           |  29 |
| Circle in the Square Theatre            |  28 |
| Hubert H. Humphrey Metrodome            |  28 |
| Oriole Park at Camden Yards             |  27 |
| Dick's Sporting Goods Park              |  26 |
+----------------------------------------+-----+
```

Fungsi LENGTH

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi LOWER

Mengkonversi string ke huruf kecil. LOWER mendukung karakter UTF-8 multibyte, hingga maksimal empat byte per karakter.

Sintaksis

```
LOWER(string)
```

Pendapat

senar

Sebuah VARCHAR string atau ekspresi apa pun yang mengevaluasi VARCHAR tipe.

Jenis pengembalian

string

Fungsi LOWER mengembalikan string yang merupakan tipe data yang sama dengan string input. Misalnya, jika input adalah CHAR string, fungsi akan mengembalikan CHAR string.

Contoh

Contoh berikut menggunakan data dari tabel CATEGORY dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengubah VARCHAR string di kolom CATNAME menjadi huruf kecil, gunakan contoh berikut.

```
SELECT catname, LOWER(catname) FROM category ORDER BY 1,2;

+-----------+-----------+
|  catname  |   lower   |
+-----------+-----------+
| Classical | classical |
| Jazz      | jazz      |
| MLB       | mlb       |
| MLS       | mls       |
| Musicals  | musicals  |
| NBA       | nba       |
| NFL       | nfl       |
| NHL       | nhl       |
| Opera     | opera     |
| Plays     | plays     |
| Pop       | pop       |
+-----------+-----------+
```

Fungsi LPAD dan RPAD

Fungsi-fungsi ini mendahului atau menambahkan karakter ke string, berdasarkan panjang yang ditentukan.

Sintaksis

```
LPAD(string1, length, [ string2 ])
```

```
RPAD(string1, length, [ string2 ])
```

Argumen

string1

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

panjang

Bilangan bulat yang mendefinisikan panjang hasil fungsi. Panjang string didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Jika string1 lebih panjang dari panjang yang ditentukan, itu terpotong (di sebelah kanan). Jika panjangnya nol atau angka negatif, hasil fungsi adalah string kosong.

senar2

(Opsional) Satu atau lebih karakter yang ditambahkan atau ditambahkan ke string1. Jika argumen ini tidak ditentukan, spasi digunakan.

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut menggunakan data dari tabel EVENT dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk memotong kumpulan nama acara tertentu menjadi 20 karakter dan menambahkan nama yang lebih pendek dengan spasi, gunakan contoh berikut.

```
SELECT LPAD(eventname, 20) FROM event
WHERE eventid BETWEEN 1 AND 5 ORDER BY 1;

+----------------------+
|         lpad         |
+----------------------+
|              Salome |
|        Il Trovatore |
```

```
|       Boris Godunov |
|     Gotterdammerung |
|La Cenerentola (Cind |
+---------------------+
```

Untuk memotong kumpulan nama acara yang sama menjadi 20 karakter tetapi menambahkan nama yang lebih pendek dengan`0123456789`, gunakan contoh berikut.

```
SELECT RPAD(eventname, 20,'0123456789') FROM event
WHERE eventid BETWEEN 1 AND 5 ORDER BY 1;

+----------------------+
|         rpad         |
+----------------------+
| Boris Godunov0123456 |
| Gotterdammerung01234 |
| Il Trovatore01234567 |
| La Cenerentola (Cind |
| Salome01234567890123 |
+----------------------+
```

Fungsi LTRIM

Memangkas karakter dari awal string. Menghapus string terpanjang yang hanya berisi karakter dalam daftar karakter trim. Pemangkasan selesai ketika karakter trim tidak muncul di string input.

Sintaksis

```
LTRIM( string [, trim_chars] )
```

Argumen

senar

Sebuah kolom string, ekspresi, atau string literal yang akan dipangkas.

trim_chars

Sebuah kolom string, ekspresi, atau string literal yang mewakili karakter yang akan dipangkas dari awal string. Jika tidak ditentukan, spasi digunakan sebagai karakter trim.

Jenis pengembalian

Fungsi LTRIM mengembalikan string karakter yang merupakan tipe data yang sama dengan string input (CHAR atau VARCHAR).

Contoh

Contoh berikut memangkas tahun dari `listime` kolom. Karakter trim dalam string literal `'2008-'` menunjukkan karakter yang akan dipangkas dari kiri. Jika Anda menggunakan karakter trim`'028-'`, Anda mencapai hasil yang sama.

```
select listid, listtime, ltrim(listtime, '2008-')
from listing
order by 1, 2, 3
limit 10;

listid |      listtime       |     ltrim
-------+---------------------+---------------
     1 | 2008-01-24 06:43:29 | 1-24 06:43:29
     2 | 2008-03-05 12:25:29 | 3-05 12:25:29
     3 | 2008-11-01 07:35:33 | 11-01 07:35:33
     4 | 2008-05-24 01:18:37 | 5-24 01:18:37
     5 | 2008-05-17 02:29:11 | 5-17 02:29:11
     6 | 2008-08-15 02:08:13 | 15 02:08:13
     7 | 2008-11-15 09:38:15 | 11-15 09:38:15
     8 | 2008-11-09 05:07:30 | 11-09 05:07:30
     9 | 2008-09-09 08:03:36 | 9-09 08:03:36
    10 | 2008-06-17 09:44:54 | 6-17 09:44:54
```

LTRIM menghapus salah satu karakter di trim_chars ketika mereka muncul di awal string. Contoh berikut memangkas karakter 'C', 'D', dan 'G' ketika mereka muncul di awal VENUENAME, yang merupakan kolom VARCHAR.

```
select venueid, venuename, ltrim(venuename, 'CDG')
from venue
where venuename like '%Park'
order by 2
limit 7;

venueid | venuename                  | btrim
--------+----------------------------+-------------------------
    121 | ATT Park                   | ATT Park
```

```
    109 | Citizens Bank Park         | itizens Bank Park
    102 | Comerica Park              | omerica Park
      9 | Dick's Sporting Goods Park | ick's Sporting Goods Park
     97 | Fenway Park                | Fenway Park
    112 | Great American Ball Park   | reat American Ball Park
    114 | Miller Park                | Miller Park
```

Contoh berikut menggunakan karakter trim 2 yang diambil dari `venueid` kolom.

```
select ltrim('2008-01-24 06:43:29', venueid)
from venue where venueid=2;

ltrim
------------------
008-01-24 06:43:29
```

Contoh berikut tidak memotong karakter apa pun 2 karena a ditemukan sebelum karakter `'0'` trim.

```
select ltrim('2008-01-24 06:43:29', '0');

ltrim
-------------------
2008-01-24 06:43:29
```

Contoh berikut menggunakan karakter trim spasi default dan memangkas dua spasi dari awal string.

```
select ltrim('  2008-01-24 06:43:29');

ltrim
-------------------
2008-01-24 06:43:29
```

Fungsi OCTETINDEX

Fungsi OCTETINDEX mengembalikan lokasi substring dalam string sebagai sejumlah byte.

Sintaksis

```
OCTETINDEX(substring, string)
```

Argumen

substring

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

senar

Sebuah CHAR string, VARCHAR string, atau ekspresi yang secara implisit mengevaluasi ke VARCHAR tipe CHAR atau.

Jenis pengembalian

INTEGER

Fungsi OCTETINDEX mengembalikan INTEGER nilai yang sesuai dengan posisi substring dalam string sebagai sejumlah byte, di mana karakter pertama dalam string dihitung sebagai 1. Jika string tidak berisi karakter multibyte, hasilnya sama dengan hasil fungsi CHARINDEX. Jika string tidak berisi substring, fungsi kembali0. Jika substring kosong, fungsi kembali1.

Contoh

Untuk mengembalikan posisi substring q dalam stringAmazon Redshift, gunakan contoh berikut. Contoh ini kembali 0 karena substring tidak ada dalam string.

```
SELECT OCTETINDEX('q', 'Amazon Redshift');

+------------+
| octetindex |
+------------+
|          0 |
+------------+
```

Untuk mengembalikan posisi substring kosong dalam stringAmazon Redshift, gunakan contoh berikut. Contoh ini kembali 1 karena substring kosong.

```
SELECT OCTETINDEX('', 'Amazon Redshift');

+------------+
| octetindex |
```

```
+------------+
|          1 |
+------------+
```

Untuk mengembalikan posisi substring `Redshift` dalam string`Amazon Redshift`, gunakan contoh berikut. Contoh ini kembali 8 karena substring dimulai pada byte kedelapan string.

```
SELECT OCTETINDEX('Redshift', 'Amazon Redshift');

+------------+
| octetindex |
+------------+
|          8 |
+------------+
```

Untuk mengembalikan posisi substring `Redshift` dalam string`Amazon Redshift`, gunakan contoh berikut. Contoh ini kembali 21 karena enam karakter pertama dari string adalah karakter byte ganda.

```
SELECT OCTETINDEX('Redshift', 'Άμαζον Amazon Redshift');

+------------+
| octetindex |
+------------+
|         21 |
+------------+
```

Fungsi OCTET_LENGTH

Mengembalikan panjang string yang ditentukan sebagai jumlah byte.

Sintaksis

```
OCTET_LENGTH(expression)
```

Pendapat

ekspresi

Sebuah `CHAR` string, `VARCHAR` string, `VARBYTE` ekspresi, atau ekspresi yang secara implisit mengevaluasi ke`CHAR`,`VARCHAR`, atau `VARBYTE` tipe.

Jenis pengembalian

INTEGER

Fungsi OCTET_LENGTH mengembalikan integer yang menunjukkan jumlah byte dalam string input.

Jika string input adalah string karakter, LENS fungsi mengembalikan jumlah karakter aktual dalam string multi-byte, bukan jumlah byte. Misalnya, `VARCHAR(12)` kolom diperlukan untuk menyimpan tiga karakter Mandarin empat byte. Fungsi OCTET_LENGTH akan kembali `12` untuk string itu, dan fungsi LEN akan kembali `3` untuk string yang sama.

Catatan penggunaan

Jika *ekspresi* adalah `CHAR` string, fungsi mengembalikan panjang `CHAR` string. Misalnya, output dari `CHAR(6)` input adalah a`CHAR(6)`.

Jika *ekspresi* adalah `VARCHAR` string, spasi akhir dihitung.

Contoh

Untuk mengembalikan jumlah byte ketika string `francais` dengan tiga spasi akhir dilemparkan ke a `CHAR` dan `VARCHAR` tipe, gunakan contoh berikut. Untuk informasi selengkapnya, lihat Fungsi CAST.

```
SELECT OCTET_LENGTH(CAST('francais   ' AS CHAR(15))) AS octet_length_char,
 OCTET_LENGTH(CAST('francais   ' AS VARCHAR(15))) AS octet_length_varchar;

+------------------+----------------------+
| octet_length_char | octet_length_varchar |
+------------------+----------------------+
|               15 |                   11 |
+------------------+----------------------+
```

Untuk mengembalikan jumlah byte dan jumlah karakter dalam string`français`, gunakan contoh berikut.

```
SELECT OCTET_LENGTH('français'), LEN('français');

+--------------+-----+
| octet_length | len |
+--------------+-----+
|            9 |   8 |
```

```
+--------------+-----+
```

Untuk mengembalikan jumlah byte ketika string `français` dicetak sebagai a`VARBYTE`, gunakan contoh berikut.

```
SELECT OCTET_LENGTH(CAST('français' AS VARBYTE));

+--------------+
| octet_length |
+--------------+
|            9 |
+--------------+
```

Fungsi POSISI

Mengembalikan lokasi substring yang ditentukan dalam string.

Lihat Fungsi CHARINDEX dan Fungsi STRPOS untuk fungsi serupa.

Sintaksis

```
POSITION(substring IN string )
```

Argumen

substring

Substring untuk dicari di dalam string.

senar

String atau kolom yang akan dicari.

Jenis pengembalian

Fungsi POSITION mengembalikan yang `INTEGER` sesuai dengan posisi substring (berbasis satu, bukan berbasis nol). Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. POSISI kembali `0` jika substring tidak ditemukan di dalam string.

Contoh

Untuk mengembalikan posisi string `fish` dalam katadog, gunakan contoh berikut.

```
SELECT POSITION('fish' IN 'dog');

+-----------+
|  position |
+-----------+
|         0 |
+-----------+
```

Untuk mengembalikan posisi string `fish` dalam kata`dogfish`, gunakan contoh berikut.

```
SELECT POSITION('fish' IN 'dogfish');

+-----------+
|  position |
+-----------+
|         4 |
+-----------+
```

Contoh berikut menggunakan tabel SALES dari database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan jumlah transaksi penjualan yang berbeda dengan komisi lebih dari 999,00 dari tabel PENJUALAN, gunakan contoh berikut. Perintah ini menghitung komisi lebih besar dari 999,00 dengan memeriksa apakah desimal lebih dari 4 tempat dari awal nilai komisi.

```
SELECT DISTINCT POSITION('.' IN commission), COUNT (POSITION('.' IN commission))
FROM sales
WHERE POSITION('.' IN commission) > 4
GROUP BY POSITION('.' IN commission)
ORDER BY 1,2;

+-----------+-------+
|  position | count |
+-----------+-------+
|         5 |   629 |
+-----------+-------+
```

Fungsi QUOTE_IDENT

Fungsi QUOTE_IDENT mengembalikan string yang ditentukan sebagai string dengan tanda kutip ganda di depan dan tanda kutip ganda di belakang. Output fungsi dapat digunakan sebagai

pengidentifikasi dalam pernyataan SQL. Fungsi ini dengan tepat menggandakan tanda kutip ganda yang tertanam.

QUOTE_IDENT menambahkan tanda kutip ganda hanya jika diperlukan untuk membuat pengidentifikasi yang valid, ketika string berisi karakter non-pengenal atau sebaliknya akan dilipat menjadi huruf kecil. Untuk selalu mengembalikan string dengan tanda kutip tunggal, gunakan QUOTE_LITERAL.

Sintaksis

```
QUOTE_IDENT(string)
```

Pendapat

senar

A CHAR atau VARCHAR string.

Jenis pengembalian

Fungsi QUOTE_IDENT mengembalikan jenis string yang sama dengan string input.

Contoh

Untuk mengembalikan string "CAT" dengan tanda kutip ganda, gunakan contoh berikut.

```
SELECT QUOTE_IDENT('"CAT"');

+-------------+
| quote_ident |
+-------------+
| """CAT"""   |
+-------------+
```

Contoh berikut menggunakan data dari tabel CATEGORY dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan kolom CATNAME dikelilingi oleh tanda kutip, gunakan contoh berikut.

```
SELECT catid, QUOTE_IDENT(catname)
FROM category
ORDER BY 1,2;
```

```
+-------+-------------+
| catid | quote_ident |
+-------+-------------+
|     1 | "MLB"       |
|     2 | "NHL"       |
|     3 | "NFL"       |
|     4 | "NBA"       |
|     5 | "MLS"       |
|     6 | "Musicals"  |
|     7 | "Plays"     |
|     8 | "Opera"     |
|     9 | "Pop"       |
|    10 | "Jazz"      |
|    11 | "Classical" |
+-------+-------------+
```

Fungsi QUOTE_LITERAL

Fungsi QUOTE_LITERAL mengembalikan string yang ditentukan sebagai string dikutip tunggal sehingga dapat digunakan sebagai string literal dalam pernyataan SQL. Jika parameter input adalah angka, QUOTE_LITERAL memperlakukannya sebagai string. Dengan tepat menggandakan tanda kutip tunggal yang disematkan dan garis miring terbalik.

Sintaksis

```
QUOTE_LITERAL(string)
```

Pendapat

senar

A CHAR atau VARCHAR string.

Jenis pengembalian

Fungsi QUOTE_LITERAL mengembalikan VARCHAR string CHAR atau yang merupakan tipe data yang sama dengan string input.

Contoh

Untuk mengembalikan string ''CAT'' dengan tanda kutip TUNGGAL, gunakan contoh berikut.

```
SELECT QUOTE_LITERAL('''CAT''');

+---------------+
| quote_literal |
+---------------+
| '''CAT'''     |
+---------------+
```

Contoh berikut menggunakan data dari tabel CATEGORY dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan kolom CATNAME dikelilingi oleh tanda kutip tunggal, gunakan contoh berikut.

```
SELECT catid, QUOTE_LITERAL(catname)
FROM category
ORDER BY 1,2;

+-------+---------------+
| catid | quote_literal |
+-------+---------------+
|     1 | 'MLB'         |
|     2 | 'NHL'         |
|     3 | 'NFL'         |
|     4 | 'NBA'         |
|     5 | 'MLS'         |
|     6 | 'Musicals'    |
|     7 | 'Plays'       |
|     8 | 'Opera'       |
|     9 | 'Pop'         |
|    10 | 'Jazz'        |
|    11 | 'Classical'   |
+-------+---------------+
```

Untuk mengembalikan kolom CATID dikelilingi oleh tanda kutip tunggal, gunakan contoh berikut.

```
SELECT QUOTE_LITERAL(catid), catname
FROM category
ORDER BY 1,2;

+---------------+-----------+
| quote_literal |  catname  |
+---------------+-----------+
```

```
| '1'           | MLB       |
| '10'          | Jazz      |
| '11'          | Classical |
| '2'           | NHL       |
| '3'           | NFL       |
| '4'           | NBA       |
| '5'           | MLS       |
| '6'           | Musicals  |
| '7'           | Plays     |
| '8'           | Opera     |
| '9'           | Pop       |
+---------------+-----------+
```

Fungsi REGEXP_COUNT

Mencari string untuk pola ekspresi reguler dan mengembalikan bilangan bulat yang menunjukkan berapa kali pola yang ditentukan terjadi dalam string. Jika tidak ada kecocokan yang ditemukan, maka fungsi kembali`0`. Untuk informasi selengkapnya tentang ekspresi reguler, lihat Operator POSIX dan Ek spresi regul er di Wikipedia.

Sintaksis

```
REGEXP_COUNT( source_string, pattern [, position [, parameters ] ] )
```

Argumen

sumber_string

A `CHAR` atau `VARCHAR` string.

pola

Sebuah UTF-8 string literal yang mewakili pola ekspresi reguler. Untuk informasi selengkapnya, lihat Operator POSIX.

posisi

(Opsional) Positif `INTEGER` yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal. Nilai default-nya 1. Jika posisi kurang dari1, pencarian dimulai pada karakter pertama source _string. Jika posisi lebih besar dari jumlah karakter dalam source_string, hasilnya adalah. `0`

parameter

(Opsional) Satu atau lebih literal string yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c - Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar-kecil.
- i — Lakukan pencocokan yang tidak peka huruf besar/kecil.
- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE). Untuk informasi selengkapnya tentang PCRE, lihat Perl Compatible Regular Expressions di Wikipedia.

Jenis pengembalian

INTEGER

Contoh

Untuk menghitung berapa kali urutan tiga huruf terjadi, gunakan contoh berikut.

```
SELECT REGEXP_COUNT('abcdefghijklmnopqrstuvwxyz', '[a-z]{3}');

+--------------+
| regexp_count |
+--------------+
|            8 |
+--------------+
```

Untuk menghitung kemunculan string FOX menggunakan pencocokan case-insensitive, gunakan contoh berikut.

```
SELECT REGEXP_COUNT('the fox', 'FOX', 1, 'i');

+--------------+
| regexp_count |
+--------------+
|            1 |
+--------------+
```

Untuk menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil, gunakan contoh berikut. Contoh menggunakan ?=

operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Contoh ini menghitung jumlah kemunculan kata-kata tersebut, dengan pencocokan peka huruf besar-kecil.

```
SELECT REGEXP_COUNT('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 'p');

+--------------+
| regexp_count |
+--------------+
|            2 |
+--------------+
```

Untuk menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil, gunakan contoh berikut. Ini menggunakan ?= operator, yang memiliki konotasi spesifik dalam PCRE. Contoh ini menghitung jumlah kemunculan kata-kata tersebut, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan yang tidak peka huruf besar-kecil.

```
SELECT REGEXP_COUNT('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 'ip');

+--------------+
| regexp_count |
+--------------+
|            3 |
+--------------+
```

Contoh berikut menggunakan data dari tabel USERS dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menghitung berapa kali nama domain tingkat atas adalah salah satu `org` atau`edu`, gunakan contoh berikut.

```
SELECT email, REGEXP_COUNT(email,'@[^.]*\.(org|edu)') FROM users
ORDER BY userid LIMIT 4;

+----------------------------------------------+--------------+
|                    email                     | regexp_count |
+----------------------------------------------+--------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |            1 |
| Suspendisse.tristique@nonnisiAenean.edu       |            1 |
```

```
| amet.faucibus.ut@condimentumegetvolutpat.ca   |           0 |
| sed@lacusUtnec.ca                             |           0 |
+-----------------------------------------------+-------------+
```

Fungsi REGEXP_INSTR

Mencari string untuk pola ekspresi reguler dan mengembalikan bilangan bulat yang menunjukkan posisi awal atau posisi akhir dari substring yang cocok. Jika tidak ada kecocokan yang ditemukan, maka fungsi kembali`0`. REGEXP_INSTR mirip dengan fungsi POSITION, tetapi dengan REGEXP_INSTR, Anda dapat mencari string untuk pola ekspresi reguler. Untuk informasi selengkapnya tentang ekspresi reguler, lihat Operator POSIX dan Ek spresi regul er di Wikipedia.

Sintaksis

```
REGEXP_INSTR( source_string, pattern [, position [, occurrence] [, option [, parameters
 ] ] ] ] )
```

Argumen

sumber_string

Ekspresi string, seperti nama kolom, untuk dicari.

pola

Sebuah UTF-8 string literal yang mewakili pola ekspresi reguler. Untuk informasi selengkapnya, lihat Operator POSIX.

posisi

(Opsional) Positif `INTEGER` yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal. Nilai default-nya 1. Jika posisi kurang dari1, pencarian dimulai pada karakter pertama source _string. Jika posisi lebih besar dari jumlah karakter dalam source_string, hasilnya adalah. `0`

kejadian

(Opsional) Positif `INTEGER` yang menunjukkan kemunculan pola mana yang akan digunakan. REGEXP_INSTR melewatkan pertandingan pertama. *`occurrence`*`-1` Nilai default-nya 1. Jika kejadian kurang dari 1 atau lebih besar dari jumlah karakter dalam source_string, pencarian diabaikan dan hasilnya adalah. `0`

pilihan

(Opsional) Nilai yang menunjukkan apakah akan mengembalikan posisi karakter pertama pertandingan (`0`) atau posisi karakter pertama setelah akhir pertandingan (1). Nilai bukan nol sama 1 dengan. Nilai default-nya adalah `0`.

parameter

(Opsional) Satu atau lebih literal string yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c - Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar-kecil.
- i — Lakukan pencocokan yang tidak peka huruf besar/kecil.
- e — Ekstrak substring menggunakan subexpression.

  Jika pola menyer takan subexpression, REGEXP_INSTR mencocokkan substring menggunakan subexpression pertama dalam pola. REGEXP_INSTR hanya mempertimbangkan subexpression pertama; subexpresi tambahan diabaikan. Jika pola tidak memiliki subexpression, REGEXP_INSTR mengabaikan parameter 'e'.
- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE). Untuk informasi selengkapnya tentang PCRE, lihat Perl Compatible Regular Expressions di Wikipedia.

Jenis pengembalian

Bilangan Bulat

Contoh

Contoh berikut menggunakan data dari tabel USERS dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mencari @ karakter yang memulai nama domain dan mengembalikan posisi awal dari kecocokan pertama, gunakan contoh berikut.

```
SELECT email, REGEXP_INSTR(email, '@[^.]*')
FROM users
ORDER BY userid LIMIT 4;
```

```
+-----------------------------------------------+--------------+
|                     email                     | regexp_instr |
+-----------------------------------------------+--------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |           21 |
| Suspendisse.tristique@nonnisiAenean.edu       |           22 |
| amet.faucibus.ut@condimentumegetvolutpat.ca   |           17 |
| sed@lacusUtnec.ca                             |            4 |
+-----------------------------------------------+--------------+
```

Untuk mencari varian kata `Center` dan mengembalikan posisi awal dari kecocokan pertama, gunakan contoh berikut.

```
SELECT venuename, REGEXP_INSTR(venuename,'[cC]ent(er|re)$')
FROM venue
WHERE REGEXP_INSTR(venuename,'[cC]ent(er|re)$') > 0
ORDER BY venueid LIMIT 4;

+----------------------+--------------+
|       venuename      | regexp_instr |
+----------------------+--------------+
| The Home Depot Center |           16 |
| Izod Center           |            6 |
| Wachovia Center       |           10 |
| Air Canada Centre     |           12 |
+----------------------+--------------+
```

Untuk menemukan posisi awal dari kejadian pertama string`FOX`, menggunakan logika pencocokan case-insensitive, gunakan contoh berikut.

```
SELECT REGEXP_INSTR('the fox', 'FOX', 1, 1, 0, 'i');

+--------------+
| regexp_instr |
+--------------+
|            5 |
+--------------+
```

Untuk menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil, gunakan contoh berikut. Ini menggunakan ?= operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Contoh ini menemukan posisi awal dari kata kedua tersebut.

```
SELECT REGEXP_INSTR('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 0, 'p');

+--------------+
| regexp_instr |
+--------------+
|           21 |
+--------------+
```

Untuk menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil, gunakan contoh berikut. Ini menggunakan ?= operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Contoh ini menemukan posisi awal dari kata kedua tersebut, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan yang tidak peka huruf besar-kecil.

```
SELECT REGEXP_INSTR('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 0, 'ip');

+--------------+
| regexp_instr |
+--------------+
|           15 |
+--------------+
```

Fungsi REGEXP_REPLACE

Mencari string untuk pola ekspresi reguler dan mengganti setiap kejadian pola dengan string yang ditentukan. REGEXP_REPLACE mirip dengan Fungsi REPLACE, tetapi dengan REGEXP_REPLACE, Anda dapat mencari string untuk pola ekspresi reguler. Untuk informasi selengkapnya tentang ekspresi reguler, lihat Operator POSIX dan Ek spresi regul er di Wikipedia.

REGEXP_REPLACE mirip dengan Fungsi TRANSLATE dan Fungsi REPLACE, kecuali bahwa TRANSLATE membuat beberapa substitusi karakter tunggal dan REPLACE mengganti satu seluruh string dengan string lain, sedangkan dengan REGEXP_REPLACE, Anda dapat mencari string untuk pola ekspresi reguler.

Sintaksis

```
REGEXP_REPLACE( source_string, pattern [, replace_string [ , position [, parameters
 ] ] ] )
```

Argumen

sumber_string

Ekspresi A CHAR atau VARCHAR string, seperti nama kolom, untuk dicari.

pola

Sebuah UTF-8 string literal yang mewakili pola ekspresi reguler. Untuk informasi selengkapnya, lihat Operator POSIX.

ganti_string

(Opsional) CHAR Ek VARCHAR spresi A atau string, seperti nama kolom, yang akan menggantikan setiap kejadian pola. Defaultnya adalah string kosong (“”).

posisi

(Opsional) Bilangan bulat positif yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal. Nilai default-nya 1. Jika posisi kurang dari1, pencarian dimulai pada karakter pertama source _string. Jika posisi lebih besar dari jumlah karakter dalam source_string, hasilnya adalah source_string.

parameter

(Opsional) Satu atau lebih literal string yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c - Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar-kecil.
- i — Lakukan pencocokan yang tidak peka huruf besar/kecil.
- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE). Untuk informasi selengkapnya tentang PCRE, lihat Perl Compatible Regular Expressions di Wikipedia.

Jenis pengembalian

VARCHAR

Jika pola atau replace_string adalahNULL, fungsi kembali. NULL

Contoh

Untuk mengganti semua kejadian string `FOX` dalam nilai `quick brown fox` menggunakan pencocokan case-insensitive, gunakan contoh berikut.

```
SELECT REGEXP_REPLACE('the fox', 'FOX', 'quick brown fox', 1, 'i');

+---------------------+
|   regexp_replace    |
+---------------------+
| the quick brown fox |
+---------------------+
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil. Ini menggunakan `?=` operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Untuk mengganti setiap kejadian kata seperti itu dengan nilai`[hidden]`, gunakan contoh berikut.

```
SELECT REGEXP_REPLACE('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 '[hidden]', 1, 'p');

+-------------------------------+
|        regexp_replace         |
+-------------------------------+
| [hidden] plain A1234 [hidden] |
+-------------------------------+
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil. Ini menggunakan `?=` operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Untuk mengganti setiap kejadian kata seperti itu dengan nilai`[hidden]`, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan yang tidak peka huruf besar-kecil, gunakan contoh berikut.

```
SELECT REGEXP_REPLACE('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 '[hidden]', 1, 'ip');

+---------------------------------+
|          regexp_replace         |
+---------------------------------+
| [hidden] plain [hidden] [hidden] |
```

```
+---------------------------------+
```

Contoh berikut menggunakan data dari tabel USERS dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk menghapus nama domain @ dan dari alamat email, gunakan contoh berikut.

```
SELECT email, REGEXP_REPLACE(email, '@.*\\.(org|gov|com|edu|ca)$')
FROM users
ORDER BY userid LIMIT 4;

+-----------------------------------------------+----------------------+
|                     email                     |    regexp_replace    |
+-----------------------------------------------+----------------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu | Etiam.laoreet.libero |
| Suspendisse.tristique@nonnisiAenean.edu       | Suspendisse.tristique |
| amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut     |
| sed@lacusUtnec.ca                             | sed                  |
+-----------------------------------------------+----------------------+
```

Untuk mengganti nama domain alamat email dengan`internal.company.com`, gunakan contoh berikut.

```
SELECT email, REGEXP_REPLACE(email, '@.*\\.[[:alpha:]]{2,3}','@internal.company.com')
FROM users
ORDER BY userid LIMIT 4;

+-----------------------------------------------
+-------------------------------------------+
|                     email                     |               regexp_replace
      |
+-----------------------------------------------
+-------------------------------------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |
 Etiam.laoreet.libero@internal.company.com  |
| Suspendisse.tristique@nonnisiAenean.edu       |
 Suspendisse.tristique@internal.company.com |
| amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut@internal.company.com
      |
| sed@lacusUtnec.ca                             | sed@internal.company.com
      |
```

```
+-------------------------------------------
+-----------------------------------------+
```

Fungsi REGEXP_SUBSTR

Mengembalikan karakter dari string dengan mencarinya untuk pola ekspresi reguler. REGEXP_SUBSTR mirip dengan fung Fungsi SUBSTRING sinya, tetapi dengan REGEXP_SUBSTR, Anda dapat mencari string untuk pola ekspresi reguler. Jika fungsi tidak dapat mencocokkan ekspresi reguler dengan karakter apa pun dalam string, ia mengembalikan string kosong. Untuk informasi selengkapnya tentang ekspresi reguler, lihat Operator POSIX dan Ek spresi regul er di Wikipedia.

Sintaksis

```
REGEXP_SUBSTR( source_string, pattern [, position [, occurrence [, parameters ] ] ] )
```

Argumen

sumber_string

Ekspresi string yang akan dicari.

pola

Sebuah UTF-8 string literal yang mewakili pola ekspresi reguler. Untuk informasi selengkapnya, lihat Operator POSIX.

posisi

Bilangan bulat positif yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Default-nya adalah 1. Jika posisi kurang dari 1, pencarian dimulai pada karakter pertama source _string. Jika posisi lebih besar dari jumlah karakter dalam source_string, hasilnya adalah string kosong (“”).

kejadian

Bilangan bulat positif yang menunjukkan kemunculan pola mana yang akan digunakan. REGEXP_SUBSTR melewatkan kejadian pertama -1 kecocokan. Default-nya adalah 1. Jika kejadian kurang dari 1 atau lebih besar dari jumlah karakter dalam source_string, pencarian diabaikan dan hasilnya kosong.

parameter

Satu atau lebih literal string yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c - Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar-kecil.
- i — Lakukan pencocokan yang tidak peka huruf besar/kecil.
- e — Ekstrak substring menggunakan subexpression.

  Jika pola menyertakan subexpression, REGEXP_SUBSTR mencocokkan substring menggunakan subexpression pertama dalam pola. Subexpression adalah ekspresi dalam pola yang dikurung dengan tanda kurung. Misalnya, untuk pola `'This is a (\\w+)'` cocok ekspresi pertama dengan string `'This is a '` diikuti oleh kata. Alih-alih mengembalikan pola, REGEXP_SUBSTR dengan e parameter hanya mengembalikan string di dalam subexpression.

  REGEXP_SUBSTR hanya mempertimbangkan subexpression pertama; subexpresi tambahan diabaikan. Jika pola tidak memiliki subexpression, REGEXP_SUBSTR mengabaikan parameter 'e'.
- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE). Untuk informasi selengkapnya tentang PCRE, lihat Perl Compatible Regular Expressions di Wikipedia.

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut mengembalikan bagian alamat email antara karakter @ dan ekstensi domain. `users`Data yang ditanyakan berasal dari data sampel Amazon Redshift. Untuk informasi selengkapnya, lihat Database sampel.

```
SELECT email, regexp_substr(email,'@[^.]*')
FROM users
ORDER BY userid LIMIT 4;

                     email                     |      regexp_substr
-----------------------------------------------+--------------------------
```

```
 Suspendisse.tristique@nonnisiAenean.edu       | @nonnisiAenean
 amet.faucibus.ut@condimentumegetvolutpat.ca   | @condimentumegetvolutpat
 sed@lacusUtnec.ca                             | @lacusUtnec
 Cum@accumsan.com                              | @accumsan
```

Contoh berikut mengembalikan bagian input yang sesuai dengan kejadian pertama string F0X menggunakan pencocokan case-insensitive.

```
SELECT regexp_substr('the fox', 'FOX', 1, 1, 'i');

 regexp_substr
---------------
 fox
```

Contoh berikut mengembalikan bagian input yang sesuai dengan kejadian kedua string F0X menggunakan pencocokan case-insensitive. Hasilnya kosong (non-null, panjang 0) karena tidak ada kejadian kedua.

```
SELECT regexp_substr('the fox', 'FOX', 1, 2, 'i');

 regexp_substr
---------------

```

Contoh berikut mengembalikan bagian pertama dari input yang dimulai dengan huruf kecil. Ini secara fungsional identik dengan pernyataan SELECT yang sama tanpa c parameter.

```
SELECT regexp_substr('THE SECRET CODE IS THE LOWERCASE PART OF 1931abc0EZ.', '[a-z]+',
 1, 1, 'c');

 regexp_substr
---------------
 abc
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Contoh ini mengembalikan bagian input yang sesuai dengan kata kedua tersebut.

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 'p');
```

```
 regexp_substr
---------------
 a1234
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang berisi setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi tampilan ke depan tertentu di PCRE. Contoh ini mengembalikan bagian input yang sesuai dengan kata kedua tersebut, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan case-insensitive.

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 'ip');

 regexp_substr
---------------
 A1234
```

Contoh berikut menggunakan subexpression untuk menemukan string kedua yang cocok dengan pola `'this is a (\\w+)'` menggunakan pencocokan case-insensitive. Ini mengembalikan subexpression di dalam tanda kurung.

```
SELECT regexp_substr(
               'This is a cat, this is a dog. This is a mouse.',
               'this is a (\\w+)', 1, 2, 'ie');

 regexp_substr
---------------
 dog
```

Fungsi REPEAT

Mengulangi string jumlah yang ditentukan beberapa kali. Jika parameter input adalah numerik, REPEAT memperlakukannya sebagai string.

Sinonim untukFungsi REPLIKA.

Sintaksis

```
REPEAT(string, integer)
```

Argumen

senar

Parameter input pertama adalah string yang akan diulang.

bilangan bulat

Parameter kedua adalah `INTEGER` menunjukkan berapa kali untuk mengulangi string.

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut menggunakan data dari tabel CATEGORY dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengulangi nilai kolom CATID dalam tabel CATEGORY tiga kali, gunakan contoh berikut.

```
SELECT catid, REPEAT(catid,3)
FROM category
ORDER BY 1,2;

+-------+--------+
| catid | repeat |
+-------+--------+
|     1 |    111 |
|     2 |    222 |
|     3 |    333 |
|     4 |    444 |
|     5 |    555 |
|     6 |    666 |
|     7 |    777 |
|     8 |    888 |
|     9 |    999 |
|    10 | 101010 |
|    11 | 111111 |
+-------+--------+
```

Contoh berikut menunjukkan menghasilkan string hingga 16.000.000 byte:

```
SELECT
    LEN(REPEAT('X', 5000000)) AS five_million_bytes,
    LEN(REPEAT('Y', 16000000)) AS sixteen_million_bytes;

 five_million_bytes  | sixteen_million_bytes
---------+-----------
 5000000  | 16000000
```

Fungsi REPLACE

Mengganti semua kemunculan satu set karakter dalam string yang ada dengan karakter tertentu lainnya.

REPLACE mirip dengan Fungsi TRANSLATE danFungsi REGEXP_REPLACE, kecuali bahwa TRANSLATE membuat beberapa substitusi karakter tunggal dan dengan REGEXP_REPLACE, Anda dapat mencari string untuk pola ekspresi reguler, sementara REPLACE mengganti satu seluruh string dengan string lain.

Sintaksis

```
REPLACE(string, old_chars, new_chars)
```

Argumen

senar

CHARatau VARCHAR string yang akan dicari pencarian

karakter lama

CHARatau VARCHAR string untuk diganti.

karakter baru

Baru CHAR atau VARCHAR string menggantikan old_ string.

Jenis pengembalian

VARCHAR

Jika old_chars atau new_chars adalah, pengembaliannya adalah. NULL NULL

Contoh

Contoh berikut menggunakan data dari tabel CATEGORY dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengonversi string Shows ke Theatre dalam bidang CATGROUP, gunakan contoh berikut.

```
SELECT catid, catgroup, REPLACE(catgroup, 'Shows', 'Theatre')
FROM category
ORDER BY 1,2,3;

+-------+----------+----------+
| catid | catgroup | replace  |
+-------+----------+----------+
|     1 | Sports   | Sports   |
|     2 | Sports   | Sports   |
|     3 | Sports   | Sports   |
|     4 | Sports   | Sports   |
|     5 | Sports   | Sports   |
|     6 | Shows    | Theatre  |
|     7 | Shows    | Theatre  |
|     8 | Shows    | Theatre  |
|     9 | Concerts | Concerts |
|    10 | Concerts | Concerts |
|    11 | Concerts | Concerts |
+-------+----------+----------+
```

Fungsi REPLIKA

Sinonim untuk fungsi REPEAT.

Lihat Fungsi REPEAT.

Fungsi REVERSE

Fungsi REVERSE beroperasi pada string dan mengembalikan karakter dalam urutan terbalik. Misalnya, `reverse('abcde')` mengembalikan `edcba`. Fungsi ini bekerja pada tipe data numerik dan tanggal serta tipe data karakter; Namun, dalam kebanyakan kasus memiliki nilai praktis untuk string karakter.

Sintaksis

```
REVERSE( expression )
```

Pendapat

ekspresi

Ekspresi dengan karakter, tanggal, stempel waktu, atau tipe data numerik yang mewakili target pembalikan karakter. Semua ekspresi secara implisit dikonversi ke VARCHAR string. Bagian akhir yang kosong dalam CHAR string diabaikan.

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut menggunakan data dari tabel USERS dan SALES dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk memilih lima nama kota yang berbeda dan nama terbalik yang sesuai dari tabel USERS, gunakan contoh berikut.

```
SELECT DISTINCT city AS cityname, REVERSE(cityname)
FROM users
ORDER BY city LIMIT 5;

+----------+----------+
| cityname | reverse  |
+----------+----------+
| Aberdeen | needrebA |
| Abilene  | enelibA  |
| Ada      | adA      |
| Agat     | tagA     |
| Agawam   | mawagA   |
+----------+----------+
```

Untuk memilih lima ID penjualan dan ID terbalik yang sesuai yang digunakan sebagai string karakter, gunakan contoh berikut.

```
SELECT salesid, REVERSE(salesid)
```

```
FROM sales
ORDER BY salesid DESC LIMIT 5;

+---------+---------+
| salesid | reverse |
+---------+---------+
|  172456 |  654271 |
|  172455 |  554271 |
|  172454 |  454271 |
|  172453 |  354271 |
|  172452 |  254271 |
+---------+---------+
```

Fungsi RTRIM

Fungsi RTRIM memangkas serangkaian karakter tertentu dari akhir string. Menghapus string terpanjang yang hanya berisi karakter dalam daftar karakter trim. Pemangkasan selesai ketika karakter trim tidak muncul di string input.

Sintaksis

```
RTRIM( string, trim_chars )
```

Argumen

senar

Sebuah kolom string, ekspresi, atau string literal yang akan dipangkas.

trim_chars

Sebuah kolom string, ekspresi, atau string literal yang mewakili karakter yang akan dipangkas dari akhir string. Jika tidak ditentukan, spasi digunakan sebagai karakter trim.

Jenis pengembalian

Sebuah string yang merupakan tipe data yang sama dengan argumen string.

Contoh

Contoh berikut memangkas bagian depan dan belakang kosong dari string: `'  abc  '`

```
select '     abc    ' as untrim, rtrim('     abc    ') as trim;
```

```
untrim    | trim
----------+------
   abc    |    abc
```

Contoh berikut menghapus string trailing dari `'xyz'` string`'xyzaxyzbxyzcxyz'`. Kemunculan di belakang `'xyz'` dihapus, tetapi kejadian yang internal di dalam string tidak dihapus.

```
select 'xyzaxyzbxyzcxyz' as untrim,
rtrim('xyzaxyzbxyzcxyz', 'xyz') as trim;

     untrim      |   trim
-----------------+-----------
 xyzaxyzbxyzcxyz | xyzaxyzbxyzc
```

Contoh berikut menghapus bagian akhir dari string `'setuphistorycassettes'` yang cocok dengan salah satu karakter dalam daftar trim_ chars. `'tes'` Setiapt,e, atau s yang terjadi sebelum karakter lain yang tidak ada dalam daftar trim_chars di akhir string input dihapus.

```
SELECT rtrim('setuphistorycassettes', 'tes');

     rtrim
-----------------
 setuphistoryca
```

Contoh berikut memangkas karakter 'Park' dari akhir VENUENAME jika ada:

```
select venueid, venuename, rtrim(venuename, 'Park')
from venue
order by 1, 2, 3
limit 10;

venueid |         venuename          |          rtrim
--------+----------------------------+-------------------------
      1 | Toyota Park                | Toyota
      2 | Columbus Crew Stadium      | Columbus Crew Stadium
      3 | RFK Stadium                | RFK Stadium
      4 | CommunityAmerica Ballpark  | CommunityAmerica Ballp
      5 | Gillette Stadium           | Gillette Stadium
      6 | New York Giants Stadium    | New York Giants Stadium
      7 | BMO Field                  | BMO Field
```

```
      8 | The Home Depot Center      | The Home Depot Cente
      9 | Dick's Sporting Goods Park | Dick's Sporting Goods
     10 | Pizza Hut Park             | Pizza Hut
```

Perhatikan bahwa RTRIM menghapus salah satu karakterP,, `ar`, atau k ketika mereka muncul di akhir VENUENAME.

Fungsi SOUNDEX

Fungsi SOUNDEX mengembalikan nilai American Soundex yang terdiri dari huruf pertama dari string input diikuti dengan pengkodean 3 digit dari suara yang mewakili pengucapan bahasa Inggris dari string yang Anda tentukan. Misalnya, `Smith` dan `Smyth` memiliki nilai Soundex yang sama.

Sintaksis

```
SOUNDEX(string)
```

Argumen

senar

Anda menentukan `VARCHAR` string `CHAR` atau yang ingin Anda konversi ke nilai kode American Soundex.

Jenis pengembalian

VARCHAR(4)

Catatan penggunaan

Fungsi SOUNDEX hanya mengubah karakter ASCII huruf kecil dan huruf besar alfabet bahasa Inggris, termasuk a—z dan A—Z. SOUNDEX mengabaikan karakter lain. SOUNDEX mengembalikan nilai Soundex tunggal untuk string beberapa kata yang dipisahkan oleh spasi.

```
SELECT SOUNDEX('AWS Amazon');

+---------+
| soundex |
+---------+
| A252    |
+---------+
```

SOUNDEX mengembalikan string kosong jika string input tidak mengandung huruf bahasa Inggris.

```
SELECT SOUNDEX('+-*/%');

+---------+
| soundex |
+---------+
|         |
+---------+
```

Contoh

Untuk mengembalikan nilai Soundex untukAmazon, gunakan contoh berikut.

```
SELECT SOUNDEX('Amazon');

+---------+
| soundex |
+---------+
| A525    |
+---------+
```

Untuk mengembalikan nilai Soundex untuk smith dansmyth, gunakan contoh berikut. Perhatikan bahwa nilai Soundex sama.

```
SELECT SOUNDEX('smith'), SOUNDEX('smyth');

+-------+-------+
| smith | smyth |
+-------+-------+
| S530  | S530  |
+-------+-------+
```

Fungsi SPLIT_PART

Membagi string pada pembatas yang ditentukan dan mengembalikan bagian pada posisi yang ditentukan.

Sintaksis

```
SPLIT_PART(string, delimiter, position)
```

Argumen

senar

Sebuah kolom string, ekspresi, atau string literal yang akan dipisahkan. String dapat berupa CHAR atau VARCHAR.

pembatas

String pembatas yang menunjukkan bagian dari string input.

Jika pembatas adalah literal, lampirkan dalam tanda kutip tunggal.

posisi

Posisi bagian string yang akan dikembalikan (dihitung dari 1). Harus bilangan bulat lebih besar dari 0. Jika posisi lebih besar dari jumlah bagian string, SPLIT_PART mengembalikan string kosong. Jika pembatas tidak ditemukan dalam string, maka nilai yang dikembalikan berisi isi dari bagian yang ditentukan, yang mungkin seluruh string atau nilai kosong.

Jenis pengembalian

String CHAR atau VARCHAR, sama dengan parameter string.

Contoh

Contoh berikut membagi string literal menjadi beberapa bagian menggunakan pem $ batas dan mengembalikan bagian kedua.

```
select split_part('abc$def$ghi','$',2)

split_part
----------
def
```

Contoh berikut membagi string literal menjadi beberapa bagian menggunakan pembatas$. Ini mengembalikan string kosong karena bagian 4 tidak ditemukan.

```
select split_part('abc$def$ghi','$',4)

split_part
----------
```

Contoh berikut membagi string literal menjadi beberapa bagian menggunakan pembatas#. Ini mengembalikan seluruh string, yang merupakan bagian pertama, karena pembatas tidak ditemukan.

```
select split_part('abc$def$ghi','#',1)

split_part
------------
abc$def$ghi
```

Contoh berikut membagi bidang stempel waktu LISTTIME menjadi komponen tahun, bulan, dan hari.

```
select listtime, split_part(listtime,'-',1) as year,
split_part(listtime,'-',2) as month,
split_part(split_part(listtime,'-',3),' ',1) as day
from listing limit 5;

      listtime       | year | month | day
---------------------+------+-------+------
 2008-03-05 12:25:29 | 2008 | 03    | 05
 2008-09-09 08:03:36 | 2008 | 09    | 09
 2008-09-26 05:43:12 | 2008 | 09    | 26
 2008-10-04 02:00:30 | 2008 | 10    | 04
 2008-01-06 08:33:11 | 2008 | 01    | 06
```

Contoh berikut memilih bidang stempel waktu LISTTIME dan membaginya pada `'-'` karakter untuk mendapatkan bulan (bagian kedua dari string LISTTIME), lalu menghitung jumlah entri untuk setiap bulan:

```
select split_part(listtime,'-',2) as month, count(*)
from listing
group by split_part(listtime,'-',2)
order by 1, 2;

 month | count
-------+-------
    01 | 18543
    02 | 16620
    03 | 17594
    04 | 16822
```

```
05 | 17618
06 | 17158
07 | 17626
08 | 17881
09 | 17378
10 | 17756
11 | 12912
12 | 4589
```

Fungsi STRPOS

Mengembalikan posisi substring dalam string tertentu.

Lihat Fungsi CHARINDEX dan Fungsi POSISI untuk fungsi serupa.

Sintaksis

```
STRPOS(string, substring )
```

Argumen

senar

Parameter input pertama adalah VARCHAR string CHAR atau yang akan dicari.

substring

Parameter kedua adalah substring untuk mencari di dalam string.

Jenis pengembalian

INTEGER

Fungsi STRPOS mengembalikan yang INTEGER sesuai dengan posisi substring (berbasis satu, bukan berbasis nol). Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal.

Catatan penggunaan

STRPOS kembali 0 jika substring tidak ditemukan di dalam string.

```
SELECT STRPOS('dogfish', 'fist');
```

```
+--------+
| strpos |
+--------+
|      0 |
+--------+
```

Contoh

Untuk menunjukkan posisi di `fish` dalam`dogfish`, gunakan contoh berikut.

```
SELECT STRPOS('dogfish', 'fish');

+--------+
| strpos |
+--------+
|      4 |
+--------+
```

Contoh berikut menggunakan data dari tabel SALES dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengembalikan jumlah transaksi penjualan dengan KOMISI lebih dari 999,00 dari tabel PENJUALAN, gunakan contoh berikut.

```
SELECT DISTINCT STRPOS(commission, '.'),
COUNT (STRPOS(commission, '.'))
FROM sales
WHERE STRPOS(commission, '.') > 4
GROUP BY STRPOS(commission, '.')
ORDER BY 1, 2;

+--------+-------+
| strpos | count |
+--------+-------+
|      5 |   629 |
+--------+-------+
```

Fungsi STRTOL

Mengkonversi ekspresi string dari sejumlah basis yang ditentukan ke nilai integer yang setara. Nilai yang dikonversi harus berada dalam rentang 64-bit yang ditandatangani.

Sintaksis

```
STRTOL(num_string, base)
```

Argumen

nom_string

Ekspresi string dari angka yang akan dikonversi. Jika num_string kosong (`''`) atau dimulai dengan karakter null (`'\0'`), nilai yang dikonversi adalah. `0` Jika num_string adalah kolom yang berisi nilai NULL, STRTOL kembali. `NULL` String dapat dimulai dengan sejumlah spasi putih, secara opsional diikuti oleh tanda plus '+' atau minus '-' tunggal untuk menunjukkan positif atau negatif. Defaultnya adalah '+'. Jika basis adalah16, string dapat secara opsional dimulai dengan '`0x`'.

dasar

`INTEGER`antara 2 dan 36.

Jenis pengembalian

BIGINT

Jika num_string adalah null, fungsi kembali. `NULL`

Contoh

Untuk mengubah string dan pasangan nilai dasar menjadi bilangan bulat, gunakan contoh berikut.

```
SELECT STRTOL('0xf',16);

+--------+
| strtol |
+--------+
|     15 |
+--------+

SELECT STRTOL('abcd1234',16);

+------------+
|   strtol   |
+------------+
```

```
| 2882343476 |
+------------+

SELECT STRTOL('1234567', 10);

+---------+
| strtol  |
+---------+
| 1234567 |
+---------+

SELECT STRTOL('1234567', 8);

+--------+
| strtol |
+--------+
| 342391 |
+--------+

SELECT STRTOL('110101', 2);

+--------+
| strtol |
+--------+
|     53 |
+--------+

SELECT STRTOL('\0', 2);

+--------+
| strtol |
+--------+
|      0 |
+--------+
```

Fungsi SUBSTRING

Mengembalikan subset dari string berdasarkan posisi awal yang ditentukan.

Jika input adalah string karakter, posisi awal dan jumlah karakter yang diekstraksi didasarkan pada karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Jika input adalah ekspresi biner, posisi awal dan substring yang diekstraksi didasarkan pada byte. Anda tidak dapat menentukan panjang negatif, tetapi Anda dapat menentukan posisi awal negatif.

Sintaksis

```
SUBSTRING(character_string FROM start_position [ FOR number_characters ] )
```

```
SUBSTRING(character_string, start_position, number_characters )
```

```
SUBSTRING(binary_expression, start_byte, number_bytes )
```

```
SUBSTRING(binary_expression, start_byte )
```

Argumen

karakter_string

String yang akan dicari. Non-character tipe data diperlakukan seperti string.

posisi awal

Posisi dalam string untuk memulai ekstraksi, mulai dari 1. Start_position didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Angka ini bisa negatif.

angka_karakter

Jumlah karakter untuk mengekstrak (panjang substring). Number_character didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Angka ini tidak boleh negatif.

Ekspresi biner

Binary_expression dari tipe data VARBYTE yang akan dicari.

start_byte

Posisi dalam ekspresi biner untuk memulai ekstraksi, mulai dari 1. Angka ini bisa negatif.

nomor_byte

Jumlah byte untuk mengekstrak, yaitu panjang substring. Angka ini tidak boleh negatif.

Jenis pengembalian

VARCHAR atau VARBYTE tergantung pada input.

Catatan Penggunaan

Berikut ini adalah beberapa contoh bagaimana Anda dapat menggunakan start_position dan number_character untuk mengekstrak substring dari berbagai posisi dalam string.

Contoh berikut mengembalikan string empat karakter yang dimulai dengan karakter keenam.

```
select substring('caterpillar',6,4);
substring
-----------
pill
(1 row)
```

Jika start_position + number_character melebihi panjang string, SUBSTRING mengembalikan substring mulai dari start_ position sampai akhir string. Contoh:

```
select substring('caterpillar',6,8);
substring
-----------
pillar
(1 row)
```

Jika negatif atau 0, fungsi SUBSTRING mengembalikan substring yang dimulai pada karakter pertama string dengan panjang `start_position` + `number_characters` -1. `start_position` Contoh:

```
select substring('caterpillar',-2,6);
substring
-----------
cat
(1 row)
```

Jika `start_position` + `number_characters` -1 kurang dari atau sama dengan nol, SUBSTRING mengembalikan string kosong. Contoh:

```
select substring('caterpillar',-5,4);
substring
-----------

(1 row)
```

Contoh

Contoh berikut mengembalikan bulan dari string LISTTIME di tabel LISTING:

```
select listid, listtime,
substring(listtime, 6, 2) as month
from listing
order by 1, 2, 3
limit 10;

 listid |      listtime       | month
--------+---------------------+-------
      1 | 2008-01-24 06:43:29 | 01
      2 | 2008-03-05 12:25:29 | 03
      3 | 2008-11-01 07:35:33 | 11
      4 | 2008-05-24 01:18:37 | 05
      5 | 2008-05-17 02:29:11 | 05
      6 | 2008-08-15 02:08:13 | 08
      7 | 2008-11-15 09:38:15 | 11
      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
     10 | 2008-06-17 09:44:54 | 06
(10 rows)
```

Contoh berikut adalah sama seperti di atas, tetapi menggunakan opsi FROM... FOR:

```
select listid, listtime,
substring(listtime from 6 for 2) as month
from listing
order by 1, 2, 3
limit 10;

 listid |      listtime       | month
--------+---------------------+-------
      1 | 2008-01-24 06:43:29 | 01
      2 | 2008-03-05 12:25:29 | 03
      3 | 2008-11-01 07:35:33 | 11
      4 | 2008-05-24 01:18:37 | 05
      5 | 2008-05-17 02:29:11 | 05
      6 | 2008-08-15 02:08:13 | 08
      7 | 2008-11-15 09:38:15 | 11
      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
```

```
     10 | 2008-06-17 09:44:54 | 06
(10 rows)
```

Anda tidak dapat menggunakan SUBSTRING untuk mengekstrak awalan string yang mungkin berisi karakter multi-byte karena Anda perlu menentukan panjang string multi-byte berdasarkan jumlah byte, bukan jumlah karakter. Untuk mengekstrak segmen awal string berdasarkan panjang dalam byte, Anda dapat CAST string sebagai VARCHAR (byte_length) untuk memotong string, di mana byte_ length adalah panjang yang diperlukan. Contoh berikut mengekstrak 5 byte pertama dari string`'Fourscore and seven'`.

```
select cast('Fourscore and seven' as varchar(5));

varchar
-------
Fours
```

Contoh berikut menunjukkan posisi awal negatif dari nilai biner`abc`. Karena posisi awal adalah -3, substring diekstraksi dari awal nilai biner. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari substring biner.

```
select substring('abc'::varbyte, -3);

 substring
-----------
 616263
```

Contoh berikut menunjukkan 1 untuk posisi awal dari nilai biner`abc`. Karena karena tidak ada panjang yang ditentukan, string diekstraksi dari posisi awal ke akhir string. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari substring biner.

```
select substring('abc'::varbyte, 1);

 substring
-----------
 616263
```

Contoh berikut menunjukkan 3 untuk posisi awal dari nilai biner`abc`. Karena karena tidak ada panjang yang ditentukan, string diekstraksi dari posisi awal ke akhir string. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari substring biner.

```
select substring('abc'::varbyte, 3);

 substring
-----------
 63
```

Contoh berikut menunjukkan 2 untuk posisi awal dari nilai binerabc. String diekstraksi dari posisi awal ke posisi 10, tetapi ujung string berada di posisi 3. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari substring biner.

```
select substring('abc'::varbyte, 2, 10);

 substring
-----------
 6263
```

Contoh berikut menunjukkan 2 untuk posisi awal dari nilai binerabc. String diekstraksi dari posisi awal untuk 1 byte. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari substring biner.

```
select substring('abc'::varbyte, 2, 1);

 substring
-----------
 62
```

Contoh berikut mengembalikan nama depan Ana yang muncul setelah spasi terakhir dalam string inputSilva, Ana.

```
select reverse(substring(reverse('Silva, Ana'), 1, position(' ' IN reverse('Silva,
 Ana'))))

 reverse
-----------
 Ana
```

Fungsi TEXTLEN

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi TRANSLATE

Untuk ekspresi tertentu, ganti semua kemunculan karakter tertentu dengan pengganti yang ditentukan. Karakter yang ada dipetakan ke karakter pengganti berdasarkan posisinya dalam argumen characters_to_replace dan characters_to_substitution. Jika lebih banyak karakter ditentukan dalam argumen characters_to_replace daripada dalam argumen characters_to_ substitut, karakter tambahan dari argumen characters_to_replace dihilangkan dalam nilai yang dikem balikan.

TRANSLATE mirip dengan Fungsi REPLACE danFungsi REGEXP_REPLACE, kecuali bahwa REPLACE mengganti satu seluruh string dengan string lain dan dengan REGEXP_REPLACE, Anda dapat mencari string untuk pola ekspresi reguler, sementara TRANSLATE membuat beberapa substitusi karakter tunggal.

Jika ada argumen null, pengembaliannya adalahNULL.

Sintaksis

```
TRANSLATE( expression, characters_to_replace, characters_to_substitute )
```

Argumen

ekspresi

Ekspresi yang akan diterjemahkan.

karakter_tu_ganti_ganti

Sebuah string yang berisi karakter yang akan diganti.

karakter_to_tu_pengganti

Sebuah string yang berisi karakter yang akan diganti.

Jenis pengembalian

VARCHAR

Contoh

Untuk mengganti beberapa karakter dalam string, gunakan contoh berikut.

```
SELECT TRANSLATE('mint tea', 'inea', 'osin');

+-----------+
| translate |
+-----------+
| most tin  |
+-----------+
```

Contoh berikut menggunakan data dari tabel USERS dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengganti tanda at (@) dengan titik untuk semua nilai dalam kolom, gunakan contoh berikut.

```
SELECT email, TRANSLATE(email, '@', '.') as obfuscated_email
FROM users LIMIT 10;

+----------------------------------------+----------------------------------------+
|                 email                  |           obfuscated_email             |
+----------------------------------------+----------------------------------------+
| Cum@accumsan.com                       | Cum.accumsan.com                       |
| lorem.ipsum@Vestibulumante.com         | lorem.ipsum.Vestibulumante.com         |
| non.justo.Proin@ametconsectetuer.edu   | non.justo.Proin.ametconsectetuer.edu   |
| non.ante.bibendum@porttitortellus.org  | non.ante.bibendum.porttitortellus.org  |
| eros@blanditatnisi.org                 | eros.blanditatnisi.org                 |
| augue@Donec.ca                         | augue.Donec.ca                         |
| cursus@pedeacurna.edu                  | cursus.pedeacurna.edu                  |
| at@Duis.com                            | at.Duis.com                            |
| quam@facilisisvitaeorci.ca             | quam.facilisisvitaeorci.ca             |
| mi.lorem@nunc.edu                      | mi.lorem.nunc.edu                      |
+----------------------------------------+----------------------------------------+
```

Untuk mengganti spasi dengan garis bawah dan menghapus titik untuk semua nilai dalam kolom, gunakan contoh berikut.

```
SELECT city, TRANSLATE(city, ' .', '_')
FROM users
WHERE city LIKE 'Sain%' OR city LIKE 'St%'
GROUP BY city
ORDER BY city;

+----------------+---------------+
|      city      |   translate   |
```

```
+----------------+---------------+
| Saint Albans   | Saint_Albans  |
| Saint Cloud    | Saint_Cloud   |
| Saint Joseph   | Saint_Joseph  |
| Saint Louis    | Saint_Louis   |
| Saint Paul     | Saint_Paul    |
| St. George     | St_George     |
| St. Marys      | St_Marys      |
| St. Petersburg | St_Petersburg |
| Stafford       | Stafford      |
| Stamford       | Stamford      |
| Stanton        | Stanton       |
| Starkville     | Starkville    |
| Statesboro     | Statesboro    |
| Staunton       | Staunton      |
| Steubenville   | Steubenville  |
| Stevens Point  | Stevens_Point |
| Stillwater     | Stillwater    |
| Stockton       | Stockton      |
| Sturgis        | Sturgis       |
+----------------+---------------+
```

Fungsi TRIM

Memangkas string dengan karakter kosong atau tertentu.

Sintaksis

```
TRIM( [ BOTH | LEADING | TRAILING ] [trim_chars FROM ] string )
```

Argumen

KEDUANYA | MEMIMPIN | TERTINGGAL

(Opsional) Menentukan dari mana untuk memangkas karakter. Gunakan BOTH untuk menghapus karakter depan dan akhiran, gunakan LEADING untuk menghapus karakter utama saja, dan gunakan TRAILING untuk menghapus karakter trailing saja. Jika parameter ini dihilangkan, karakter depan dan belakang dipangkas.

trim_karakter

(Opsional) Karakter yang akan dipangkas dari string. Jika parameter ini dihilangkan, blanko dipangkas.

tali

Tali yang akan dipangkas.

Jenis pengembalian

Fungsi TRIM mengembalikan CHAR string VARCHAR atau. Jika Anda menggunakan fungsi TRIM dengan perintah SQL, Amazon Redshift secara implisit mengonversi hasilnya menjadi. VARCHAR Jika Anda menggunakan fungsi TRIM dalam daftar SELECT untuk fungsi SQL, Amazon Redshift tidak secara implisit mengonversi hasil, dan Anda mungkin perlu melakukan konversi eksplisit untuk menghindari kesalahan ketidakcocokan tipe data. Lihat Fungsi CONVERT fungsi Fungsi CAST dan untuk informasi tentang konversi eksplisit.

Contoh

Untuk memotong bagian depan dan belakang kosong dari string dog , gunakan contoh berikut.

```
SELECT TRIM('    dog    ');

+-------+
| btrim |
+-------+
| dog   |
+-------+
```

Untuk memotong bagian depan dan belakang kosong dari string dog , gunakan contoh berikut.

```
SELECT TRIM(BOTH FROM '    dog    ');

+-------+
| btrim |
+-------+
| dog   |
+-------+
```

Untuk menghapus tanda kutip ganda utama dari string"dog", gunakan contoh berikut.

```
SELECT TRIM(LEADING '"' FROM'"dog"');

+-------+
| ltrim |
```

```
+-------+
| dog"  |
+-------+
```

Untuk menghapus tanda kutip ganda yang tertinggal dari string"dog", gunakan contoh berikut.

```
SELECT TRIM(TRAILING '"' FROM'"dog"');

+-------+
| rtrim |
+-------+
| "dog  |
+-------+
```

TRIM menghapus salah satu karakter di trim_chars ketika mereka muncul di awal atau akhir string. Contoh berikut memangkas karakter 'C', 'D', dan 'G' ketika mereka muncul di awal atau akhir VENUENAME, yang merupakan kolom. VARCHAR Untuk informasi selengkapnya, lihat Database sampel.

```
SELECT venueid, venuename, TRIM('CDG' FROM venuename)
FROM venue
WHERE venuename LIKE '%Park'
ORDER BY 2
LIMIT 7;

+---------+----------------------------+----------------------------+
| venueid |         venuename          |           btrim            |
+---------+----------------------------+----------------------------+
|     121 | AT&T Park                  | AT&T Park                  |
|     109 | Citizens Bank Park         | itizens Bank Park          |
|     102 | Comerica Park              | omerica Park               |
|       9 | Dick's Sporting Goods Park | ick's Sporting Goods Park  |
|      97 | Fenway Park                | Fenway Park                |
|     112 | Great American Ball Park   | reat American Ball Park    |
|     114 | Miller Park                | Miller Park                |
+---------+----------------------------+----------------------------+
```

Fungsi ATAS

Mengkonversi string ke huruf besar. UPPER mendukung karakter UTF-8 multibyte, hingga maksimal empat byte per karakter.

Sintaksis

```
UPPER(string)
```

Argumen

tali

Parameter input adalah VARCHAR string atau tipe data lainnya, sepertiCHAR, yang dapat dikonversi secara implisit menjadiVARCHAR.

Jenis pengembalian

Fungsi UPPER mengembalikan string karakter yang merupakan tipe data yang sama dengan string input. Misalnya, fungsi akan mengembalikan VARCHAR string jika inputnya adalah VARCHAR string.

Contoh

Contoh berikut menggunakan data dari tabel CATEGORY dalam database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Untuk mengonversi bidang CATNAME menjadi huruf besar, gunakan yang berikut ini.

```
SELECT catname, UPPER(catname)
FROM category
ORDER BY 1,2;

+-----------+-----------+
|  catname  |   upper   |
+-----------+-----------+
| Classical | CLASSICAL |
| Jazz      | JAZZ      |
| MLB       | MLB       |
| MLS       | MLS       |
| Musicals  | MUSICALS  |
| NBA       | NBA       |
| NFL       | NFL       |
| NHL       | NHL       |
| Opera     | OPERA     |
| Plays     | PLAYS     |
| Pop       | POP       |
```

```
+-----------+-----------+
```

Fungsi informasi tipe SUPER

Berikut ini, Anda dapat menemukan deskripsi fungsi informasi tipe untuk SQL yang didukung Amazon Redshift untuk memperoleh informasi dinamis dari input tipe SUPER data.

Topik

- Fungsi DECIMAL_PRECISION
- Fungsi DECIMAL_SCALE
- Fungsi IS_ARRAY
- Fungsi IS_BIGINT
- Fungsi IS_BOOLEAN
- Fungsi IS_CHAR
- Fungsi IS_DECIMAL
- Fungsi IS_FLOAT
- Fungsi IS_INTEGER
- Fungsi IS_OBJECT
- Fungsi IS_SCALAR
- Fungsi IS_SMALLINT
- Fungsi IS_VARCHAR
- Fungsi JSON_SIZE
- Fungsi JSON_TYPEOF
- UKURAN

Fungsi DECIMAL_PRECISION

Memeriksa ketepatan jumlah maksimum digit desimal yang akan disimpan. Angka ini mencakup digit kiri dan kanan dari titik desimal. Kisaran presisi adalah dari 1 hingga 38, dengan default 38.

Sintaksis

```
DECIMAL_PRECISION(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

INTEGER

Contoh

Untuk menerapkan fungsi DECIMAL_PRECISION ke tabel t, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_PRECISION(s) FROM t;

+-------------------+
| decimal_precision |
+-------------------+
|                 6 |
+-------------------+
```

Fungsi DECIMAL_SCALE

Memeriksa jumlah digit desimal yang akan disimpan di sebelah kanan titik desimal. Rentang skala adalah dari 0 ke titik presisi, dengan default 0.

Sintaksis

```
DECIMAL_SCALE(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

`INTEGER`

Contoh

Untuk menerapkan fungsi DECIMAL_SCALE ke tabel t, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_SCALE(s) FROM t;

+---------------+
| decimal_scale |
+---------------+
|             5 |
+---------------+
```

Fungsi IS_ARRAY

Memeriksa apakah variabel adalah array. Fungsi kembali `true` jika variabel adalah array. Fungsi ini juga mencakup array kosong. Jika tidak, fungsi kembali `false` untuk semua nilai lainnya, termasuk null.

Sintaksis

```
IS_ARRAY(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `[1,2]` apakah array menggunakan fungsi IS_ARRAY, gunakan contoh berikut.

```
SELECT IS_ARRAY(JSON_PARSE('[1,2]'));

+----------+
| is_array |
+----------+
| true     |
+----------+
```

Fungsi IS_BIGINT

Memeriksa apakah suatu nilai adalah a`BIGINT`. Fungsi IS_BIGINT mengembalikan `true` untuk jumlah skala 0 dalam rentang 64-bit. Jika tidak, fungsi kembali `false` untuk semua nilai lainnya, termasuk angka nol dan float point.

Fungsi IS_BIGINT adalah superset dari IS_INTEGER.

Sintaksis

```
IS_BIGINT(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa 5 apakah `BIGINT` menggunakan fungsi IS_BIGINT, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_BIGINT(s) FROM t;
```

```
+---+-----------+
| s | is_bigint |
+---+-----------+
| 5 | true      |
+---+-----------+
```

Fungsi IS_BOOLEAN

Memeriksa apakah suatu nilai adalah a`BOOLEAN`. Fungsi IS_BOOLEAN kembali `true` untuk Boolean JSON konstan. Fungsi mengembalikan `false` nilai lainnya, termasuk null.

Sintaksis

```
IS_BOOLEAN(super_expression)
```

Argumen

super_ekspresi

Eksp `SUPER` resi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `TRUE` apakah `BOOLEAN` menggunakan fungsi IS_BOOLEAN, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (TRUE);

SELECT s, IS_BOOLEAN(s) FROM t;

+------+------------+
|  s   | is_boolean |
+------+------------+
| true | true       |
+------+------------+
```

Fungsi IS_CHAR

Memeriksa apakah suatu nilai adalah a`CHAR`. Fungsi IS_CHAR mengembalikan `true` untuk string yang hanya memiliki karakter ASCII, karena tipe CHAR hanya dapat menyimpan karakter yang berada dalam format ASCII. Fungsi kembali `false` untuk nilai lainnya.

Sintaksis

```
IS_CHAR(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `t` apakah `CHAR` menggunakan fungsi IS_CHAR, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('t');

SELECT s, IS_CHAR(s) FROM t;

+-----+---------+
|  s  | is_char |
+-----+---------+
| "t" | true    |
+-----+---------+
```

Fungsi IS_DECIMAL

Memeriksa apakah suatu nilai adalah a`DECIMAL`. Fungsi IS_DECIMAL mengembalikan `true` angka yang bukan titik mengambang. Fungsi mengembalikan `false` nilai lainnya, termasuk null.

Fungsi IS_DECIMAL adalah superset dari IS_BIGINT.

Sintaksis

```
IS_DECIMAL(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa 1.22 apakah DECIMAL menggunakan fungsi IS_DECIMAL, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (1.22);

SELECT s, IS_DECIMAL(s) FROM t;

+------+------------+
|  s   | is_decimal |
+------+------------+
| 1.22 | true       |
+------+------------+
```

Fungsi IS_FLOAT

Memeriksa apakah suatu nilai adalah angka float point. Fungsi IS_FLOAT kembali `true` untuk bilangan titik mengambang (FLOAT4danFLOAT8). Fungsi kembali `false` untuk nilai lainnya.

Himpunan IS_DECIMAL himpunan IS_FLOAT terputus-putus.

Sintaksis

```
IS_FLOAT(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa `2.22::FLOAT` apakah FLOAT menggunakan fungsi IS_FLOAT, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES(2.22::FLOAT);

SELECT s, IS_FLOAT(s) FROM t;

+---------+----------+
|    s    | is_float |
+---------+----------+
| 2.22e+0 | true     |
+---------+----------+
```

Fungsi IS_INTEGER

Meng `true` embalikan untuk angka skala 0 dalam rentang 32-bit, dan `false` untuk hal lain (termasuk angka nol dan float point).

Fungsi IS_INTEGER adalah superset dari fungsi IS_SMALLINT.

Sintaksis

```
IS_INTEGER(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa 5 apakah INTEGER menggunakan fungsi IS_INTEGER, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_INTEGER(s) FROM t;

+---+------------+
| s | is_integer |
+---+------------+
| 5 | true       |
+---+------------+
```

Fungsi IS_OBJECT

Memeriksa apakah variabel adalah objek. Fungsi IS_OBJECT mengembalikan `true` untuk objek, termasuk objek kosong. Fungsi mengembalikan `false` nilai lainnya, termasuk null.

Sintaksis

```
IS_OBJECT(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa `{"name": "Joe"}` apakah objek menggunakan fungsi IS_OBJECT, gunakan contoh berikut.

```
CREATE TABLE t(s super);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_OBJECT(s) FROM t;

+----------------+-----------+
|       s        | is_object |
+----------------+-----------+
| {"name":"Joe"} | true      |
+----------------+-----------+
```

Fungsi IS_SCALAR

Memeriksa apakah variabel adalah skalar. Fungsi IS_SCALAR mengembalikan `true` untuk nilai apa pun yang bukan array atau objek. Fungsi mengembalikan `false` nilai lainnya, termasuk null.

Himpunan IS_ARRAY, IS_OBJECT, dan IS_SCALAR mencakup semua nilai kecuali nol.

Sintaksis

```
IS_SCALAR(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa `{"name": "Joe"}` apakah skalar menggunakan fungsi IS_SCALAR, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));
```

```
SELECT s, IS_SCALAR(s.name) FROM t;

+----------------+-----------+
|       s        | is_scalar |
+----------------+-----------+
| {"name":"Joe"} | true      |
+----------------+-----------+
```

Fungsi IS_SMALLINT

Memeriksa apakah variabel adalah a`SMALLINT`. Fungsi IS_SMALLINT mengembalikan `true` angka skala 0 dalam rentang 16-bit. Fungsi mengembalikan `false` nilai lainnya, termasuk angka nol dan float point.

Sintaksis

```
IS_SMALLINT(super_expression)
```

Argumen

super_ekspresi

Eksp `SUPER` resi atau kolom.

Nilai yang ditampilkan

`BOOLEAN`

Contoh

Untuk memeriksa 5 apakah `SMALLINT` menggunakan fungsi IS_SMALLINT, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_SMALLINT(s) FROM t;

+---+-------------+
| s | is_smallint |
```

```
+---+-------------+
| 5 | true        |
+---+-------------+
```

Fungsi IS_VARCHAR

Memeriksa apakah variabel adalah a`VARCHAR`. Fungsi IS_VARCHAR mengembalikan `true` untuk semua string. Fungsi kembali `false` untuk nilai lainnya.

Fungsi IS_VARCHAR adalah superset dari fungsi IS_CHAR.

Sintaksis

```
IS_VARCHAR(super_expression)
```

Argumen

super_ekspresi

Eksp `SUPER` resi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `abc` apakah `VARCHAR` menggunakan fungsi IS_VARCHAR, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('abc');

SELECT s, IS_VARCHAR(s) FROM t;

+-------+------------+
|   s   | is_varchar |
+-------+------------+
| "abc" | true       |
+-------+------------+
```

Fungsi JSON_SIZE

Fungsi JSON_SIZE mengembalikan jumlah byte dalam SUPER ekspresi yang diberikan ketika diserialkan menjadi string.

Sintaksis

```
JSON_SIZE(super_expression)
```

Argumen

super_ekspresi

Kon SUPER stanta atau ekspresi.

Jenis pengembalian

INTEGER

Fungsi JSON_SIZE mengembalikan INTEGER menunjukkan jumlah byte dalam string input. Nilai ini berbeda dari jumlah karakter. Misalnya, UTF-8 karakter, titik hitam, berukuran 3 byte meskipun 1 karakter.

Catatan penggunaan

JSON_SIZE (x) secara fungsional identik dengan OCTET_LENGTH (JSON_SERIALIZE). Namun, perhatikan bahwa JSON_SERIALIZE mengembalikan kesalahan ketika SUPER ekspresi yang disediakan akan melebihi VARCHAR batas sistem saat diserialkan. JSON_SIZE tidak memiliki batasan ini.

Contoh

Untuk mengembalikan panjang SUPER nilai yang diserialkan ke string, gunakan contoh berikut.

```
SELECT JSON_SIZE(JSON_PARSE('[10001,10002,"#"]'));

+-----------+
| json_size |
+-----------+
|        19 |
+-----------+
```

Perhatikan bahwa SUPER ekspresi yang disediakan panjangnya 17 karakter, tetapi adalah karakter 3-byte, jadi JSON_SIZE kembali. 19

Fungsi JSON_TYPEOF

Fungsi skalar JSON_TYPEOF mengembalikan a VARCHAR dengan nilai boolean, number, string, object, array, atau null, tergantung pada tipe dinamis dari nilai. SUPER

Sintaksis

```
JSON_TYPEOF(super_expression)
```

Argumen

super_ekspresi

Eksp SUPER resi atau kolom.

Jenis pengembalian

VARCHAR

Contoh

Untuk memeriksa jenis JSON untuk array [1,2] menggunakan fungsi JSON_TYPEOF, gunakan contoh berikut.

```
SELECT JSON_TYPEOF(ARRAY(1,2));

+-------------+
| json_typeof |
+-------------+
| array       |
+-------------+
```

Untuk memeriksa jenis JSON untuk objek {"name":"Joe"} menggunakan fungsi JSON_TYPEOF, gunakan contoh berikut.

```
SELECT JSON_TYPEOF(JSON_PARSE('{"name":"Joe"}'));

+-------------+
| json_typeof |
```

```
+-------------+
| object      |
+-------------+
```

UKURAN

Mengembalikan ukuran biner dalam memori dari konstanta SUPER tipe atau ekspresi sebagaiINTEGER.

Sintaksis

```
SIZE(super_expression)
```

Argumen

super_ekspresi

Kon SUPER stanta tipe atau ekspresi.

Jenis pengembalian

INTEGER

Contoh

Untuk menggunakan SIZE untuk mendapatkan ukuran dalam memori dari beberapa ekspresi SUPER tipe, gunakan contoh berikut.

```
CREATE TABLE test_super_size(a SUPER);

INSERT INTO test_super_size
VALUES
  (null),
  (TRUE),
  (JSON_PARSE('[0,1,2,3]')),
  (JSON_PARSE('{"a":0,"b":1,"c":2,"d":3}'))
;

SELECT a, SIZE(a)
FROM test_super_size
ORDER BY 2, 1;
```

```
+---------------------------+------+
|             a             | size |
+---------------------------+------+
| true                      |    4 |
| NULL                      |    4 |
| [0,1,2,3]                 |   23 |
| {"a":0,"b":1,"c":2,"d":3} |   52 |
+---------------------------+------+
```

Fungsi dan operator VARBYTE

Fungsi Amazon Redshift dan operator yang mendukung tipe data VARBYTE meliputi:

- Operator VARBYTE
- DARI_HEX
- DARI_VARBYTE
- GETBIT
- TO_HEX
- TO_VARBYTE
- MEMBANTAH
- LENS
- Fungsi LENGTH
- OCTET_LENGTH
- Fungsi SUBSTRING

Operator VARBYTE

Tabel berikut mencantumkan operator VARBYTE. Operator bekerja dengan nilai biner tipe data VARBYTE. Jika salah satu atau kedua input adalah nol, hasilnya adalah nol.

Operator yang didukung

Operator	Deskripsi	Jenis pengembalian
<	Kurang dari	BOOLEAN

Operator	Deskripsi	Jenis pengembalian
<=	Kurang dari atau sama	BOOLEAN
=	Sama	BOOLEAN
>	Lebih besar dari	BOOLEAN
>=	Lebih besar dari atau sama	BOOLEAN
! = atau <>	Tidak sama	BOOLEAN
||	Rangkaian	VARBYTE
+	Rangkaian	VARBYTE
~	Bitwise tidak	VARBYTE
&	Bitwise dan	VARBYTE
|	Bitwise atau	VARBYTE
#	Bitwise xor	VARBYTE

Contoh

Dalam contoh berikut, nilai `'a'::VARBYTE` is 61 dan nilai `'b'::VARBYTE` is62. Itu `::` melemparkan string ke tipe VARBYTE data. Untuk informasi selengkapnya tentang tipe data casting, lihatPEMERAN.

Untuk membandingkan `'a'` jika kurang dari `'b'` menggunakan < operator, gunakan contoh berikut.

```
SELECT 'a'::VARBYTE < 'b'::VARBYTE AS less_than;

+-----------+
| less_than |
+-----------+
| true      |
+-----------+
```

Untuk membandingkan jika `'a'` sama `'b'` dengan menggunakan = operator, gunakan contoh berikut.

```
SELECT 'a'::VARBYTE = 'b'::VARBYTE AS equal;

+-------+
| equal |
+-------+
| false |
+-------+
```

Untuk menggabungkan dua nilai biner menggunakan `||` operator, gunakan contoh berikut.

```
SELECT 'a'::VARBYTE || 'b'::VARBYTE AS concat;

+--------+
| concat |
+--------+
|   6162 |
+--------+
```

Untuk menggabungkan dua nilai biner menggunakan + operator, gunakan contoh berikut.

```
SELECT 'a'::VARBYTE + 'b'::VARBYTE AS concat;

+--------+
| concat |
+--------+
|   6162 |
+--------+
```

Untuk meniadakan setiap bit dari nilai biner input menggunakan fungsi FROM_VARBYTE, gunakan contoh berikut. String `'a'` mengevaluasi. `01100001` Untuk informasi selengkapnya, lihat DARI_VARBYTE.

```
SELECT FROM_VARBYTE(~'a'::VARBYTE, 'binary');

+--------------+
| from_varbyte |
+--------------+
|     10011110 |
+--------------+
```

Untuk menerapkan & operator pada dua nilai biner input, gunakan contoh berikut. String `'a'` mengevaluasi `01100001` dan `'b'` mengevaluasi. `01100010`

```
SELECT FROM_VARBYTE('a'::VARBYTE & 'b'::VARBYTE, 'binary');

+--------------+
| from_varbyte |
+--------------+
|     01100000 |
+--------------+
```

Fungsi FROM_HEX

FROM_HEX mengkonversi heksadesimal ke nilai biner.

Sintaksis

```
FROM_HEX(hex_string)
```

Argumen

hex_string

String heksadesimal tipe data VARCHAR atau TEXT yang akan dikonversi. Formatnya harus berupa nilai literal.

Jenis pengembalian

VARBYTE

Contoh

Untuk mengkonversi representasi heksadesimal `'6162'` ke nilai biner, gunakan contoh berikut. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari nilai biner.

```
SELECT FROM_HEX('6162');

+----------+
| from_hex |
+----------+
|     6162 |
+----------+
```

Fungsi FROM_VARBYTE

FROM_VARBYTE mengkonversi nilai biner ke string karakter dalam format yang ditentukan.

Sintaksis

```
FROM_VARBYTE(binary_value, format)
```

Argumen

binary_value

Nilai biner dari tipe dataVARBYTE.

format

Format string karakter yang dikembalikan. Nilai valid yang tidak peka huruf besar/kecil adalah `hex`binary,, `utf8` (juga `utf-8` dan`utf_8`), dan`base64`.

Jenis pengembalian

`VARCHAR`

Contoh

Untuk mengonversi nilai biner `'ab'` menjadi heksadesimal, gunakan contoh berikut.

```
SELECT FROM_VARBYTE('ab', 'hex');
```

```
+--------------+
| from_varbyte |
+--------------+
|         6162 |
+--------------+
```

Untuk mengembalikan representasi biner `'4d'`, gunakan contoh berikut. Representasi biner dari `'4d'` adalah string karakter`01001101`.

```
SELECT FROM_VARBYTE(FROM_HEX('4d'), 'binary');

+--------------+
| from_varbyte |
+--------------+
|     01001101 |
+--------------+
```

Fungsi GETBIT

GETBIT mengembalikan nilai bit dari nilai biner pada indeks yang ditentukan.

Sintaksis

```
GETBIT(binary_value, index)
```

Argumen

binary_value

Nilai biner dari tipe data`VARBYTE`.

indeks

Nomor indeks bit dalam nilai biner yang dikembalikan. Nilai biner adalah array bit berbasis 0 yang diindeks dari bit paling kanan (bit paling tidak signifikan) ke bit paling kiri (bit paling signifikan).

Jenis pengembalian

`INTEGER`

Contoh

Untuk mengembalikan bit pada indeks 2 nilai binerfrom_hex('4d'), gunakan contoh berikut. Representasi biner dari '4d' adalah01001101.

```
SELECT GETBIT(FROM_HEX('4d'), 2);

+--------+
| getbit |
+--------+
|      1 |
+--------+
```

Untuk mengembalikan bit di delapan lokasi indeks dari nilai biner yang dikembalikan olehfrom_hex('4d'), gunakan contoh berikut. Representasi biner dari '4d' adalah01001101.

```
SELECT GETBIT(FROM_HEX('4d'), 7), GETBIT(FROM_HEX('4d'), 6),
  GETBIT(FROM_HEX('4d'), 5), GETBIT(FROM_HEX('4d'), 4),
  GETBIT(FROM_HEX('4d'), 3), GETBIT(FROM_HEX('4d'), 2),
  GETBIT(FROM_HEX('4d'), 1), GETBIT(FROM_HEX('4d'), 0);

+--------+--------+--------+--------+--------+--------+--------+--------+
| getbit | getbit | getbit | getbit | getbit | getbit | getbit | getbit |
+--------+--------+--------+--------+--------+--------+--------+--------+
|      0 |      1 |      0 |      0 |      1 |      1 |      0 |      1 |
+--------+--------+--------+--------+--------+--------+--------+--------+
```

Fungsi TO_HEX

TO_HEX mengkonversi angka atau nilai biner ke representasi heksadesimal.

Sintaksis

```
TO_HEX(value)
```

Argumen

nilai

Entah angka atau nilai biner (VARBYTE) yang akan dikonversi.

Jenis pengembalian

VARCHAR

Contoh

Untuk mengonversi angka ke representasi heksadesimal, gunakan contoh berikut.

```
SELECT TO_HEX(2147676847);

+----------+
|  to_hex  |
+----------+
| 8002f2af |
+----------+
```

Untuk mengonversi VARBYTE representasi `'abc'` menjadi bilangan heksadesimal, gunakan contoh followign.

```
SELECT TO_HEX('abc'::VARBYTE);

+--------+
| to_hex |
+--------+
| 616263 |
+--------+
```

Untuk membuat tabel, masukkan VARBYTE representasi `'abc'` ke nomor heksadesimal, dan pilih kolom dengan nilai, gunakan contoh berikut.

```
CREATE TABLE t (vc VARCHAR);
INSERT INTO t SELECT TO_HEX('abc'::VARBYTE);
SELECT vc FROM t;

+--------+
|   vc   |
+--------+
| 616263 |
+--------+
```

Untuk menunjukkan bahwa saat VARBYTE VARCHAR mentransmisikan nilai ke format UTF-8, gunakan contoh berikut.

```
CREATE TABLE t (vc VARCHAR);
INSERT INTO t SELECT 'abc'::VARBYTE::VARCHAR;

SELECT vc FROM t;

+-----+
| vc  |
+-----+
| abc |
+-----+
```

Fungsi TO_VARBYTE

TO_VARBYTE mengkonversi string dalam format tertentu ke nilai biner.

Sintaksis

```
TO_VARBYTE(string, format)
```

Argumen

tali

A CHAR atau VARCHAR string.

format

Format string input. Nilai valid yang tidak peka huruf besar/kecil adalah `hexbinary`,, `utf8` (juga `utf-8` dan`utf_8`), dan`base64`.

Jenis pengembalian

VARBYTE

Contoh

Untuk mengkonversi hex 6162 ke nilai biner, gunakan contoh berikut. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari nilai biner.

```
SELECT TO_VARBYTE('6162', 'hex');
```

```
+------------+
| to_varbyte |
+------------+
|       6162 |
+------------+
```

Untuk mengembalikan representasi biner4d, gunakan contoh berikut. Representasi biner dari '4d' adalah01001101.

```
SELECT TO_VARBYTE('01001101', 'binary');

+------------+
| to_varbyte |
+------------+
|         4d |
+------------+
```

Untuk mengkonversi string `'a'` UTF-8 ke nilai biner, gunakan contoh berikut. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari nilai biner.

```
SELECT TO_VARBYTE('a', 'utf8');

+------------+
| to_varbyte |
+------------+
|         61 |
+------------+
```

Untuk mengkonversi string `'4'` dalam heksadesimal ke nilai biner, gunakan contoh berikut. Jika panjang string heksadesimal adalah angka ganjil, maka a `0` ditambahkan untuk membentuk angka heksadesimal yang valid.

```
SELECT TO_VARBYTE('4', 'hex');

+------------+
| to_varbyte |
+------------+
|         04 |
+------------+
```

Fungsi jendela

Dengan menggunakan fungsi jendela, Anda dapat membuat kueri bisnis analitik dengan lebih efisien. Fungsi jendela beroperasi pada partisi atau "jendela" dari kumpulan hasil, dan mengembalikan nilai untuk setiap baris di jendela itu. Sebaliknya, fungsi non-jendela melakukan perhitungan mereka sehubungan dengan setiap baris dalam kumpulan hasil. Tidak seperti fungsi grup yang menggabungkan baris hasil, fungsi jendela mempertahankan semua baris dalam ekspresi tabel.

Nilai yang dikembalikan dihitung dengan menggunakan nilai dari kumpulan baris di jendela itu. Untuk setiap baris dalam tabel, jendela mendefinisikan satu set baris yang digunakan untuk menghitung atribut tambahan. Sebuah jendela didefinisikan menggunakan spesifikasi jendela (klausa OVER), dan didasarkan pada tiga konsep utama:

- Partisi jendela, yang membentuk kelompok baris (klausa PARTISI)
- Pemesanan jendela, yang mendefinisikan urutan atau urutan baris dalam setiap partisi (klausa ORDER BY)
- Bingkai jendela, yang didefinisikan relatif terhadap setiap baris untuk lebih membatasi kumpulan baris (spesifikasi ROWS)

Fungsi jendela adalah kumpulan operasi terakhir yang dilakukan dalam kueri kecuali untuk klausa ORDER BY akhir. Semua gabungan dan semua klausa WHERE, GROUP BY, dan HAVING diselesaikan sebelum fungsi jendela diproses. Oleh karena itu, fungsi jendela hanya dapat muncul dalam daftar pilih atau klausa ORDER BY. Anda dapat menggunakan beberapa fungsi jendela dalam satu kueri dengan klausa bingkai yang berbeda. Anda juga dapat menggunakan fungsi jendela dalam ekspresi skalar lainnya, seperti CASE.

Fungsi jendela tidak dapat bersarang. Misalnya, fungsi agregat JUMLAH dapat muncul di dalam fungsi jendelaJUMLAH, tetapi fungsi jendela SUM tidak dapat muncul di dalam fungsi jendela lain SUM. Berikut ini tidak didukung karena fungsi jendela bersarang dalam fungsi jendela lain.

```
SELECT SUM(SUM(selectcol) OVER (PARTITION BY ordercol)) OVER (Partition by ordercol)
 FROM t;
```

Ringkasan sintaks fungsi jendela

Fungsi jendela mengikuti sintaks standar, yaitu sebagai berikut.

```
function (expression) OVER (
```

```
[ PARTITION BY expr_list ]
[ ORDER BY order_list [ frame_clause ] ] )
```

Di sini, fungsi adalah salah satu fungsi yang dijelaskan di bagian ini.

Expr_list adalah sebagai berikut.

```
expression | column_name [, expr_list ]
```

Order_list adalah sebagai berikut.

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

Frame_Clause adalah sebagai berikut.

```
ROWS
{ UNBOUNDED PRECEDING | unsigned_value PRECEDING | CURRENT ROW } |

{ BETWEEN
{ UNBOUNDED PRECEDING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW}
AND
{ UNBOUNDED FOLLOWING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW }}
```

Argumen

fungsi

Fungsi jendela. Untuk detailnya, lihat deskripsi fungsi individu.

DI ATAS

Klausa yang mendefinisikan spesifikasi jendela. Klausa OVER adalah wajib untuk fungsi jendela, dan membedakan fungsi jendela dari fungsi SQL lainnya.

PARTISI OLEH expr_list

(Opsional) Klausa PARTITION BY membagi hasil yang ditetapkan menjadi partisi, seperti klausa GROUP BY. Jika klausa partisi hadir, fungsi dihitung untuk baris di setiap partisi. Jika tidak ada klausa partisi yang ditentukan, partisi tunggal berisi seluruh tabel, dan fungsi dihitung untuk tabel lengkap itu.

Fungsi peringkat DENSE_RANK, NTILE, RANK, dan ROW_NUMBER memerlukan perbandingan global dari semua baris dalam kumpulan hasil. Ketika klausa PARTITION BY digunakan, pengoptimal kueri dapat menjalankan setiap agregasi secara paralel dengan menyebarkan beban kerja di beberapa irisan sesuai dengan partisi. Jika klausa PARTITION BY tidak ada, langkah agregasi harus dijalankan secara serial pada satu irisan, yang dapat memiliki dampak negatif yang signifikan pada kinerja, terutama untuk cluster besar.

Amazon Redshift tidak mendukung literal string dalam klausa PARTITION BY.

PESANAN BERDASARKAN order_list

(Opsional) Fungsi jendela diterapkan ke baris dalam setiap partisi yang diurutkan sesuai dengan spesifikasi pesanan di ORDER BY. Klausa ORDER BY ini berbeda dari dan sama sekali tidak terkait dengan klausa ORDER BY dalam frame_ clause. Klausa ORDER BY dapat digunakan tanpa klausa PARTITION BY.

Untuk fungsi peringkat, klausa ORDER BY mengidentifikasi ukuran untuk nilai peringkat. Untuk fungsi agregasi, baris yang dipartisi harus diurutkan sebelum fungsi agregat dihitung untuk setiap frame. Untuk selengkapnya tentang jenis fungsi jendela, lihatFungsi jendela.

Pengidentifikasi kolom atau ekspresi yang mengevaluasi pengidentifikasi kolom diperlukan dalam daftar urutan. Konstanta maupun ekspresi konstan tidak dapat digunakan sebagai pengganti nama kolom.

Nilai NULLS diperlakukan sebagai grup mereka sendiri, diurutkan dan diberi peringkat sesuai dengan opsi NULLS FIRST atau NULLS LAST. Secara default, nilai NULL diurutkan dan diberi peringkat terakhir dalam urutan ASC, dan diurutkan dan diberi peringkat pertama dalam urutan DESC.

Amazon Redshift tidak mendukung literal string dalam klausa ORDER BY.

Jika klausa ORDER BY dihilangkan, urutan baris tidak deterministik.

> **Note**
>
> Dalam sistem paralel apa pun seperti Amazon Redshift, ketika klausa ORDER BY tidak menghasilkan urutan data yang unik dan total, urutan baris bersifat nondeterministik. Artinya, jika ekspresi ORDER BY menghasilkan nilai duplikat (urutan parSIAL), urutan pengembalian baris tersebut mungkin berbeda dari satu proses Amazon Redshift ke yang berikutnya. Pada gilirannya, fungsi jendela mungkin mengembalikan hasil yang tidak

terduga atau tidak konsisten. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

nama_kolom

Nama kolom yang akan dipartisi oleh atau diurutkan berdasarkan.

ASC | DESKRIPSI

Opsi yang mendefinisikan urutan untuk ekspresi, sebagai berikut:

- ASC: naik (misalnya, rendah ke tinggi untuk nilai numerik dan 'A' ke 'Z' untuk string karakter). Jika tidak ada opsi yang ditentukan, data diurutkan dalam urutan menaik secara default.
- DESC: menurun (tinggi ke rendah untuk nilai numerik; 'Z' ke 'A' untuk string).

NULLS FIRST | NULL TERAKHIR

Opsi yang menentukan apakah NULLS harus diurutkan terlebih dahulu, sebelum nilai non-null, atau terakhir, setelah nilai non-null. Secara default, NULLS diurutkan dan diberi peringkat terakhir dalam pengurutan ASC, dan diurutkan dan diberi peringkat pertama dalam pemesanan DESC.

frame_klausa

Untuk fungsi agregat, klausa bingkai lebih lanjut menyempurnakan kumpulan baris di jendela fungsi saat menggunakan ORDER BY. Dengan klausa bingkai, Anda dapat menyertakan atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait.

Klausa frame tidak berlaku untuk fungsi peringkat. Juga, klausa frame tidak diperlukan ketika tidak ada klausa ORDER BY yang digunakan dalam klausa OVER untuk fungsi agregat. Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan.

Ketika tidak ada klausa ORDER BY yang ditentukan, frame tersirat tidak terbatas, setara dengan ROWS BETWEEN UNBOUNDED PRECEDING DAN UNBOUNDED FOLLOWING.

BARIS

Klausa ini mendefinisikan bingkai jendela dengan menentukan offset fisik dari baris saat ini.

Klausa ini menentukan baris di jendela atau partisi saat ini yang akan digabungkan dengan nilai di baris saat ini. Ini menggunakan argumen yang menentukan posisi baris, yang bisa sebelum atau sesudah baris saat ini. Titik referensi untuk semua bingkai jendela adalah baris saat ini. Setiap baris menjadi baris saat ini secara bergantian saat bingkai jendela meluncur ke depan di partisi.

Bingkai dapat berupa sekumpulan baris sederhana hingga dan termasuk baris saat ini.

```
{UNBOUNDED PRECEDING | offset PRECEDING | CURRENT ROW}
```

Atau bisa berupa sekumpulan baris antara dua batas.

```
BETWEEN
{ UNBOUNDED PRECEDING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
AND
{ UNBOUNDED FOLLOWING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
```

UNBOUNDED PRECEDING menunjukkan bahwa jendela dimulai pada baris pertama partisi; offset PRECEDING menunjukkan bahwa jendela memulai sejumlah baris yang setara dengan nilai offset sebelum baris saat ini. UNBOUNDED PRECEDING adalah default.

BARIS SAAT INI menunjukkan jendela dimulai atau berakhir pada baris saat ini.

UNBOUNDED FOLLOWING menunjukkan bahwa jendela berakhir di baris terakhir partisi; offset FOLLOWING menunjukkan bahwa jendela mengakhiri sejumlah baris yang setara dengan nilai offset setelah baris saat ini.

offset mengidentifikasi jumlah fisik baris sebelum atau sesudah baris saat ini. Dalam hal ini, offset harus menjadi konstanta yang mengevaluasi nilai numerik positif. Misalnya, 5 FOLLOWING mengakhiri bingkai lima baris setelah baris saat ini.

Dimana BETWEEN tidak ditentukan, frame secara implisit dibatasi oleh baris saat ini. Misalnya, `ROWS 5 PRECEDING` sama dengan`ROWS BETWEEN 5 PRECEDING AND CURRENT ROW`. Juga, `ROWS UNBOUNDED FOLLOWING` sama dengan`ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING`.

> **Note**
>
> Anda tidak dapat menentukan bingkai di mana batas awal lebih besar dari batas akhir. Misalnya, Anda tidak dapat menentukan salah satu frame berikut.
>
> ```
> between 5 following and 5 preceding
> between current row and 2 preceding
> between 3 following and current row
> ```

Urutan data yang unik untuk fungsi jendela

Jika klausa ORDER BY untuk fungsi jendela tidak menghasilkan urutan data yang unik dan total, urutan baris tidak deterministik. Jika ekspresi ORDER BY menghasilkan nilai duplikat (urutan parSIAL), urutan pengembalian baris tersebut dapat bervariasi dalam beberapa kali proses. Dalam hal ini, fungsi jendela juga dapat mengembalikan hasil yang tidak terduga atau tidak konsisten.

Misalnya, kueri berikut mengembalikan hasil yang berbeda selama beberapa kali proses. Hasil yang berbeda ini terjadi karena `order by dateid` tidak menghasilkan urutan data yang unik untuk fungsi jendela SUM.

```
select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
--------+-----------+-------------
1827 |   1730.00 |     1730.00
1827 |    708.00 |     2438.00
1827 |    234.00 |     2672.00
...

select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
--------+-----------+-------------
1827 |    234.00 |      234.00
1827 |    472.00 |      706.00
1827 |    347.00 |     1053.00
...
```

Dalam hal ini, menambahkan kolom ORDER BY kedua ke fungsi jendela dapat menyelesaikan masalah.

```
select dateid, pricepaid,
sum(pricepaid) over(order by dateid, pricepaid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;
```

```
dateid | pricepaid | sumpaid
--------+-----------+---------
1827 |    234.00 |  234.00
1827 |    337.00 |  571.00
1827 |    347.00 |  918.00
...
```

Fungsi yang didukung

Amazon Redshift mendukung dua jenis fungsi jendela: agregat dan peringkat.

Berikut ini adalah fungsi agregat yang didukung:

- Fungsi jendela AVG
- Fungsi jendela HITUNG
- Fungsi jendela CUME_DIST
- Fungsi jendela DENSE_RANK
- Fungsi jendela FIRST_VALUE
- Fungsi jendela LAG
- Fungsi jendela LAST_VALUE
- Fungsi jendela LEAD
- Fungsi jendela LISTAGG
- Fungsi jendela MAX
- Fungsi jendela MEDIAN
- Fungsi jendela MIN
- Fungsi jendela NTH_VALUE
- Fungsi jendela PERCENTILE_CONT
- Fungsi jendela PERCENTILE_DISC
- Fungsi jendela RATIO_TO_REPORT
- Fungsi jendela STDDEV_SAMP dan STDDEV_POP(STDDEV_SAMP dan STDDEV adalah sinonim)
- Fungsi jendela SUM

- Fungsi jendela VAR_SAMP dan VAR_POP(VAR_SAMP dan VARIANCE adalah sinonim)

Berikut ini adalah fungsi peringkat yang didukung:

- Fungsi jendela DENSE_RANK
- Fungsi jendela NTILE
- Fungsi jendela PERCENT_RANK
- Fungsi jendela RANK
- Fungsi jendela ROW_NUMBER

Tabel sampel untuk contoh fungsi jendela

Anda dapat menemukan contoh fungsi jendela tertentu dengan setiap deskripsi fungsi. Beberapa contoh menggunakan tabel bernama WINSALES, yang berisi 11 baris, seperti yang ditunjukkan berikut.

ID PENJUALAN	DATEID	SELERID	BUYERS	KUANTITAS	QTY_DIKIRIM
30001	8/2/2003	3	B	10	10
10001	12/24/2003	1	C	10	10
10005	12/24/2003	1	A	30	
40001	1/9/2004	4	A	40	
10006	1/18/2004	1	C	10	
20001	2/12/2004	2	B	20	20
40005	2/12/2004	4	A	10	10
20002	2/16/2004	2	C	20	20
30003	4/18/2004	3	B	15	
30004	4/18/2004	3	B	20	

ID PENJUALAN	DATEID	SELERID	BUYERS	KUANTITAS	QTY_DIKIRIM
30007	9/7/2004	3	C	30	

Skrip berikut membuat dan mengisi tabel sampel WINSALES.

```
CREATE TABLE winsales(
  salesid int,
  dateid date,
  sellerid int,
  buyerid char(10),
  qty int,
  qty_shipped int);

INSERT INTO winsales VALUES
  (30001, '8/2/2003', 3, 'b', 10, 10),
  (10001, '12/24/2003', 1, 'c', 10, 10),
  (10005, '12/24/2003', 1, 'a', 30, null),
  (40001, '1/9/2004', 4, 'a', 40, null),
  (10006, '1/18/2004', 1, 'c', 10, null),
  (20001, '2/12/2004', 2, 'b', 20, 20),
  (40005, '2/12/2004', 4, 'a', 10, 10),
  (20002, '2/16/2004', 2, 'c', 20, 20),
  (30003, '4/18/2004', 3, 'b', 15, null),
  (30004, '4/18/2004', 3, 'b', 20, null),
  (30007, '9/7/2004', 3, 'c', 30, null);
```

Fungsi jendela AVG

Fungsi jendela AVG mengembalikan rata-rata (rata-rata aritmatika) dari nilai ekspresi input. Fungsi AVG bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaksis

```
AVG ( [ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi AVG dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Jenis argumen yang didukung oleh fungsi AVG adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Jenis pengembalian yang didukung oleh fungsi AVG adalah:

- BIGINT untuk argumen SMALINT atau INTEGER
- NUMERIK untuk argumen BIGINT
- PRESISI GANDA untuk argumen float point

Contoh

Contoh berikut menghitung rata-rata bergulir jumlah yang dijual berdasarkan tanggal; urutkan hasil berdasarkan ID tanggal dan ID penjualan:

```
select salesid, dateid, sellerid, qty,
avg(qty) over
(order by dateid, salesid rows unbounded preceding) as avg
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | avg
--------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  10
10005 | 2003-12-24 |        1 |  30 |  16
40001 | 2004-01-09 |        4 |  40 |  22
10006 | 2004-01-18 |        1 |  10 |  20
20001 | 2004-02-12 |        2 |  20 |  20
40005 | 2004-02-12 |        4 |  10 |  18
20002 | 2004-02-16 |        2 |  20 |  18
30003 | 2004-04-18 |        3 |  15 |  18
30004 | 2004-04-18 |        3 |  20 |  18
30007 | 2004-09-07 |        3 |  30 |  19
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Fungsi jendela HITUNG

Fungsi jendela COUNT menghitung baris yang ditentukan oleh ekspresi.

Fungsi COUNT memiliki dua variasi. COUNT (*) menghitung semua baris dalam tabel target apakah mereka menyertakan nol atau tidak. COUNT (ekspresi) menghitung jumlah baris dengan nilai non-null dalam kolom atau ekspresi tertentu.

Sintaksis

```
COUNT ( * | [ ALL ] expression) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
```

```
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi COUNT dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Fungsi COUNT mendukung semua tipe data argumen.

Jenis pengembalian yang didukung oleh fungsi COUNT adalah BIGINT.

Contoh

Contoh berikut menunjukkan ID penjualan, kuantitas, dan jumlah semua baris dari awal jendela data:

```
select salesid, qty,
count(*) over (order by salesid rows unbounded preceding) as count
```

```
from winsales
order by salesid;

salesid | qty | count
---------+-----+-----
10001 |  10 |   1
10005 |  30 |   2
10006 |  10 |   3
20001 |  20 |   4
20002 |  20 |   5
30001 |  10 |   6
30003 |  15 |   7
30004 |  20 |   8
30007 |  30 |   9
40001 |  40 |   10
40005 |  10 |   11
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut menunjukkan bagaimana ID penjualan, kuantitas, dan jumlah baris non-null dari awal jendela data. (Dalam tabel WINSALES, kolom QTY_SHIPPED berisi beberapa NULL.)

```
select salesid, qty, qty_shipped,
count(qty_shipped)
over (order by salesid rows unbounded preceding) as count
from winsales
order by salesid;

salesid | qty | qty_shipped | count
---------+-----+-------------+-------
10001 |  10 |          10 |   1
10005 |  30 |             |   1
10006 |  10 |             |   1
20001 |  20 |          20 |   2
20002 |  20 |          20 |   3
30001 |  10 |          10 |   4
30003 |  15 |             |   4
30004 |  20 |             |   4
30007 |  30 |             |   4
40001 |  40 |             |   4
40005 |  10 |          10 |   5
(11 rows)
```

Fungsi jendela CUME_DIST

Menghitung distribusi kumulatif nilai dalam jendela atau partisi. Dengan asumsi urutan menaik, distribusi kumulatif ditentukan menggunakan rumus ini:

```
count of rows with values <= x / count of rows in the window or partition
```

dimana `x` sama dengan nilai pada baris kolom saat ini yang ditentukan dalam klausa ORDER BY. Dataset berikut menggambarkan penggunaan rumus ini:

```
Row# Value   Calculation    CUME_DIST
1        2500    (1)/(5)    0.2
2        2600    (2)/(5)    0.4
3        2800    (3)/(5)    0.6
4        2900    (4)/(5)    0.8
5        3100    (5)/(5)    1.0
```

Rentang nilai pengembalian adalah> 0 hingga 1, inklusif.

Sintaksis

```
CUME_DIST ()
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

Argumen

DI ATAS

Klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI DENGAN partition_expression

Tidak wajib. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN BERDASARKAN order_list

Ekspresi yang digunakan untuk menghitung distribusi kumulatif. Ekspresi harus memiliki tipe data numerik atau secara implisit dapat dikonversi menjadi satu. Jika ORDER BY dihilangkan, nilai pengembalian adalah 1 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris bersifat nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

FLOAT8

Contoh

Contoh berikut menghitung distribusi kumulatif kuantitas untuk setiap penjual:

```
select sellerid, qty, cume_dist()
over (partition by sellerid order by qty)
from winsales;

sellerid   qty    cume_dist
------------------------------------------------
1          10.00    0.33
1          10.64    0.67
1          30.37    1
3          10.04    0.25
3          15.15    0.5
3          20.75    0.75
3          30.55    1
2          20.09    0.5
2          20.12    1
4          10.12    0.5
4          40.23    1
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Fungsi jendela DENSE_RANK

Fungsi jendela DENSE_RANK menentukan peringkat nilai dalam kelompok nilai, berdasarkan ekspresi ORDER BY dalam klausa OVER. Jika klausa PARTITION BY opsional hadir, peringkat diatur ulang untuk setiap kelompok baris. Baris dengan nilai yang sama untuk kriteria peringkat menerima peringkat yang sama. Fungsi DENSE_RANK berbeda dari RANK dalam satu hal: jika dua atau lebih baris menyatu, tidak ada celah dalam urutan nilai peringkat. Misalnya, jika dua baris diberi peringkat1, peringkat berikutnya adalah2.

Anda dapat memiliki fungsi peringkat dengan klausa PARTITION BY dan ORDER BY yang berbeda dalam kueri yang sama.

Sintaksis

```
DENSE_RANK() OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Klausa jendela untuk fungsi DENSE_RANK.

PARTISI OLEH expr_list

(Opsional) Satu atau lebih ekspresi yang menentukan jendela.

PESANAN BERDASARKAN order_list

(Opsional) Ekspresi yang menjadi dasar nilai peringkat. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel. Jika ORDER BY dihilangkan, nilai pengembalian adalah 1 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris bersifat nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

`BIGINT`

Contoh

Contoh berikut menggunakan tabel sampel untuk fungsi jendela. Untuk informasi selengkapnya, lihat Tabel sampel untuk contoh fungsi jendela.

Contoh berikut mengurutkan tabel berdasarkan jumlah yang dijual dan menetapkan peringkat padat dan peringkat reguler untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
SELECT salesid, qty,
DENSE_RANK() OVER(ORDER BY qty DESC) AS d_rnk,
RANK() OVER(ORDER BY qty DESC) AS rnk
FROM winsales
ORDER BY 2,1;

+---------+-----+-------+-----+
| salesid | qty | d_rnk | rnk |
+---------+-----+-------+-----+
|   10001 |  10 |     5 |   8 |
|   10006 |  10 |     5 |   8 |
|   30001 |  10 |     5 |   8 |
|   40005 |  10 |     5 |   8 |
|   30003 |  15 |     4 |   7 |
|   20001 |  20 |     3 |   4 |
|   20002 |  20 |     3 |   4 |
|   30004 |  20 |     3 |   4 |
|   10005 |  30 |     2 |   2 |
|   30007 |  30 |     2 |   2 |
|   40001 |  40 |     1 |   1 |
+---------+-----+-------+-----+
```

Perhatikan perbedaan peringkat yang ditetapkan ke kumpulan baris yang sama ketika fungsi DENSE_RANK dan RANK digunakan berdampingan dalam kueri yang sama.

Contoh berikut mempartisi tabel dengan sellerid, memesan setiap partisi berdasarkan kuantitas, dan menetapkan peringkat padat untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
SELECT salesid, sellerid, qty,
DENSE_RANK() OVER(PARTITION BY sellerid ORDER BY qty DESC) AS d_rnk
FROM winsales
ORDER BY 2,3,1;

+---------+----------+-----+-------+
| salesid | sellerid | qty | d_rnk |
+---------+----------+-----+-------+
|   10001 |        1 |  10 |     2 |
|   10006 |        1 |  10 |     2 |
|   10005 |        1 |  30 |     1 |
|   20001 |        2 |  20 |     1 |
|   20002 |        2 |  20 |     1 |
```

```
|   30001 |        3 |  10 |     4 |
|   30003 |        3 |  15 |     3 |
|   30004 |        3 |  20 |     2 |
|   30007 |        3 |  30 |     1 |
|   40005 |        4 |  10 |     2 |
|   40001 |        4 |  40 |     1 |
+---------+----------+-----+-------+
```

Untuk berhasil menggunakan contoh terakhir, gunakan perintah berikut untuk menyisipkan baris ke tabel WINSALES. Baris ini memiliki buyerid, sellerid, dan qtysold yang sama dengan baris lainnya. Ini akan menyebabkan dua baris diikat dalam contoh terakhir dan dengan demikian akan menunjukkan perbedaan antara fungsi DENSE_RANK dan RANK.

```
INSERT INTO winsales VALUES(30009, '2/2/2003', 3, 'b', 20, NULL);
```

Contoh berikut mempartisi tabel dengan buyerid dan sellerid, memesan setiap partisi berdasarkan kuantitas, dan menetapkan peringkat padat dan peringkat reguler untuk setiap baris. Hasilnya diurutkan setelah fungsi jendela diterapkan.

```
SELECT salesid, sellerid, qty, buyerid,
DENSE_RANK() OVER(PARTITION BY buyerid, sellerid ORDER BY qty DESC) AS d_rnk,
RANK() OVER (PARTITION BY buyerid, sellerid ORDER BY qty DESC) AS rnk
FROM winsales
ORDER BY rnk;

+---------+----------+-----+---------+-------+-----+
| salesid | sellerid | qty | buyerid | d_rnk | rnk |
+---------+----------+-----+---------+-------+-----+
|   20001 |        2 |  20 | b       |     1 |   1 |
|   30007 |        3 |  30 | c       |     1 |   1 |
|   10006 |        1 |  10 | c       |     1 |   1 |
|   10005 |        1 |  30 | a       |     1 |   1 |
|   20002 |        2 |  20 | c       |     1 |   1 |
|   30009 |        3 |  20 | b       |     1 |   1 |
|   40001 |        4 |  40 | a       |     1 |   1 |
|   30004 |        3 |  20 | b       |     1 |   1 |
|   10001 |        1 |  10 | c       |     1 |   1 |
|   40005 |        4 |  10 | a       |     2 |   2 |
|   30003 |        3 |  15 | b       |     2 |   3 |
|   30001 |        3 |  10 | b       |     3 |   4 |
+---------+----------+-----+---------+-------+-----+
```

Fungsi jendela FIRST_VALUE

Diberikan set baris yang teratur, FIRST_VALUE mengembalikan nilai ekspresi yang ditentukan sehubungan dengan baris pertama dalam bingkai jendela.

Untuk informasi tentang memilih baris terakhir dalam bingkai, lihatFungsi jendela LAST_VALUE.

Sintaksis

```
FIRST_VALUE( expression )[ IGNORE NULLS | RESPECT NULLS ]
OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

ABAIKAN NULLS

Ketika opsi ini digunakan dengan FIRST_VALUE, fungsi mengembalikan nilai pertama dalam bingkai yang bukan NULL (atau NULL jika semua nilai adalah NULL).

RESPECT NULLS

Menunjukkan bahwa Amazon Redshift harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Memperkenalkan klausa jendela untuk fungsi.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada klausa PARTITION BY yang ditentukan, ORDER BY mengurutkan seluruh tabel. Jika Anda menentukan klausa ORDER BY, Anda juga harus menentukan klausa frame_.

Hasil fungsi FIRST_VALUE tergantung pada urutan data. Hasilnya nondeterministik dalam kasus-kasus berikut:

- Ketika tidak ada klausa ORDER BY yang ditentukan dan partisi berisi dua nilai berbeda untuk ekspresi
- Ketika ekspresi mengevaluasi nilai berbeda yang sesuai dengan nilai yang sama dalam daftar ORDER BY.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis pengembalian

Fungsi-fungsi ini mendukung ekspresi yang menggunakan tipe data Amazon Redshift primitif. Jenis pengembalian sama dengan tipe data ekspresi.

Contoh

Contoh berikut menggunakan tabel VENUE dari sampel data TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Contoh berikut mengembalikan kapasitas tempat duduk untuk setiap tempat di tabel VENUE, dengan hasil diurutkan berdasarkan kapasitas (tinggi ke rendah). Fungsi FIRST_VALUE digunakan untuk memilih nama tempat yang sesuai dengan baris pertama dalam bingkai: dalam hal ini, baris dengan jumlah kursi tertinggi. Hasilnya dipartisi berdasarkan negara, jadi ketika nilai VENUESTATE berubah, nilai pertama yang baru dipilih. Bingkai jendela tidak terbatas sehingga nilai pertama yang sama dipilih untuk setiap baris di setiap partisi.

Untuk California, `Qualcomm Stadium` memiliki jumlah kursi tertinggi (`70561`), jadi nama ini adalah nilai pertama untuk semua baris di `CA` partisi.

```
select venuestate, venueseats, venuename,
first_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
```

```
order by venuestate;

venuestate | venueseats |           venuename            |         first_value
-----------+------------+--------------------------------
+-----------------------------
CA         |      70561 | Qualcomm Stadium               | Qualcomm Stadium
CA         |      69843 | Monster Park                   | Qualcomm Stadium
CA         |      63026 | McAfee Coliseum                | Qualcomm Stadium
CA         |      56000 | Dodger Stadium                 | Qualcomm Stadium
CA         |      45050 | Angel Stadium of Anaheim       | Qualcomm Stadium
CA         |      42445 | PETCO Park                     | Qualcomm Stadium
CA         |      41503 | AT&T Park                      | Qualcomm Stadium
CA         |      22000 | Shoreline Amphitheatre         | Qualcomm Stadium
CO         |      76125 | INVESCO Field                  | INVESCO Field
CO         |      50445 | Coors Field                    | INVESCO Field
DC         |      41888 | Nationals Park                 | Nationals Park
FL         |      74916 | Dolphin Stadium                | Dolphin Stadium
FL         |      73800 | Jacksonville Municipal Stadium | Dolphin Stadium
FL         |      65647 | Raymond James Stadium          | Dolphin Stadium
FL         |      36048 | Tropicana Field                | Dolphin Stadium
...
```

Contoh berikut menunjukkan penggunaan opsi IGNORE NULLS dan bergantung pada penambahan baris baru ke tabel VENUE:

```
insert into venue values(2000,null,'Stanford','CA',90000);
```

Baris baru ini berisi nilai NULL untuk kolom VENUENAME. Sekarang ulangi kueri FIRST_VALUE yang ditampilkan sebelumnya di bagian ini:

```
select venuestate, venueseats, venuename,
first_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |         venuename          | first_value
-----------+------------+----------------------------+-------------
CA         |      90000 | NULL                       | NULL
CA         |      70561 | Qualcomm Stadium           | NULL
```

```
CA         |      69843 | Monster Park               | NULL
...
```

Karena baris baru berisi nilai VENUESEATS tertinggi (90000) dan VENUENAME-nya adalah NULL, fungsi FIRST_VALUE mengembalikan NULL untuk partisi. CA Untuk mengabaikan baris seperti ini dalam evaluasi fungsi, tambahkan opsi IGNORE NULLS ke argumen fungsi:

```
select venuestate, venueseats, venuename,
first_value(venuename) ignore nulls
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venuestate='CA')
order by venuestate;

venuestate | venueseats |         venuename          |   first_value
------------+------------+----------------------------+------------------
CA         |      90000 | NULL                       | Qualcomm Stadium
CA         |      70561 | Qualcomm Stadium           | Qualcomm Stadium
CA         |      69843 | Monster Park               | Qualcomm Stadium
...
```

Fungsi jendela LAG

Fungsi jendela LAG mengembalikan nilai untuk baris pada offset tertentu di atas (sebelum) baris saat ini di partisi.

Sintaksis

```
LAG (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

Argumen

nilai_expr

Kolom target atau ekspresi tempat fungsi beroperasi.

mengimbangi

Parameter opsional yang menentukan jumlah baris sebelum baris saat ini untuk mengembalikan nilainya. Offset dapat berupa bilangan bulat konstan atau ekspresi yang mengevaluasi ke

bilangan bulat. Jika Anda tidak menentukan offset, Amazon Redshift menggunakan 1 nilai default. Offset `0` menunjukkan baris saat ini.

ABAIKAN NULLS

Spesifikasi opsional yang menunjukkan bahwa Amazon Redshift harus melewati nilai nol dalam penentuan baris mana yang akan digunakan. Nilai null disertakan jika IGNORE NULLS tidak terdaftar.

> **Note**
>
> Anda dapat menggunakan ekspresi NVL atau COALESCE untuk mengganti nilai null dengan nilai lain. Untuk informasi selengkapnya, lihat Fungsi NVL dan COALESCE.

RESPECT NULLS

Menunjukkan bahwa Amazon Redshift harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Menentukan partisi jendela dan pemesanan. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH window_partition

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN BERDASARKAN window_

Mengurutkan baris dalam setiap partisi.

Fungsi jendela LAG mendukung ekspresi yang menggunakan salah satu jenis data Amazon Redshift. Jenis pengembalian sama dengan tipe value_ expr.

Contoh

Contoh berikut menunjukkan jumlah tiket yang dijual kepada pembeli dengan ID pembeli 3 dan waktu pembeli 3 membeli tiket. Untuk membandingkan setiap penjualan dengan penjualan sebelumnya

untuk pembeli 3, kueri mengembalikan jumlah sebelumnya yang terjual untuk setiap penjualan. Karena tidak ada pembelian sebelum 1/16 /2008, nilai jual kuantitas pertama sebelumnya adalah null:

```
select buyerid, saletime, qtysold,
lag(qtysold,1) over (order by buyerid, saletime) as prev_qtysold
from sales where buyerid = 3 order by buyerid, saletime;

buyerid |      saletime       | qtysold | prev_qtysold
---------+---------------------+---------+--------------
3 | 2008-01-16 01:06:09 |       1 |
3 | 2008-01-28 02:10:01 |       1 |            1
3 | 2008-03-12 10:39:53 |       1 |            1
3 | 2008-03-13 02:56:07 |       1 |            1
3 | 2008-03-29 08:21:39 |       2 |            1
3 | 2008-04-27 02:39:01 |       1 |            2
3 | 2008-08-16 07:04:37 |       2 |            1
3 | 2008-08-22 11:45:26 |       2 |            2
3 | 2008-09-12 09:11:25 |       1 |            2
3 | 2008-10-01 06:22:37 |       1 |            1
3 | 2008-10-20 01:55:51 |       2 |            1
3 | 2008-10-28 01:30:40 |       1 |            2
(12 rows)
```

Fungsi jendela LAST_VALUE

Diberikan set baris yang teratur, Fungsi LAST_VALUE mengembalikan nilai ekspresi sehubungan dengan baris terakhir dalam bingkai.

Untuk informasi tentang memilih baris pertama dalam bingkai, lihatFungsi jendela FIRST_VALUE.

Sintaksis

```
LAST_VALUE( expression )[ IGNORE NULLS | RESPECT NULLS ]
OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

ABAIKAN NULLS

Fungsi mengembalikan nilai terakhir dalam frame yang bukan NULL (atau NULL jika semua nilai adalah NULL).

RESPECT NULLS

Menunjukkan bahwa Amazon Redshift harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Memperkenalkan klausa jendela untuk fungsi.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada klausa PARTITION BY yang ditentukan, ORDER BY mengurutkan seluruh tabel. Jika Anda menentukan klausa ORDER BY, Anda juga harus menentukan klausa frame_.

Hasilnya tergantung pada urutan data. Hasilnya nondeterministik dalam kasus-kasus berikut:

- Ketika tidak ada klausa ORDER BY yang ditentukan dan partisi berisi dua nilai berbeda untuk ekspresi
- Ketika ekspresi mengevaluasi nilai berbeda yang sesuai dengan nilai yang sama dalam daftar ORDER BY.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis pengembalian

Fungsi-fungsi ini mendukung ekspresi yang menggunakan tipe data Amazon Redshift primitif. Jenis pengembalian sama dengan tipe data ekspresi.

Contoh

Contoh berikut menggunakan tabel VENUE dari sampel data TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

Contoh berikut mengembalikan kapasitas tempat duduk untuk setiap tempat di tabel VENUE, dengan hasil diurutkan berdasarkan kapasitas (tinggi ke rendah). Fungsi LAST_VALUE digunakan untuk memilih nama tempat yang sesuai dengan baris terakhir dalam bingkai: dalam hal ini, baris dengan jumlah kursi paling sedikit. Hasilnya dipartisi berdasarkan negara, jadi ketika nilai VENUESTATE berubah, nilai terakhir baru dipilih. Bingkai jendela tidak terbatas sehingga nilai terakhir yang sama dipilih untuk setiap baris di setiap partisi.

Untuk California, `Shoreline Amphitheatre` dikembalikan untuk setiap baris di partisi karena memiliki jumlah kursi terendah (`22000`).

```
select venuestate, venueseats, venuename,
last_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |           venuename            |          last_value
-----------+------------+--------------------------------
+----------------------------
CA         |      70561 | Qualcomm Stadium               | Shoreline Amphitheatre
CA         |      69843 | Monster Park                   | Shoreline Amphitheatre
CA         |      63026 | McAfee Coliseum                | Shoreline Amphitheatre
CA         |      56000 | Dodger Stadium                 | Shoreline Amphitheatre
CA         |      45050 | Angel Stadium of Anaheim       | Shoreline Amphitheatre
CA         |      42445 | PETCO Park                     | Shoreline Amphitheatre
CA         |      41503 | AT&T Park                      | Shoreline Amphitheatre
CA         |      22000 | Shoreline Amphitheatre         | Shoreline Amphitheatre
CO         |      76125 | INVESCO Field                  | Coors Field
CO         |      50445 | Coors Field                    | Coors Field
DC         |      41888 | Nationals Park                 | Nationals Park
FL         |      74916 | Dolphin Stadium                | Tropicana Field
FL         |      73800 | Jacksonville Municipal Stadium | Tropicana Field
FL         |      65647 | Raymond James Stadium          | Tropicana Field
FL         |      36048 | Tropicana Field                | Tropicana Field
...
```

Fungsi jendela LEAD

Fungsi jendela LEAD mengembalikan nilai untuk baris pada offset tertentu di bawah (setelah) baris saat ini di partisi.

Sintaksis

```
LEAD (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

Argumen

nilai_expr

Kolom target atau ekspresi tempat fungsi beroperasi.

mengimbangi

Parameter opsional yang menentukan jumlah baris di bawah baris saat ini untuk mengembalikan nilainya. Offset dapat berupa bilangan bulat konstan atau ekspresi yang mengevaluasi ke bilangan bulat. Jika Anda tidak menentukan offset, Amazon Redshift menggunakan 1 nilai default. Offset `0` menunjukkan baris saat ini.

ABAIKAN NULLS

Spesifikasi opsional yang menunjukkan bahwa Amazon Redshift harus melewati nilai nol dalam penentuan baris mana yang akan digunakan. Nilai null disertakan jika IGNORE NULLS tidak terdaftar.

> **Note**
>
> Anda dapat menggunakan ekspresi NVL atau COALESCE untuk mengganti nilai null dengan nilai lain. Untuk informasi selengkapnya, lihat Fungsi NVL dan COALESCE.

RESPECT NULLS

Menunjukkan bahwa Amazon Redshift harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Menentukan partisi jendela dan pemesanan. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH window_partition

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN BERDASARKAN window_

Mengurutkan baris dalam setiap partisi.

Fungsi jendela LEAD mendukung ekspresi yang menggunakan salah satu jenis data Amazon Redshift. Jenis pengembalian sama dengan tipe value_ expr.

Contoh

Contoh berikut memberikan komisi untuk acara di tabel PENJUALAN yang tiketnya dijual pada 1 Januari 2008 dan 2 Januari 2008 dan komisi yang dibayarkan untuk penjualan tiket untuk penjualan berikutnya. Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

```
SELECT eventid, commission, saletime, LEAD(commission, 1) over ( ORDER BY saletime ) AS
 next_comm
FROM sales
WHERE saletime BETWEEN '2008-01-09 00:00:00' AND '2008-01-10 12:59:59'
LIMIT 10;

+---------+------------+---------------------+-----------+
| eventid | commission |      saletime       | next_comm |
+---------+------------+---------------------+-----------+
|    1664 |       13.2 | 2008-01-09 01:00:21 |      69.6 |
|     184 |       69.6 | 2008-01-09 01:00:36 |     116.1 |
|    6870 |      116.1 | 2008-01-09 01:02:37 |      11.1 |
|    3718 |       11.1 | 2008-01-09 01:05:19 |     205.5 |
|    6772 |      205.5 | 2008-01-09 01:14:04 |      38.4 |
|    3074 |       38.4 | 2008-01-09 01:26:50 |     209.4 |
|    5254 |      209.4 | 2008-01-09 01:29:16 |      26.4 |
|    3724 |       26.4 | 2008-01-09 01:40:09 |      57.6 |
|    5303 |       57.6 | 2008-01-09 01:40:21 |      51.6 |
|    3678 |       51.6 | 2008-01-09 01:42:54 |      43.8 |
+---------+------------+---------------------+-----------+
```

Contoh berikut memberikan perbedaan maksimum untuk komisi untuk acara di tabel PENJUALAN untuk dan komisi yang dibayarkan untuk penjualan tiket untuk penjualan berikutnya untuk acara yang sama. Contoh ini menunjukkan cara menggunakan LEAD dengan klausa GROUP BY. Karena fungsi jendela tidak diperbolehkan dalam klausa agregat, contoh ini menggunakan subquery. Contoh berikut menggunakan database sampel TICKIT. Untuk informasi selengkapnya, lihat Database sampel.

```
SELECT eventid, eventname, max(next_comm_diff) as max_commission_difference
FROM
(
    SELECT sales.eventid, eventname, commission - LEAD(commission, 1) over (ORDER BY
 sales.eventid, saletime) AS next_comm_diff
    FROM sales JOIN event ON sales.eventid = event.eventid
)
GROUP BY eventid, eventname
ORDER BY eventid

LIMIT 10

| eventid | eventname                   | max_commission_difference |
+---------+-----------------------------+---------------------------+
| 1       | Gotterdammerung             | 7.95                      |
| 2       | Boris Godunov               | 227.85                    |
| 3       | Salome                      | 1350.9                    |
| 4       | La Cenerentola (Cinderella) | 790.05                    |
| 5       | Il Trovatore                | 214.05                    |
| 6       | L Elisir d Amore            | 510.9                     |
| 7       | Doctor Atomic               | 180.6                     |
| 9       | The Fly                     | 147                       |
| 10      | Rigoletto                   | 186.6                     |
+---------+-----------------------------+---------------------------+
```

Fungsi jendela LISTAGG

Untuk setiap grup dalam kueri, fungsi jendela LISTAGG mengurutkan baris untuk grup tersebut sesuai dengan ekspresi ORDER BY, kemudian menggabungkan nilai-nilai menjadi satu string.

Sintaksis

```
LISTAGG( [DISTINCT] expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]
OVER ( [PARTITION BY partition_expression] )
```

Argumen

DISTINCT

(Opsional) Klausa yang menghilangkan nilai duplikat dari ekspresi yang ditentukan sebelum menggabungkan. Spasi akhir diabaikan, sehingga string `'a'` dan `'a '` diperlakukan sebagai duplikat. LISTAGG menggunakan nilai pertama yang ditemui. Untuk informasi selengkapnya, lihat Signifikansi blanko yang tertinggal.

ekspresi agregat

Setiap ekspresi valid (seperti nama kolom) yang menyediakan nilai untuk digabungkan. Nilai NULL dan string kosong diabaikan.

pembatas

(Opsional) Konstanta string untuk akan memisahkan nilai gabungan. Default-nya adalah NULL.

DALAM GRUP (ORDER BY order_list)

(Opsional) Klausa yang menentukan urutan urutan nilai agregat. Deterministik hanya jika ORDER BY memberikan urutan yang unik. Defaultnya adalah untuk menggabungkan semua baris dan mengembalikan nilai tunggal.

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI DENGAN partition_expression

(Opsional) Menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Pengembalian

Jika set hasil lebih besar dari 16.000.000 byte, maka LISTAGG mengembalikan kesalahan berikut:

```
Invalid operation: Result size exceeds LISTAGG limit
```

Contoh

Contoh berikut menggunakan tabel WINSALES. Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut mengembalikan daftar ID penjual, diurutkan berdasarkan ID penjual.

```
select listagg(sellerid)
within group (order by sellerid)
over() from winsales;

  listagg
------------
 11122333344
...
...
 11122333344
 11122333344
   (11 rows)
```

Contoh berikut mengembalikan daftar ID penjual untuk pembeli B, diurutkan berdasarkan tanggal.

```
select listagg(sellerid)
within group (order by dateid)
over () as seller
from winsales
where buyerid = 'b' ;

  seller
---------
    3233
    3233
    3233
    3233
```

Contoh berikut mengembalikan daftar tanggal penjualan yang dipisahkan koma untuk pembeli B.

```
select listagg(dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid  = 'b';

            dates
----------------------------------------
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
```

```
2003-08-02,2004-04-18,2004-04-18,2004-02-12
```

Contoh berikut menggunakan DISTINCT untuk mengembalikan daftar tanggal penjualan unik untuk pembeli B.

```
select listagg(distinct dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid  = 'b';

           dates
-------------------------------
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
```

Contoh berikut mengembalikan daftar ID penjualan yang dipisahkan koma untuk setiap ID pembeli.

```
select buyerid,
listagg(salesid,',')
within group (order by salesid)
over (partition by buyerid) as sales_id
from winsales
order by buyerid;

+---------+------------------------+
| buyerid |        sales_id        |
+---------+------------------------+
| a       | 10005,40001,40005      |
| a       | 10005,40001,40005      |
| a       | 10005,40001,40005      |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
+---------+------------------------+
```

Contoh berikut menunjukkan dukungan LISTAGG dengan hasil gabungan hingga 16.000.000 byte:

```
CREATE TABLE large_data (
    id INT,
    content VARCHAR(65535)
);

INSERT INTO large_data VALUES
    (1, REPEAT('A', 65535)),
    (2, REPEAT('B', 65535)),
    (3, REPEAT('C', 65535));

SELECT LEN(LISTAGG(content, ',') WITHIN GROUP (ORDER BY id)) AS total_length
FROM large_data;

 total_length
--------------
       196607
```

Fungsi jendela MAX

Fungsi jendela MAX mengembalikan nilai ekspresi masukan maksimum. Fungsi MAX bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaksis

```
MAX ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Klausa yang menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi MAX dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Menerima jenis data apa pun sebagai input. Mengembalikan tipe data yang sama dengan ekspresi.

Contoh

Contoh berikut menunjukkan ID penjualan, kuantitas, dan kuantitas maksimum dari awal jendela data:

```
select salesid, qty,
max(qty) over (order by salesid rows unbounded preceding) as max
from winsales
order by salesid;

salesid | qty | max
---------+-----+-----
10001 |  10 |  10
10005 |  30 |  30
10006 |  10 |  30
20001 |  20 |  30
20002 |  20 |  30
```

```
30001 |  10 |  30
30003 |  15 |  30
30004 |  20 |  30
30007 |  30 |  30
40001 |  40 |  40
40005 |  10 |  40
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut menunjukkan salesid, kuantitas, dan kuantitas maksimum dalam bingkai terbatas:

```
select salesid, qty,
max(qty) over (order by salesid rows between 2 preceding and 1 preceding) as max
from winsales
order by salesid;

salesid | qty | max
---------+-----+-----
10001 |  10 |
10005 |  30 |  10
10006 |  10 |  30
20001 |  20 |  30
20002 |  20 |  20
30001 |  10 |  20
30003 |  15 |  20
30004 |  20 |  15
30007 |  30 |  20
40001 |  40 |  30
40005 |  10 |  40
(11 rows)
```

Fungsi jendela MEDIAN

Menghitung nilai median untuk rentang nilai di jendela atau partisi. Nilai NULL dalam rentang diabaikan.

MEDIAN adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu.

Sintaksis

```
MEDIAN ( median_expression )
```

```
OVER ( [ PARTITION BY partition_expression ] )
```

Argumen

median_ekspresi

Ekspresi, seperti nama kolom, yang menyediakan nilai untuk menentukan median. Ekspresi harus memiliki tipe data numerik atau datetime atau secara implisit dapat dikonversi menjadi satu.

DI ATAS

Klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI DENGAN partition_expression

Tidak wajib. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Jenis Data

Jenis pengembalian ditentukan oleh tipe data median_expression. Tabel berikut menunjukkan tipe pengembalian untuk setiap tipe data median_expression.

Tipe input	Tipe Pengembalian
INT2, INT4, INT8, NUMERIK, DESIMAL	DECIMAL
MENGAPUNG, GANDA	DOUBLE
DATE	DATE

Catatan penggunaan

Jika argumen median_expression adalah tipe data DECIMAL yang ditentukan dengan presisi maksimum 38 digit, ada kemungkinan MEDIAN akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi MEDIAN melebihi 38 digit, hasilnya dipotong agar pas, yang menyebabkan hilangnya presisi. Jika, selama interpolasi, hasil perantara melebihi presisi maksimum, terjadi luapan numerik dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi yang lebih rendah atau melemparkan argumen median_expression ke presisi yang lebih rendah.

Misalnya, fungsi SUM dengan argumen DECIMAL mengembalikan presisi default 38 digit. Skala hasilnya sama dengan skala argumen. Jadi, misalnya, SUM kolom DECIMAL (5,2) mengembalikan tipe data DECIMAL (38,2).

Contoh berikut menggunakan fungsi SUM dalam argumen median_expression dari fungsi MEDIAN. Tipe data kolom PRICEPAID adalah DECIMAL (8,2), sehingga fungsi SUM mengembalikan DECIMAL (38,2).

```
select salesid, sum(pricepaid), median(sum(pricepaid))
over() from sales where salesid < 10 group by salesid;
```

Untuk menghindari potensi kehilangan presisi atau kesalahan luapan, berikan hasilnya ke tipe data DECIMAL dengan presisi yang lebih rendah, seperti yang ditunjukkan contoh berikut.

```
select salesid, sum(pricepaid), median(sum(pricepaid)::decimal(30,2))
over() from sales where salesid < 10 group by salesid;
```

Contoh

Contoh berikut menghitung jumlah penjualan rata-rata untuk setiap penjual:

```
select sellerid, qty, median(qty)
over (partition by sellerid)
from winsales
order by sellerid;


sellerid qty median
---------------------------
1  10 10.0
1  10 10.0
1  30 10.0
2  20 20.0
2  20 20.0
3  10 17.5
3  15 17.5
3  20 17.5
3  30 17.5
4  10 25.0
4  40 25.0
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Fungsi jendela MIN

Fungsi jendela MIN mengembalikan minimum nilai ekspresi masukan. Fungsi MIN bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaksis

```
MIN ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi MIN dalam hal satu atau lebih ekspresi.

PESAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Menerima jenis data apa pun sebagai input. Mengembalikan tipe data yang sama dengan ekspresi.

Contoh

Contoh berikut menunjukkan ID penjualan, kuantitas, dan jumlah minimum dari awal jendela data:

```
select salesid, qty,
min(qty) over
(order by salesid rows unbounded preceding)
from winsales
order by salesid;

salesid | qty | min
---------+-----+-----
10001 |  10 |  10
10005 |  30 |  10
10006 |  10 |  10
20001 |  20 |  10
20002 |  20 |  10
30001 |  10 |  10
30003 |  15 |  10
30004 |  20 |  10
30007 |  30 |  10
40001 |  40 |  10
40005 |  10 |  10
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut menunjukkan ID penjualan, kuantitas, dan kuantitas minimum dalam bingkai terbatas:

```
select salesid, qty,
min(qty) over
(order by salesid rows between 2 preceding and 1 preceding) as min
from winsales
order by salesid;

salesid | qty | min
---------+-----+-----
10001 |  10 |
10005 |  30 |  10
```

```
10006 |  10 |  10
20001 |  20 |  10
20002 |  20 |  10
30001 |  10 |  20
30003 |  15 |  10
30004 |  20 |  10
30007 |  30 |  15
40001 |  40 |  20
40005 |  10 |  30
(11 rows)
```

Fungsi jendela NTH_VALUE

Fungsi jendela NTH_VALUE mengembalikan nilai ekspresi dari baris yang ditentukan dari bingkai jendela relatif terhadap baris pertama jendela.

Sintaksis

```
NTH_VALUE (expr, offset)
[ IGNORE NULLS | RESPECT NULLS ]
OVER
( [ PARTITION BY window_partition ]
[ ORDER BY window_ordering
                        frame_clause ] )
```

Argumen

expres

Kolom target atau ekspresi tempat fungsi beroperasi.

mengimbangi

Menentukan nomor baris relatif terhadap baris pertama di jendela untuk mengembalikan ekspresi. Offset dapat berupa konstanta atau ekspresi dan harus berupa bilangan bulat positif yang lebih besar dari 0.

ABAIKAN NULLS

Spesifikasi opsional yang menunjukkan bahwa Amazon Redshift harus melewati nilai nol dalam penentuan baris mana yang akan digunakan. Nilai null disertakan jika IGNORE NULLS tidak terdaftar.

RESPECT NULLS

Menunjukkan bahwa Amazon Redshift harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Menentukan partisi jendela, pemesanan, dan bingkai jendela.

PARTISI OLEH window_partition

Menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESAN BERDASARKAN window_

Mengurutkan baris dalam setiap partisi. Jika ORDER BY dihilangkan, bingkai default terdiri dari semua baris di partisi.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Fungsi jendela NTH_VALUE mendukung ekspresi yang menggunakan salah satu jenis data Amazon Redshift. Jenis pengembalian sama dengan tipe expr.

Contoh

Contoh berikut menunjukkan jumlah kursi di tempat terbesar ketiga di California, Florida, dan New York dibandingkan dengan jumlah kursi di tempat-tempat lain di negara bagian tersebut:

```
select venuestate, venuename, venueseats,
nth_value(venueseats, 3)
ignore nulls
over(partition by venuestate order by venueseats desc
rows between unbounded preceding and unbounded following)
as third_most_seats
from (select * from venue where venueseats > 0 and
venuestate in('CA', 'FL', 'NY'))
order by venuestate;

venuestate |           venuename            | venueseats | third_most_seats
```

```
------------+--------------------------------+------------+------------------
CA          | Qualcomm Stadium               |      70561 |            63026
CA          | Monster Park                   |      69843 |            63026
CA          | McAfee Coliseum                |      63026 |            63026
CA          | Dodger Stadium                 |      56000 |            63026
CA          | Angel Stadium of Anaheim       |      45050 |            63026
CA          | PETCO Park                     |      42445 |            63026
CA          | AT&T Park                      |      41503 |            63026
CA          | Shoreline Amphitheatre         |      22000 |            63026
FL          | Dolphin Stadium                |      74916 |            65647
FL          | Jacksonville Municipal Stadium |      73800 |            65647
FL          | Raymond James Stadium          |      65647 |            65647
FL          | Tropicana Field                |      36048 |            65647
NY          | Ralph Wilson Stadium           |      73967 |            20000
NY          | Yankee Stadium                 |      52325 |            20000
NY          | Madison Square Garden          |      20000 |            20000
(15 rows)
```

Fungsi jendela NTILE

Fungsi jendela NTILE membagi baris yang diurutkan dalam partisi ke dalam jumlah grup peringkat yang ditentukan dengan ukuran yang sama mungkin dan mengembalikan grup tempat baris tertentu jatuh.

Sintaksis

```
NTILE (expr)
OVER (
[ PARTITION BY expression_list ]
[ ORDER BY order_list ]
)
```

Argumen

expres

Jumlah grup peringkat dan harus menghasilkan nilai integer positif (lebih besar dari 0) untuk setiap partisi. Argum en expr tidak boleh dibatalkan.

DI ATAS

Klausa yang menentukan partisi jendela dan pemesanan. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH window_partition

Tidak wajib. Kisaran catatan untuk setiap grup dalam klausa OVER.

PESAN BERDASARKAN window_

Tidak wajib. Ekspresi yang mengurutkan baris dalam setiap partisi. Jika klausa ORDER BY dihilangkan, perilaku peringkatnya sama.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris bersifat nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

BIGINT

Contoh

Contoh berikut mengurutkan ke dalam empat kelompok peringkat harga yang dibayarkan untuk tiket Hamlet pada 26 Agustus 2008. Kumpulan hasil adalah 17 baris, dibagi hampir merata di antara peringkat 1 hingga 4:

```
select eventname, caldate, pricepaid, ntile(4)
over(order by pricepaid desc) from sales, event, date
where sales.eventid=event.eventid and event.dateid=date.dateid and eventname='Hamlet'
and caldate='2008-08-26'
order by 4;

eventname |  caldate   | pricepaid | ntile
----------+------------+-----------+-------
Hamlet    | 2008-08-26 |   1883.00 |     1
Hamlet    | 2008-08-26 |   1065.00 |     1
Hamlet    | 2008-08-26 |    589.00 |     1
Hamlet    | 2008-08-26 |    530.00 |     1
Hamlet    | 2008-08-26 |    472.00 |     1
Hamlet    | 2008-08-26 |    460.00 |     2
Hamlet    | 2008-08-26 |    355.00 |     2
Hamlet    | 2008-08-26 |    334.00 |     2
Hamlet    | 2008-08-26 |    296.00 |     2
Hamlet    | 2008-08-26 |    230.00 |     3
Hamlet    | 2008-08-26 |    216.00 |     3
Hamlet    | 2008-08-26 |    212.00 |     3
```

```
Hamlet    | 2008-08-26 |    106.00 |     3
Hamlet    | 2008-08-26 |    100.00 |     4
Hamlet    | 2008-08-26 |     94.00 |     4
Hamlet    | 2008-08-26 |     53.00 |     4
Hamlet    | 2008-08-26 |     25.00 |     4
(17 rows)
```

Fungsi jendela PERCENT_RANK

Menghitung peringkat persen dari baris tertentu. Peringkat persen ditentukan menggunakan rumus ini:

```
(x - 1) / (the number of rows in the window or partition - 1)
```

dimana `x` adalah peringkat baris saat ini. Dataset berikut menggambarkan penggunaan rumus ini:

```
Row# Value Rank Calculation PERCENT_RANK
1 15 1 (1-1)/(7-1) 0.0000
2 20 2 (2-1)/(7-1) 0.1666
3 20 2 (2-1)/(7-1) 0.1666
4 20 2 (2-1)/(7-1) 0.1666
5 30 5 (5-1)/(7-1) 0.6666
6 30 5 (5-1)/(7-1) 0.6666
7 40 7 (7-1)/(7-1) 1.0000
```

Rentang nilai pengembalian adalah 0 hingga 1, inklusif. Baris pertama dalam set apa pun memiliki PERCENT_RANK 0.

Sintaksis

```
PERCENT_RANK ()
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI DENGAN partition_expression

Tidak wajib. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN BERDASARKAN order_list

Tidak wajib. Ekspresi untuk menghitung peringkat persen. Ekspresi harus memiliki tipe data numerik atau secara implisit dapat dikonversi menjadi satu. Jika ORDER BY dihilangkan, nilai pengembalian adalah 0 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris bersifat nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

FLOAT8

Contoh

Contoh berikut menghitung peringkat persen dari jumlah penjualan untuk setiap penjual:

```
select sellerid, qty, percent_rank()
over (partition by sellerid order by qty)
from winsales;

sellerid qty  percent_rank
----------------------------------------
1  10.00  0.0
1  10.64  0.5
1  30.37  1.0
3  10.04  0.0
3  15.15  0.33
3  20.75  0.67
3  30.55  1.0
2  20.09  0.0
2  20.12  1.0
4  10.12  0.0
```

```
4  40.23  1.0
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Fungsi jendela PERCENTILE_CONT

PERCENTILE_CONT adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu. Dibutuhkan nilai persentil dan spesifikasi pengurutan, dan mengembalikan nilai interpolasi yang akan jatuh ke dalam nilai persentil yang diberikan sehubungan dengan spesifikasi pengurutan.

PERCENTILE_CONT menghitung interpolasi linier antara nilai-nilai setelah mengurutkannya. Menggunakan nilai persentil `(P)` dan jumlah baris bukan nol `(N)` dalam grup agregasi, fungsi menghitung nomor baris setelah mengurutkan baris sesuai dengan spesifikasi pengurutan. Nomor baris ini `(RN)` dihitung sesuai dengan rumus`RN = (1+ (P*(N-1))`. Hasil akhir dari fungsi agregat dihitung dengan interpolasi linier antara nilai-nilai dari baris pada nomor `CRN = CEILING(RN)` baris dan. `FRN = FLOOR(RN)`

Hasil akhirnya adalah sebagai berikut.

`(CRN = FRN = RN)`Jika hasilnya adalah `(value of expression from row at RN)`

Jika tidak, hasilnya adalah sebagai berikut:

`(CRN - RN) * (value of expression for row at FRN) + (RN - FRN) * (value of expression for row at CRN)`.

Anda hanya dapat menentukan klausa PARTITION dalam klausa OVER. Jika PARTITION ditentukan, untuk setiap baris, PERCENTILE_CONT mengembalikan nilai yang akan jatuh ke dalam persentil yang ditentukan di antara satu set nilai dalam partisi tertentu.

Sintaksis

```
PERCENTILE_CONT ( percentile )
WITHIN GROUP (ORDER BY expr)
OVER (  [ PARTITION BY expr_list ]  )
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Menentukan numerik atau date/time nilai untuk mengurutkan dan menghitung persentil.

DI ATAS

Menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH expr

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Pengembalian

Jenis pengembalian ditentukan oleh tipe data ekspresi ORDER BY dalam klausa WITHIN GROUP. Tabel berikut menunjukkan tipe pengembalian untuk setiap tipe data ekspresi ORDER BY.

Tipe input	Tipe Pengembalian
INT2, INT4, INT8, NUMERIK, DESIMAL	DECIMAL
MENGAPUNG, GANDA	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP

Catatan penggunaan

Jika ekspresi ORDER BY adalah tipe data DECIMAL yang ditentukan dengan presisi maksimum 38 digit, ada kemungkinan bahwa PERCENTILE_CONT akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi PERCENTILE_CONT melebihi 38 digit, hasilnya dipotong agar pas, yang menyebabkan hilangnya presisi. Jika, selama interpolasi, hasil perantara melebihi presisi maksimum, terjadi luapan numerik dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi yang lebih rendah atau merapikan ekspresi ORDER BY ke presisi yang lebih rendah.

Misalnya, fungsi SUM dengan argumen DECIMAL mengembalikan presisi default 38 digit. Skala hasilnya sama dengan skala argumen. Jadi, misalnya, SUM kolom DECIMAL (5,2) mengembalikan tipe data DECIMAL (38,2).

Contoh berikut menggunakan fungsi SUM dalam klausa ORDER BY dari fungsi PERCENTILE_CONT. Tipe data kolom PRICEPAID adalah DECIMAL (8,2), sehingga fungsi SUM mengembalikan DECIMAL (38,2).

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid) desc) over()
from sales where salesid < 10 group by salesid;
```

Untuk menghindari potensi kehilangan presisi atau kesalahan luapan, berikan hasilnya ke tipe data DECIMAL dengan presisi yang lebih rendah, seperti yang ditunjukkan contoh berikut.

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid)::decimal(30,2) desc) over()
from sales where salesid < 10 group by salesid;
```

Contoh

Contoh berikut menggunakan tabel WINSALES. Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
over() as median from winsales;

 sellerid | qty | median
----------+-----+--------
        1 |  10 |   20.0
        1 |  10 |   20.0
        3 |  10 |   20.0
        4 |  10 |   20.0
        3 |  15 |   20.0
        2 |  20 |   20.0
        3 |  20 |   20.0
        2 |  20 |   20.0
        3 |  30 |   20.0
        1 |  30 |   20.0
        4 |  40 |   20.0
(11 rows)
```

```
select sellerid, qty, percentile_cont(0.5)
```

```
within group (order by qty)
over(partition by sellerid) as median from winsales;

 sellerid | qty | median
----------+-----+--------
        2 |  20 |   20.0
        2 |  20 |   20.0
        4 |  10 |   25.0
        4 |  40 |   25.0
        1 |  10 |   10.0
        1 |  10 |   10.0
        1 |  30 |   10.0
        3 |  10 |   17.5
        3 |  15 |   17.5
        3 |  20 |   17.5
        3 |  30 |   17.5
(11 rows)
```

Contoh berikut menghitung PERCENTILE_CONT dan PERCENTILE_DISC dari penjualan tiket untuk penjual di negara bagian Washington.

```
SELECT sellerid, state, sum(qtysold*pricepaid) sales,
percentile_cont(0.6) within group (order by sum(qtysold*pricepaid::decimal(14,2) )
 desc) over(),
percentile_disc(0.6) within group (order by sum(qtysold*pricepaid::decimal(14,2) )
 desc) over()
from sales s, users u
where s.sellerid = u.userid and state = 'WA' and sellerid < 1000
group by sellerid, state;

 sellerid | state |  sales  | percentile_cont | percentile_disc
----------+-------+---------+-----------------+-----------------
      127 | WA    | 6076.00 |         2044.20 |         1531.00
      787 | WA    | 6035.00 |         2044.20 |         1531.00
      381 | WA    | 5881.00 |         2044.20 |         1531.00
      777 | WA    | 2814.00 |         2044.20 |         1531.00
       33 | WA    | 1531.00 |         2044.20 |         1531.00
      800 | WA    | 1476.00 |         2044.20 |         1531.00
        1 | WA    | 1177.00 |         2044.20 |         1531.00
(7 rows)
```

Fungsi jendela PERCENTILE_DISC

PERCENTILE_DISC adalah fungsi distribusi terbalik yang mengasumsikan model distribusi diskrit. Dibutuhkan nilai persentil dan spesifikasi pengurutan dan mengembalikan elemen dari himpunan yang diberikan.

Untuk nilai persentil P yang diberikan, PERCENTILE_DISC mengurutkan nilai ekspresi dalam klausa ORDER BY dan mengembalikan nilai dengan nilai distribusi kumulatif terkecil (sehubungan dengan spesifikasi pengurutan yang sama) yang lebih besar dari atau sama dengan P.

Anda hanya dapat menentukan klausa PARTITION dalam klausa OVER.

Sintaksis

```
PERCENTILE_DISC ( percentile )
WITHIN GROUP (ORDER BY expr)
OVER (  [ PARTITION BY expr_list ]  )
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Menentukan numerik atau date/time nilai untuk mengurutkan dan menghitung persentil.

DI ATAS

Menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH expr

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Pengembalian

Jenis data yang sama dengan ekspresi ORDER BY dalam klausa WITHIN GROUP.

Contoh

Contoh berikut menggunakan tabel WINSALES. Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

```
SELECT sellerid, qty, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY qty)
OVER() AS MEDIAN FROM winsales;

+----------+-----+--------+
| sellerid | qty | median |
+----------+-----+--------+
| 3        | 10  | 20     |
| 1        | 10  | 20     |
| 1        | 10  | 20     |
| 4        | 10  | 20     |
| 3        | 15  | 20     |
| 2        | 20  | 20     |
| 2        | 20  | 20     |
| 3        | 20  | 20     |
| 1        | 30  | 20     |
| 3        | 30  | 20     |
| 4        | 40  | 20     |
+----------+-----+--------+

SELECT sellerid, qty, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS MEDIAN FROM winsales;

+----------+-----+--------+
| sellerid | qty | median |
+----------+-----+--------+
| 4        | 10  | 10     |
| 4        | 40  | 10     |
| 3        | 10  | 15     |
| 3        | 15  | 15     |
| 3        | 20  | 15     |
| 3        | 30  | 15     |
| 2        | 20  | 20     |
| 2        | 20  | 20     |
| 1        | 10  | 10     |
| 1        | 10  | 10     |
| 1        | 30  | 10     |
```

```
+----------+-----+--------+
```

Untuk menemukan PERCENTILE_DISC (0.25) dan PERCENTILE_DISC (0.75) untuk kuantitas saat dipartisi oleh ID penjual, gunakan contoh berikut.

```
SELECT sellerid, qty, PERCENTILE_DISC(0.25)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS quartile1 FROM winsales;

+----------+-----+-----------+
| sellerid | qty | quartile1 |
+----------+-----+-----------+
| 4        | 10  | 10        |
| 4        | 40  | 10        |
| 2        | 20  | 20        |
| 2        | 20  | 20        |
| 3        | 10  | 10        |
| 3        | 15  | 10        |
| 3        | 20  | 10        |
| 3        | 30  | 10        |
| 1        | 10  | 10        |
| 1        | 10  | 10        |
| 1        | 30  | 10        |
+----------+-----+-----------+

SELECT sellerid, qty, PERCENTILE_DISC(0.75)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS quartile3 FROM winsales;

+----------+-----+-----------+
| sellerid | qty | quartile3 |
+----------+-----+-----------+
| 3        | 10  | 20        |
| 3        | 15  | 20        |
| 3        | 20  | 20        |
| 3        | 30  | 20        |
| 4        | 10  | 40        |
| 4        | 40  | 40        |
| 2        | 20  | 20        |
| 2        | 20  | 20        |
| 1        | 10  | 30        |
| 1        | 10  | 30        |
| 1        | 30  | 30        |
```

```
+----------+-----+-----------+
```

Fungsi jendela RANK

Fungsi jendela RANK menentukan peringkat nilai dalam kelompok nilai, berdasarkan ekspresi ORDER BY dalam klausa OVER. Jika klausa PARTITION BY opsional hadir, peringkat diatur ulang untuk setiap kelompok baris. Baris dengan nilai yang sama untuk kriteria peringkat menerima peringkat yang sama. Amazon Redshift menambahkan jumlah baris terikat ke peringkat terikat untuk menghitung peringkat berikutnya dan dengan demikian peringkat mungkin bukan angka berturut-turut. Misalnya, jika dua baris diberi peringkat 1, peringkat berikutnya adalah 3.

RANK berbeda dari satu Fungsi jendela DENSE_RANK hal: Untuk DENSE_RANK, jika dua baris atau lebih berdampingan, tidak ada celah dalam urutan nilai peringkat. Misalnya, jika dua baris diberi peringkat 1, peringkat berikutnya adalah 2.

Anda dapat memiliki fungsi peringkat dengan klausa PARTITION BY dan ORDER BY yang berbeda dalam kueri yang sama.

Sintaksis

```
RANK () OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Klausa jendela untuk fungsi RANK.

PARTISI OLEH expr_list

Tidak wajib. Satu atau lebih ekspresi yang menentukan jendela.

PESAN BERDASARKAN order_list

Tidak wajib. Mendefinisikan kolom yang menjadi dasar nilai peringkat. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel. Jika ORDER BY dihilangkan, nilai pengembalian adalah 1 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris bersifat nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

INTEGER

Contoh

Contoh berikut mengurutkan tabel berdasarkan jumlah yang dijual (default naik), dan menetapkan peringkat untuk setiap baris. Nilai peringkat 1 adalah nilai peringkat tertinggi. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan:

```
select salesid, qty,
rank() over (order by qty) as rnk
from winsales
order by 2,1;

salesid | qty | rnk
--------+-----+-----
10001 |  10 |  1
10006 |  10 |  1
30001 |  10 |  1
40005 |  10 |  1
30003 |  15 |  5
20001 |  20 |  6
20002 |  20 |  6
30004 |  20 |  6
10005 |  30 |  9
30007 |  30 |  9
40001 |  40 |  11
(11 rows)
```

Perhatikan bahwa klausa ORDER BY luar dalam contoh ini menyertakan kolom 2 dan 1 untuk memastikan bahwa Amazon Redshift mengembalikan hasil yang diurutkan secara konsisten setiap kali kueri ini dijalankan. Misalnya, baris dengan ID penjualan 10001 dan 10006 memiliki nilai QTY

dan RNK yang identik. Memesan hasil akhir yang ditetapkan oleh kolom 1 memastikan bahwa baris 10001 selalu jatuh sebelum 10006. Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Dalam contoh berikut, urutan dibalik untuk fungsi window (`order by qty desc`). Sekarang nilai peringkat tertinggi berlaku untuk nilai QTY terbesar.

```
select salesid, qty,
rank() over (order by qty desc) as rank
from winsales
order by 2,1;

 salesid | qty | rank
---------+-----+-----
   10001 |  10 |   8
   10006 |  10 |   8
   30001 |  10 |   8
   40005 |  10 |   8
   30003 |  15 |   7
   20001 |  20 |   4
   20002 |  20 |   4
   30004 |  20 |   4
   10005 |  30 |   2
   30007 |  30 |   2
   40001 |  40 |   1
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut mempartisi tabel dengan SELLERID dan mengurutkan setiap partisi berdasarkan kuantitas (dalam urutan menurun) dan menetapkan peringkat untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
select salesid, sellerid, qty, rank() over
(partition by sellerid
order by qty desc) as rank
from winsales
order by 2,3,1;

salesid | sellerid | qty | rank
--------+----------+-----+-----
  10001 |        1 |  10 |  2
```

```
  10006 |        1 |  10 |  2
  10005 |        1 |  30 |  1
  20001 |        2 |  20 |  1
  20002 |        2 |  20 |  1
  30001 |        3 |  10 |  4
  30003 |        3 |  15 |  3
  30004 |        3 |  20 |  2
  30007 |        3 |  30 |  1
  40005 |        4 |  10 |  2
  40001 |        4 |  40 |  1
(11 rows)
```

Fungsi jendela RATIO_TO_REPORT

Menghitung rasio nilai dengan jumlah nilai di jendela atau partisi. Rasio terhadap nilai laporan ditentukan menggunakan rumus:

`value of` rasio_ekspresi rasio_ekspresi `argument for the current row / sum of argument for the window or partition`

Dataset berikut menggambarkan penggunaan rumus ini:

```
Row# Value Calculation RATIO_TO_REPORT
1 2500 (2500)/(13900) 0.1798
2 2600 (2600)/(13900) 0.1870
3 2800 (2800)/(13900) 0.2014
4 2900 (2900)/(13900) 0.2086
5 3100 (3100)/(13900) 0.2230
```

Rentang nilai pengembalian adalah 0 hingga 1, inklusif. Jika ratio_expression adalah NULL, maka nilai kembalinya adalah. NULL Jika nilai dalam partition_expression unik, maka fungsi akan kembali 1 untuk nilai itu.

Sintaksis

```
RATIO_TO_REPORT ( ratio_expression )
OVER ( [ PARTITION BY partition_expression ] )
```

Argumen

rasio_ekspresi

Sebuah ekspresi, seperti nama kolom, yang memberikan nilai untuk menentukan rasio. Ekspresi harus memiliki tipe data numerik atau secara implisit dapat dikonversi menjadi satu.

Anda tidak dapat menggunakan fungsi analitik lainnya di ratio_expression.

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI DENGAN partition_expression

Tidak wajib. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Jenis pengembalian

FLOAT8

Contoh

Contoh berikut menggunakan tabel WINSALES. Untuk informasi tentang cara membuat tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut menghitung nilai rasio terhadap laporan dari setiap baris kuantitas penjual dengan total jumlah semua penjual.

```
select sellerid, qty, ratio_to_report(qty)
over()
from winsales
order by sellerid;

sellerid  qty    ratio_to_report
-------------------------------------
1         30     0.13953488372093023
1         10     0.046511627906976744
1         10     0.046511627906976744
2         20     0.09302325581395349
2         20     0.09302325581395349
```

```
3          30     0.13953488372093023
3          20     0.09302325581395349
3          15     0.06976744186046512
3          10     0.046511627906976744
4          10     0.046511627906976744
4          40     0.18604651162790697
```

Contoh berikut menghitung rasio jumlah penjualan untuk setiap penjual berdasarkan partisi.

```
select sellerid, qty, ratio_to_report(qty)
over(partition by sellerid)
from winsales;

sellerid   qty    ratio_to_report
-----------------------------------------
2          20     0.5
2          20     0.5
4          40     0.8
4          10     0.2
1          10     0.2
1          30     0.6
1          10     0.2
3          10     0.13333333333333333
3          15     0.2
3          20     0.26666666666666666
3          30     0.4
```

Fungsi jendela ROW_NUMBER

Menetapkan nomor ordinal dari baris saat ini dalam kelompok baris, dihitung dari 1, berdasarkan ekspresi ORDER BY dalam klausa OVER. Jika klausa PARTITION BY opsional hadir, nomor ordinal diatur ulang untuk setiap kelompok baris. Baris dengan nilai yang sama untuk ekspresi ORDER BY menerima nomor baris yang berbeda secara nondeterministik.

Sintaksis

```
ROW_NUMBER() OVER(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Klausa fungsi jendela untuk fungsi ROW_NUMBER.

PARTISI OLEH expr_list

Tidak wajib. Satu atau lebih ekspresi kolom yang membagi hasil menjadi kumpulan baris.

PESANAN BERDASARKAN order_list

Tidak wajib. Satu atau lebih ekspresi kolom yang mendefinisikan urutan baris dalam satu set. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

Jika ORDER BY tidak menghasilkan urutan unik atau dihilangkan, urutan baris tidak deterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

BIGINT

Contoh

Contoh-contoh berikut menggunakan `WINSALES` tabel. Untuk deskripsi `WINSALES` tabel, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut mengurutkan tabel berdasarkan QTY (dalam urutan menaik), lalu menetapkan nomor baris untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
SELECT salesid, sellerid, qty,
ROW_NUMBER() OVER(
   ORDER BY qty ASC) AS row
FROM winsales
ORDER BY 4,1;

salesid   sellerid   qty   row
---------+----------+-----+-----
   30001 |        3 |  10 |   1
```

```
   10001 |        1 |  10 |   2
   10006 |        1 |  10 |   3
   40005 |        4 |  10 |   4
   30003 |        3 |  15 |   5
   20001 |        2 |  20 |   6
   20002 |        2 |  20 |   7
   30004 |        3 |  20 |   8
   10005 |        1 |  30 |   9
   30007 |        3 |  30 |  10
   40001 |        4 |  40 |  11
```

Contoh berikut mempartisi tabel dengan SELLERID dan mengurutkan setiap partisi berdasarkan QTY (dalam urutan menaik), lalu menetapkan nomor baris untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
SELECT salesid, sellerid, qty,
ROW_NUMBER() OVER(
  PARTITION BY sellerid
  ORDER BY qty ASC) AS row_by_seller
FROM winsales
ORDER BY 2,4;

 salesid | sellerid | qty | row_by_seller
---------+----------+-----+-----
   10001 |        1 |  10 |   1
   10006 |        1 |  10 |   2
   10005 |        1 |  30 |   3
   20001 |        2 |  20 |   1
   20002 |        2 |  20 |   2
   30001 |        3 |  10 |   1
   30003 |        3 |  15 |   2
   30004 |        3 |  20 |   3
   30007 |        3 |  30 |   4
   40005 |        4 |  10 |   1
   40001 |        4 |  40 |   2
```

Contoh berikut menunjukkan hasil saat tidak menggunakan klausa opsional.

```
SELECT salesid, sellerid, qty, ROW_NUMBER() OVER() AS row
FROM winsales
ORDER BY 4,1;
```

```
salesid   sellerid   qty   row
---------+----------+-----+-----
   30001 |        3 |  10 |   1
   10001 |        1 |  10 |   2
   10005 |        1 |  30 |   3
   40001 |        4 |  40 |   4
   10006 |        1 |  10 |   5
   20001 |        2 |  20 |   6
   40005 |        4 |  10 |   7
   20002 |        2 |  20 |   8
   30003 |        3 |  15 |   9
   30004 |        3 |  20 |  10
   30007 |        3 |  30 |  11
```

Fungsi jendela STDDEV_SAMP dan STDDEV_POP

Fungsi jendela STDDEV_SAMP dan STDDEV_POP mengembalikan sampel dan standar deviasi populasi dari satu set nilai numerik (integer, desimal, atau floating-point). Lihat juga Fungsi STDDEV_SAMP dan STDDEV_POP.

STDDEV_SAMP dan STDDEV adalah sinonim untuk fungsi yang sama.

Sintaksis

```
STDDEV_SAMP | STDDEV | STDDEV_POP
( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Jenis argumen yang didukung oleh fungsi STDDEV adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Terlepas dari tipe data ekspresi, tipe pengembalian fungsi STDDEV adalah angka presisi ganda.

Contoh

Contoh berikut menunjukkan cara menggunakan fungsi STDDEV_POP dan VAR_POP sebagai fungsi jendela. Kueri menghitung varians populasi dan standar deviasi populasi untuk nilai PRICEPAID dalam tabel SALES.

```
select salesid, dateid, pricepaid,
round(stddev_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as stddevpop,
round(var_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as varpop
from sales
order by 2,1;

salesid | dateid | pricepaid | stddevpop | varpop
```

```
--------+--------+-----------+-----------+---------
  33095 |   1827 |    234.00 |         0 |       0
  65082 |   1827 |    472.00 |       119 |   14161
  88268 |   1827 |    836.00 |       248 |   61283
  97197 |   1827 |    708.00 |       230 |   53019
 110328 |   1827 |    347.00 |       223 |   49845
 110917 |   1827 |    337.00 |       215 |   46159
 150314 |   1827 |    688.00 |       211 |   44414
 157751 |   1827 |   1730.00 |       447 |  199679
 165890 |   1827 |   4192.00 |      1185 | 1403323
...
```

Deviasi standar sampel dan fungsi varians dapat digunakan dengan cara yang sama.

Fungsi jendela SUM

Fungsi jendela SUM mengembalikan jumlah kolom input atau nilai ekspresi. Fungsi SUM bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaksis

```
SUM ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi SUM dalam hal satu atau lebih ekspresi.

PESAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Jenis argumen yang didukung oleh fungsi SUM adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Jenis pengembalian yang didukung oleh fungsi SUM adalah:

- BIGINT untuk argumen SMALINT atau INTEGER
- NUMERIK untuk argumen BIGINT
- PRESISI GANDA untuk argumen floating-point

Contoh

Contoh berikut membuat jumlah kumulatif (bergulir) dari jumlah penjualan yang diurutkan berdasarkan tanggal dan ID penjualan:

```
select salesid, dateid, sellerid, qty,
sum(qty) over (order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | sum
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  20
```

```
10005 | 2003-12-24 |        1 |  30 |  50
40001 | 2004-01-09 |        4 |  40 |  90
10006 | 2004-01-18 |        1 |  10 | 100
20001 | 2004-02-12 |        2 |  20 | 120
40005 | 2004-02-12 |        4 |  10 | 130
20002 | 2004-02-16 |        2 |  20 | 150
30003 | 2004-04-18 |        3 |  15 | 165
30004 | 2004-04-18 |        3 |  20 | 185
30007 | 2004-09-07 |        3 |  30 | 215
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut membuat jumlah kumulatif (bergulir) jumlah penjualan berdasarkan tanggal, mempartisi hasil berdasarkan ID penjual, dan mengurutkan hasil berdasarkan tanggal dan ID penjualan di dalam partisi:

```
select salesid, dateid, sellerid, qty,
sum(qty) over (partition by sellerid
order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | sum
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  10
10005 | 2003-12-24 |        1 |  30 |  40
40001 | 2004-01-09 |        4 |  40 |  40
10006 | 2004-01-18 |        1 |  10 |  50
20001 | 2004-02-12 |        2 |  20 |  20
40005 | 2004-02-12 |        4 |  10 |  50
20002 | 2004-02-16 |        2 |  20 |  40
30003 | 2004-04-18 |        3 |  15 |  25
30004 | 2004-04-18 |        3 |  20 |  45
30007 | 2004-09-07 |        3 |  30 |  75
(11 rows)
```

Contoh berikut menamai semua baris secara berurutan dalam kumpulan hasil, diurutkan berdasarkan kolom SELLERID dan SALESID:

```
select salesid, sellerid, qty,
```

```
sum(1) over (order by sellerid, salesid rows unbounded preceding) as rownum
from winsales
order by 2,1;

salesid | sellerid |  qty | rownum
--------+----------+------+--------
10001 |        1 |   10 |     1
10005 |        1 |   30 |     2
10006 |        1 |   10 |     3
20001 |        2 |   20 |     4
20002 |        2 |   20 |     5
30001 |        3 |   10 |     6
30003 |        3 |   15 |     7
30004 |        3 |   20 |     8
30007 |        3 |   30 |     9
40001 |        4 |   40 |    10
40005 |        4 |   10 |    11
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatTabel sampel untuk contoh fungsi jendela.

Contoh berikut memberi nomor semua baris secara berurutan dalam kumpulan hasil, mempartisi hasil dengan SELLERID, dan mengurutkan hasil berdasarkan SELLERID dan SALESID dalam partisi:

```
select salesid, sellerid, qty,
sum(1) over (partition by sellerid
order by sellerid, salesid rows unbounded preceding) as rownum
from winsales
order by 2,1;

salesid | sellerid | qty | rownum
---------+----------+-----+--------
10001 |        1 |  10 |      1
10005 |        1 |  30 |      2
10006 |        1 |  10 |      3
20001 |        2 |  20 |      1
20002 |        2 |  20 |      2
30001 |        3 |  10 |      1
30003 |        3 |  15 |      2
30004 |        3 |  20 |      3
30007 |        3 |  30 |      4
40001 |        4 |  40 |      1
```

```
40005 |        4 |  10 |      2
(11 rows)
```

Fungsi jendela VAR_SAMP dan VAR_POP

Fungsi jendela VAR_SAMP dan VAR_POP mengembalikan varian sampel dan populasi dari sekumpulan nilai numerik (integer, desimal, atau floating-point). Lihat juga Fungsi VAR_SAMP dan VAR_POP.

VAR_SAMP dan VARIANCE adalah sinonim untuk fungsi yang sama.

Sintaksis

```
VAR_SAMP | VARIANCE | VAR_POP
( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_klausa

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa frame eksplisit diperlukan. Klausa frame memurnikan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis Data

Jenis argumen yang didukung oleh fungsi VARIANCE adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Terlepas dari tipe data ekspresi, tipe pengembalian fungsi VARIANCE adalah angka presisi ganda.

Fungsi administrasi sistem

Topik

- UBAH_KUERI_PRIORITAS
- UBAH_PRIORITAS_SESI
- UBAH_PRIORITAS_PENGGUNA
- SETELINGAN_SAAT INI
- PG_CANCEL_BACKEND
- PG_TERMINATE_BACKEND
- BOOT_CLUSTER
- SETEL_KONFIGURASI

Amazon Redshift mendukung beberapa fungsi administrasi sistem.

UBAH_KUERI_PRIORITAS

CHANGE_QUERY_PRIORITY memungkinkan pengguna super untuk memodifikasi prioritas kueri yang sedang berjalan atau menunggu dalam manajemen beban kerja (WLM).

Fungsi ini memungkinkan pengguna super untuk segera mengubah prioritas permintaan apa pun dalam sistem. Hanya satu kueri, pengguna, atau sesi yang dapat dijalankan dengan prioritas`CRITICAL`.

Sintaksis

```
CHANGE_QUERY_PRIORITY(query_id, priority)
```

Argumen

Query_id

Pengidentifikasi kueri dari kueri yang prioritasnya diubah. Membutuhkan `INTEGER` nilai.

prioritas

Prioritas baru yang akan ditetapkan ke kueri. Argumen ini harus berupa string dengan nilai`CRITICAL`,`HIGHEST`,`HIGH`,`NORMAL`,`LOW`, atau`LOWEST`.

Tipe Pengembalian

Tidak ada

Contoh

Untuk menampilkan kolom `query_priority` dalam tabel sistem STV_WLM_QUERY_STATE, gunakan contoh berikut.

```
SELECT query, service_class, query_priority, state
FROM stv_wlm_query_state WHERE service_class = 101;

+-------+---------------+----------------+---------+
| query | service_class | query_priority |  state  |
+-------+---------------+----------------+---------+
|  1076 |           101 | Lowest         | Running |
|  1075 |           101 | Lowest         | Running |
+-------+---------------+----------------+---------+
```

Untuk menunjukkan hasil superuser menjalankan fungsi `change_query_priority` untuk mengubah prioritas`CRITICAL`, gunakan contoh berikut.

```
SELECT CHANGE_QUERY_PRIORITY(1076, 'Critical');

+-------------------------------------------------------------------------------+
|                            change_query_priority                              |
+-------------------------------------------------------------------------------+
| Succeeded to change query priority. Priority changed from Lowest to Critical. |
```

```
+-------------------------------------------------------------------------------+
```

UBAH_PRIORITAS_SESI

CHANGE_SESSION_PRIORITY memungkinkan pengguna super untuk segera mengubah prioritas sesi apa pun dalam sistem. Hanya satu sesi, pengguna, atau kueri yang dapat dijalankan dengan prioritas`CRITICAL`.

Sintaksis

```
CHANGE_SESSION_PRIORITY(pid, priority)
```

Argumen

pid

Pengidentifikasi proses sesi yang prioritasnya diubah. Nilai `-1` mengacu pada sesi saat ini. Membutuhkan `INTEGER` nilai.

prioritas

Prioritas baru yang akan ditetapkan ke sesi. Argumen ini harus berupa string dengan nilai`CRITICAL`,`HIGHEST`,`HIGH`,`NORMAL`,`LOW`, atau`LOWEST`.

Jenis pengembalian

Tidak ada

Contoh

Untuk mengembalikan pengidentifikasi proses dari proses server yang menangani sesi saat ini, gunakan contoh berikut.

```
SELECT pg_backend_pid();

+----------------+
| pg_backend_pid |
+----------------+
|          30311 |
+----------------+
```

Dalam contoh ini, prioritas diubah menjadi `LOWEST` untuk sesi saat ini.

```
SELECT CHANGE_SESSION_PRIORITY(30311, 'Lowest');

+-------------------------------------------------------------------------------
+
|                               change_session_priority
 |
+-------------------------------------------------------------------------------
+
| Succeeded to change session priority. Changed session (pid:30311) priority to lowest.
 |
+-------------------------------------------------------------------------------
+
```

Dalam contoh ini, prioritas diubah menjadi `HIGH` untuk sesi saat ini.

```
SELECT CHANGE_SESSION_PRIORITY(-1, 'High');

+-------------------------------------------------------------------------------
+
|                                 change_session_priority
           |
+-------------------------------------------------------------------------------
+
| Succeeded to change session priority. Changed session (pid:30311) priority from
 lowest to high. |
+-------------------------------------------------------------------------------
+
```

Untuk membuat prosedur tersimpan yang mengubah prioritas sesi, gunakan contoh berikut. Izin untuk menjalankan prosedur tersimpan ini diberikan kepada pengguna database`test_user`.

```
CREATE OR REPLACE PROCEDURE sp_priority_low(pid IN int, result OUT varchar)
AS $$
BEGIN
  SELECT CHANGE_SESSION_PRIORITY(pid, 'low') into result;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER;
GRANT EXECUTE ON PROCEDURE sp_priority_low(int) TO test_user;
```

Kemudian pengguna database bernama `test_user` memanggil prosedur.

```
CALL sp_priority_low(pg_backend_pid());

+-----------------------------------------------------+
|                        result                       |
+-----------------------------------------------------+
| Success. Change session (pid:13155) priority to low. |
+-----------------------------------------------------+
```

UBAH_PRIORITAS_PENGGUNA

CHANGE_USER_PRIORITY memungkinkan pengguna super untuk memodifikasi prioritas semua kueri yang dikeluarkan oleh pengguna yang sedang berjalan atau menunggu dalam manajemen beban kerja (WLM). Hanya satu pengguna, sesi, atau kueri yang dapat dijalankan dengan prioritas`CRITICAL`.

Sintaksis

```
CHANGE_USER_PRIORITY(user_name, priority)
```

Argumen

nama_pengguna

Nama pengguna database yang prioritas kuerinya diubah.

prioritas

Prioritas baru yang akan ditetapkan untuk semua kueri yang dikeluarkan oleh`user_name`. Argumen ini harus berupa string dengan nilai`CRITICAL`,`HIGHEST`,`HIGH`,`NORMAL`,`LOW`,`LOWEST`, atau`RESET`. Hanya pengguna super yang dapat mengubah prioritas menjadi`CRITICAL`. Mengubah prioritas untuk `RESET` menghapus pengaturan prioritas untuk`user_name`.

Jenis pengembalian

Tidak ada

Contoh

Untuk mengubah prioritas pengguna `analysis_user`LOWEST, gunakan contoh berikut.

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'lowest');
```

```
+-------------------------------------------------------------------------------------+
|                               change_user_priority                                  |
+-------------------------------------------------------------------------------------+
| Succeeded to change user priority. Changed user (analysis_user) priority to lowest. |
+-------------------------------------------------------------------------------------+
```

Untuk mengubah prioritas menjadiL0W, gunakan contoh berikut.

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'low');

+-------------------------------------------------------------------------------------------------------------
+
|                                    change_user_priority
        |
+-------------------------------------------------------------------------------------------------------------
+
| Succeeded to change user priority. Changed user (analysis_user) priority from Lowest
 to low. |
+-------------------------------------------------------------------------------------------------------------
+
```

Untuk mengatur ulang prioritas, gunakan contoh berikut.

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'reset');

+-------------------------------------------------------+
|                change_user_priority                   |
+-------------------------------------------------------+
| Succeeded to reset priority for user (analysis_user). |
+-------------------------------------------------------+
```

SETELINGAN_SAAT INI

CURRENT_SETTING mengembalikan nilai saat ini dari parameter konfigurasi yang ditentukan.

Fungsi ini setara dengan MENUNJUKKAN perintah.

Sintaksis

```
current_setting('parameter')
```

Pernyataan berikut mengembalikan nilai saat ini dari variabel konteks sesi yang ditentukan.

```
current_setting('variable_name')
current_setting('variable_name'[, error_if_undefined])
```

Argumen

Parameter

Nilai parameter untuk ditampilkan. Untuk daftar parameter konfigurasi, lihat Referensi konfigurasi

nama_variabel

Nama variabel yang akan ditampilkan. Ini harus menjadi konstanta string untuk variabel konteks sesi.

error_jika_undefined

(Opsional) Nilai boolean yang menentukan perilaku jika nama variabel tidak ada. Ketika error_if_undefined disetel keTRUE, yang merupakan default, Amazon Redshift menampilkan kesalahan. Ketika error_if_undefined disetel ke, Amazon Redshift kembali. FALSE NULL Amazon Redshift mendukung parameter error_if_undefined hanya untuk variabel konteks sesi. Ini tidak dapat digunakan ketika input adalah parameter konfigurasi.

Jenis pengembalian

Mengembalikan VARCHAR string CHAR atau.

Contoh

Untuk mengembalikan pengaturan saat ini untuk query_group parameter, gunakan contoh berikut.

```
SELECT CURRENT_SETTING('query_group');

+-----------------+
| current_setting |
+-----------------+
| unset           |
+-----------------+
```

Untuk mengembalikan pengaturan saat ini untuk variabelapp_context.user_id, gunakan contoh berikut.

```
SELECT CURRENT_SETTING('app_context.user_id', FALSE);
```

PG_CANCEL_BACKEND

Membatalkan kueri. PG_CANCEL_BACKEND secara fungsional setara dengan perintah. CANCEL (BATALKAN) Anda dapat membatalkan kueri yang sedang dijalankan oleh pengguna Anda. Pengguna super dapat membatalkan kueri apa pun.

Sintaksis

```
pg_cancel_backend( pid )
```

Argumen

pid

ID proses (PID) dari kueri yang akan dibatalkan. Anda tidak dapat membatalkan kueri dengan menentukan ID kueri; Anda harus menentukan ID proses kueri. Membutuhkan `INTEGER` nilai.

Jenis pengembalian

Tidak ada

Catatan penggunaan

Jika kueri dalam beberapa sesi menahan kunci pada tabel yang sama, Anda dapat menggunakan PG_TERMINATE_BACKEND fungsi untuk mengakhiri salah satu sesi, yang memaksa transaksi yang sedang berjalan dalam sesi yang dihentikan untuk melepaskan semua kunci dan mengembalikan transaksi. Kueri tabel katalog PG__LOCKS untuk melihat kunci yang saat ini dipegang. Jika Anda tidak dapat membatalkan kueri karena berada di blok transaksi (BEGIN... END), Anda dapat mengakhiri sesi di mana kueri berjalan dengan menggunakan fungsi PG_TERMINATE_BACKEND.

Contoh

Untuk membatalkan kueri yang sedang berjalan, pertama-tama ambil ID proses untuk kueri yang ingin Anda batalkan. Untuk menentukan ID proses untuk semua kueri yang sedang berjalan, jalankan perintah berikut.

```
SELECT pid, TRIM(starttime) AS start,
duration, TRIM(user_name) AS user,
SUBSTRING(query,1,40) AS querytxt
FROM stv_recents
```

```
WHERE status = 'Running';

+-----+----------------------+----------+--------+----------------------------+
| pid |       starttime      | duration |  user  |          querytxt          |
+-----+----------------------+----------+--------+----------------------------+
| 802 | 2013-10-14 09:19:03.55 |      132 | dwuser | select venuename from venue |
| 834 | 2013-10-14 08:33:49.47 |  1250414 | dwuser | select * from listing;      |
| 964 | 2013-10-14 08:30:43.29 |   326179 | dwuser | select sellerid from sales  |
+-----+----------------------+----------+--------+----------------------------+
```

Untuk membatalkan kueri dengan ID proses 802, gunakan contoh berikut.

```
SELECT PG_CANCEL_BACKEND(802);
```

PG_TERMINATE_BACKEND

Mengakhiri sesi. Anda dapat mengakhiri sesi yang dimiliki oleh pengguna Anda. Seorang superuser dapat menghentikan sesi apa pun.

Sintaksis

```
pg_terminate_backend( pid )
```

Argumen

pid

ID proses sesi yang akan dihentikan. Membutuhkan `INTEGER` nilai.

Jenis pengembalian

Tidak ada

Catatan penggunaan

Jika Anda hampir mencapai batas untuk koneksi bersamaan, gunakan PG_TERMINATE_BACKEND untuk mengakhiri sesi idle dan membebaskan koneksi. Untuk informasi selengkapnya, lihat Batas di Amazon Redshift.

Jika kueri dalam beberapa sesi menahan kunci pada tabel yang sama, Anda dapat menggunakan PG_TERMINATE_BACKEND untuk mengakhiri salah satu sesi, yang memaksa transaksi yang

sedang berjalan dalam sesi yang dihentikan untuk melepaskan semua kunci dan mengembalikan transaksi. Kueri tabel katalog PG_LOCKS untuk melihat kunci yang saat ini dipegang.

Jika kueri tidak ada dalam blok transaksi (BEGIN... END), Anda dapat membatalkan kueri dengan menggunakan CANCEL (BATALKAN) perintah atau PG_CANCEL_BACKEND fungsi.

Contoh

Untuk menanyakan tabel SVV_TRANSACTIONS untuk melihat semua kunci yang berlaku untuk transaksi saat ini, gunakan contoh berikut.

```
SELECT * FROM svv_transactions;

+----------+--------+-------+------+--------------------+----------------
+----------------------+----------+---------+
| txn_owner | txn_db |  xid  | pid  |      txn_start      |    lock_mode    |
 lockable_object_type | relation | granted |
+----------+--------+-------+------+--------------------+----------------
+----------------------+----------+---------+
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
           |    51940 | true    |
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
           |    52000 | true    |
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
           |   108623 | true    |
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | ExclusiveLock   |
 transactionid        |          | true    |
+----------+--------+-------+------+--------------------+----------------
+----------------------+----------+---------+
```

UNTUK mengakhiri sesi memegang kunci, gunakan contoh berikut.

```
SELECT PG_TERMINATE_BACKEND(8585);
```

BOOT_CLUSTER

Mulai ulang cluster Amazon Redshift tanpa menutup koneksi ke cluster. Anda harus menjadi superuser database untuk menjalankan perintah ini.

Setelah soft reboot ini selesai, cluster Amazon Redshift mengembalikan kesalahan ke aplikasi pengguna dan mengharuskan aplikasi pengguna untuk mengirimkan kembali transaksi atau kueri apa pun yang terganggu oleh soft reboot.

Sintaksis

```
SELECT REBOOT_CLUSTER();
```

SETEL_KONFIGURASI

Menetapkan parameter konfigurasi ke pengaturan baru.

Fungsi ini setara dengan perintah SET di SQL.

Sintaksis

```
SET_CONFIG('parameter', 'new_value' , is_local)
```

Pernyataan berikut menetapkan variabel konteks sesi ke pengaturan baru.

```
set_config('variable_name', 'new_value' , is_local)
```

Argumen

parameter

Parameter untuk diatur.

nama_variabel

Nama variabel yang akan ditetapkan.

nilai_baru

Nilai baru dari parameter.

adalah_lokal

Jika benar, nilai parameter hanya berlaku untuk transaksi saat ini. Nilai yang valid adalah `true` atau `1` dan `false` atau`0`.

Jenis pengembalian

Mengembalikan VARCHAR string CHAR atau.

Contoh

Untuk mengatur nilai `query_group` parameter `test` untuk transaksi saat ini saja, gunakan contoh berikut.

```
SELECT SET_CONFIG('query_group', 'test', true);

+------------+
| set_config |
+------------+
| test       |
+------------+
```

Untuk mengatur variabel konteks sesi, gunakan contoh berikut.

```
SELECT SET_CONFIG(‘app.username’, ‘cuddy’, FALSE);
```

Fungsi informasi sistem

Amazon Redshift mendukung berbagai fungsi informasi sistem.

Topik

- AKUN_AWS_SAAT INI
- DATABASE_SAAT INI
- RUANG NAMA SAAT INI
- CURRENT_SCHEMA
- SKEMA SAAT INI
- SESSION_SAAT INI_ARN
- CURRENT_USER
- USER_ID SAAT INI
- DB_COLLATION
- DEFAULT_IAM_ROLE
- GET_MOUNTED_ROLE
- MEMILIKI HAK ASSUMEROLE
- MEMILIKI HAK ISTIMEWA BASIS DATA
- MEMILIKI HAK ISTIMEWA SKEMA

- MEMILIKI HAK ISTIMEWA TABEL_
- ID KUERI PENGGUNA TERAKHIR
- PG_BACKEND_PID
- PG_GET_COLS
- PG_DAPAT_PENERIMA HIBAH OLEH_IAM_ROLE
- PG_GET_IAM_ROL_OLE_OLE_PENGGUNA
- PG_GET_LATE_BINDING_VIEW_COLS
- PG_GET_SESSION_ROLE
- PG_LAST_COPY_COUNT
- PG_LAST_COPY_ID
- PG_LAST_UNLOAD_ID
- PG_LAST_QUERY_ID
- PG_LAST_UNLOAD_COUNT
- Fungsi SLICE_NUM
- USER
- ROL_IS_ANGGOTA_DARI
- PENGGUNA_ADALAH_ANGGOTA_DARI
- VERSI

AKUN_AWS_SAAT INI

Mengembalikan AWS akun yang terkait dengan cluster Amazon Redshift yang mengirimkan kueri.

Sintaksis

```
current_aws_account
```

Jenis pengembalian

Mengembalikan bilangan bulat.

Contoh

Query berikut mengembalikan nama database saat ini.

```
select user, current_aws_account;
current_user | current_account
-------------+---------------
dwuser       | 987654321

(1 row)
```

DATABASE_SAAT INI

Mengembalikan nama database tempat Anda saat ini terhubung.

Sintaksis

```
current_database()
```

Jenis pengembalian

Mengembalikan string CHAR atau VARCHAR.

Contoh

Query berikut mengembalikan nama database saat ini.

```
select current_database();

current_database
------------------
tickit
(1 row)
```

RUANG NAMA SAAT INI

Mengembalikan namespace cluster Amazon Redshift saat ini. Namespace cluster Amazon Redshift adalah ID unik dari cluster Amazon Redshift.

Sintaksis

```
current_namespace
```

Jenis pengembalian

Mengembalikan string CHAR atau VARCHAR.

Contoh

Query berikut mengembalikan nama namespace saat ini.

```
select user, current_namespace;
current_user | current_namespace
-------------+--------------------------------------
dwuser       | 86b5169f-01dc-4a6f-9fbb-e2e24359e9a8

(1 row)
```

CURRENT_SCHEMA

Mengembalikan nama skema di bagian depan jalur pencarian. Skema ini akan digunakan untuk setiap tabel atau objek bernama lainnya yang dibuat tanpa menentukan skema target.

Sintaksis

> **Note**
>
> Ini adalah fungsi leader-node. Fungsi ini mengembalikan kesalahan jika mereferensikan tabel yang dibuat pengguna, tabel sistem STL atau STV, atau tampilan sistem SVV atau SVL.

```
current_schema()
```

Jenis pengembalian

CURRENT_SCHEMA mengembalikan string CHAR atau VARCHAR.

Contoh

Kueri berikut mengembalikan skema saat ini:

```
select current_schema();

current_schema
---------------
public
```

```
(1 row)
```

SKEMA SAAT INI

Mengembalikan array nama skema apa pun di jalur pencarian saat ini. Jalur pencarian saat ini ditentukan dalam parameter search_path.

Sintaksis

> **Note**
>
> Ini adalah fungsi leader-node. Fungsi ini mengembalikan kesalahan jika mereferensikan tabel yang dibuat pengguna, tabel sistem STL atau STV, atau tampilan sistem SVV atau SVL.

```
current_schemas(include_implicit)
```

Pendapat

termasukan_implisit

Jika benar, menentukan bahwa jalur pencarian harus menyertakan skema sistem yang disertakan secara implisit. Nilai yang valid adalah `true` dan `false`. Biasanya, jika`true`, parameter ini mengembalikan `pg_catalog` skema selain skema saat ini.

Jenis pengembalian

Mengembalikan string CHAR atau VARCHAR.

Contoh

Contoh berikut mengembalikan nama-nama skema di jalur pencarian saat ini, tidak termasuk skema sistem yang disertakan secara implisit:

```
select current_schemas(false);

current_schemas
-----------------
{public}
```

```
(1 row)
```

Contoh berikut mengembalikan nama-nama skema di jalur pencarian saat ini, termasuk skema sistem yang disertakan secara implisit:

```
select current_schemas(true);

current_schemas
---------------------
{pg_catalog,public}
(1 row)
```

SESSION_SAAT INI_ARN

Mengembalikan ARN pengguna global yang saat ini diotorisasi. Pengguna global hadir dengan identitas yang sama di seluruh akun Redshift, cluster, dan grup kerja Tanpa Server. Pengguna global masuk melalui Pusat Identitas IAM, atau melalui otentikasi IAM-based sesi. Pengguna danau data adalah AWS pengguna global.

Fungsi ini biasanya digunakan dalam konteks penggunaan tampilan multi-dialek AWS Glue . Untuk informasi selengkapnya tentang manajemen identitas dengan IAM Identity Center dan Redshift, lihat Menghubun gkan Redshift dengan Pusat Identitas IAM untuk memberi pengguna pengalaman masuk tunggal. Untuk informasi selengkapnya tentang tampilan Glue multi-dialek Membuat tampilan di Katalog AWS Glue Data.

Sintaksis

```
current_session_arn()
```

Jenis pengembalian

Mengembalikan string VARCHAR dari pengguna yang diautentikasi secara global atau nilai null.

Catatan penggunaan

Pengguna lokal tidak didukung dan menghasilkan respons nol.

Contoh

Kueri berikut mengembalikan nama sesi ARN saat ini:

```
SELECT current_session_arn();

current_session_arn
--------------
arn:aws:iam::123456789012:user/user
(1 row)
```

CURRENT_USER

Mengembalikan nama pengguna dari pengguna database “efektif” saat ini, sebagaimana berlaku untuk memeriksa izin. Biasanya, nama pengguna ini akan sama dengan pengguna sesi; Namun, ini kadang-kadang dapat diubah oleh pengguna super.

> **Note**
>
> Jangan gunakan tanda kurung akhir saat memanggil CURRENT_USER.

Sintaksis

```
current_user
```

Jenis pengembalian

CURRENT_USER mengembalikan tipe data NAME dan dapat di-cast sebagai string CHAR atau VARCHAR.

Catatan penggunaan

Jika prosedur tersimpan dibuat menggunakan opsi SECURITY DEFINER dari perintah CREATE_PROCEDURE, saat memanggil fungsi CURRENT_USER dari dalam prosedur yang disimpan, Amazon Redshift mengembalikan nama pengguna pemilik prosedur tersimpan.

Contoh

Kueri berikut mengembalikan nama pengguna database saat ini:

```
select current_user;
```

```
current_user
--------------
dwuser
(1 row)
```

USER_ID SAAT INI

Mengembalikan pengenal unik untuk pengguna Amazon Redshift yang masuk ke sesi saat ini.

Sintaksis

```
CURRENT_USER_ID
```

Jenis pengembalian

Fungsi CURRENT_USER_ID mengembalikan integer.

Contoh

Contoh berikut mengembalikan nama pengguna dan ID pengguna saat ini untuk sesi ini:

```
select user, current_user_id;

 current_user | current_user_id
--------------+-----------------
   dwuser     |               1
(1 row)
```

DB_COLLATION

Mengembalikan pengaturan kolasi dari database saat ini.

Sintaksis

```
db_collation()
```

Jenis pengembalian

Mengembalikan string VARCHAR yang mewakili kolasi database saat ini. Nilai yang mungkin adalah `case_sensitive` atau `case_insensitive`.

Contoh

Contoh berikut mengembalikan kolasi database saat ini.

```
select db_collation();

db_collation
----------------
case_sensitive
(1 row)
```

DEFAULT_IAM_ROLE

Mengembalikan peran IAM default yang saat ini terkait dengan cluster Amazon Redshift. Fungsi mengembalikan none jika tidak ada peran IAM default yang terkait.

Sintaksis

```
select default_iam_role();
```

Jenis pengembalian

Mengembalikan string VARCHAR.

Contoh

Contoh berikut mengembalikan peran IAM default yang saat ini terkait dengan cluster Amazon Redshift tertentu,

```
select default_iam_role();
              default_iam_role
-----------------------------------------------
 arn:aws:iam::123456789012:role/myRedshiftRole
(1 row)
```

GET_MOUNTED_ROLE

Ketika dipanggil sebagai bagian dari tampilan multi-dialek AWS Glue , ini memungkinkan pengembalian peran IAM yang digunakan untuk memasang skema atau database Lake Formation. Multi-dialect berarti bahwa SQL didukung di beberapa mesin kueri, seperti Amazon EMR dan

Redshift. Untuk informasi selengkapnya tentang tampilan Glue multi-dialek, lihat Membuat tampilan di Katalog AWS Glue Data.

Sintaksis

```
get_mounted_role()
```

Jenis pengembalian

Mengembalikan string VARCHAR atau nilai null.

Catatan penggunaan

Fungsi ini mengembalikan null untuk kasus penggunaan apa pun di luar tampilan Formasi Danau eksternal.

Contoh

Kueri berikut mengembalikan identitas untuk memasang sumber daya Formasi Danau.

```
CREATE EXTERNAL PROTECTED VIEW external_schema.remote_view AS
SELECT mycol, get_mounted_role() FROM external_schema.remote_table;

mycol | get_mounted_role
---------------------------
1       arn:aws:iam::123456789012:role/salesrole
(1 row)
```

MEMILIKI HAK ASSUMEROLE

Mengembalikan Boolean `true` (`t`) jika pengguna yang ditentukan memiliki peran IAM yang ditentukan dengan hak istimewa untuk menjalankan perintah yang ditentukan. Fungsi mengembalikan `false` (`f`) jika pengguna tidak memiliki peran IAM yang ditentukan dengan hak istimewa untuk menjalankan perintah yang ditentukan. Untuk informasi selengkapnya tentang hak istimewa, lihatHIBAH.

Sintaksis

```
has_assumerole_privilege( [ user, ] iam_role_arn, cmd_type)
```

Argumen

pengguna

Nama pengguna untuk memeriksa hak istimewa peran IAM. Defaultnya adalah memeriksa pengguna saat ini. Pengguna super dan pengguna dapat menggunakan fungsi ini. Namun, pengguna hanya dapat melihat hak istimewa mereka sendiri.

iam_role_arn

Peran IAM yang telah diberikan hak istimewa perintah.

cmd_tipe

Perintah yang aksesnya telah diberikan. Nilai yang valid adalah sebagai berikut:

- MENYALIN
- MEMBONGKAR
- FUNGSI EKSTERNAL
- BUAT MODEL

Jenis pengembalian

BOOLEAN

Contoh

Kueri berikut mengonfirmasi bahwa pengguna `reg_user1` memiliki hak istimewa untuk `Redshift-S3-Read` peran untuk menjalankan perintah COPY.

```
select has_assumerole_privilege('reg_user1', 'arn:aws:iam::123456789012:role/Redshift-S3-Read', 'copy');
```

```
has_assumerole_privilege
------------------------
true
(1 row)
```

MEMILIKI HAK ISTIMEWA BASIS DATA

Meng `true` embalikan jika pengguna memiliki hak istimewa yang ditentukan untuk database yang ditentukan. Untuk informasi selengkapnya tentang hak istimewa, lihatHIBAH.

Sintaksis

> **Note**
>
> Ini adalah fungsi leader-node. Fungsi ini mengembalikan kesalahan jika mereferensikan tabel yang dibuat pengguna, tabel sistem STL atau STV, atau tampilan sistem SVV atau SVL.

```
has_database_privilege( [ user, ] database, privilege)
```

Argumen

pengguna

Nama pengguna untuk memeriksa hak istimewa database. Defaultnya adalah memeriksa pengguna saat ini.

basis data

Database yang terkait dengan hak istimewa.

hak istimewa

Hak istimewa untuk memeriksa. Nilai yang valid adalah sebagai berikut:

- CREATE
- SEMENTARA
- TEMP

Jenis pengembalian

Mengembalikan string CHAR atau VARCHAR.

Contoh

Kueri berikut mengonfirmasi bahwa pengguna GUEST memiliki hak istimewa TEMP pada database TICKIT.

```
select has_database_privilege('guest', 'tickit', 'temp');

has_database_privilege
-----------------------
```

```
true
(1 row)
```

MEMILIKI HAK ISTIMEWA SKEMA

Mengembalikan `true` jika pengguna memiliki hak istimewa yang ditentukan untuk skema yang ditentukan. Untuk informasi selengkapnya tentang hak istimewa, lihatHIBAH.

Sintaksis

> **Note**
>
> Ini adalah fungsi leader-node. Fungsi ini mengembalikan kesalahan jika mereferensikan tabel yang dibuat pengguna, tabel sistem STL atau STV, atau tampilan sistem SVV atau SVL.

```
has_schema_privilege( [ user, ] schema, privilege)
```

Argumen

pengguna

Nama pengguna untuk memeriksa hak istimewa skema. Defaultnya adalah memeriksa pengguna saat ini.

skema

Skema yang terkait dengan hak istimewa.

hak istimewa

Hak istimewa untuk memeriksa. Nilai yang valid adalah sebagai berikut:

- CREATE
- PENGGUNAAN
- MENGUBAH
- MENJATUHKAN

Jenis pengembalian

Mengembalikan string CHAR atau VARCHAR.

Contoh

Kueri berikut mengonfirmasi bahwa pengguna GUEST memiliki hak istimewa CREATE pada skema PUBLIK:

```
select has_schema_privilege('guest', 'public', 'create');

has_schema_privilege
----------------------
true
(1 row)
```

MEMILIKI HAK ISTIMEWA TABEL_

Meng `true` embalikan jika pengguna memiliki hak istimewa yang ditentukan untuk tabel yang ditentukan dan mengembalikan `false` sebaliknya.

Sintaksis

> **Note**
>
> Ini adalah fungsi leader-node. Fungsi ini mengembalikan kesalahan jika mereferensikan tabel yang dibuat pengguna, tabel sistem STL atau STV, atau tampilan sistem SVV atau SVL. Untuk informasi selengkapnya tentang hak istimewa, lihatHIBAH.

```
has_table_privilege( [ user, ] table, privilege)
```

Argumen

pengguna

Nama pengguna untuk memeriksa hak istimewa tabel. Defaultnya adalah memeriksa pengguna saat ini.

tabel

Tabel terkait dengan hak istimewa.

hak istimewa

Hak istimewa untuk memeriksa. Nilai yang valid adalah sebagai berikut:

- SELECT
- INSERT
- UPDATE
- DELETE
- MENJATUHKAN
- REFERENSI

Jenis pengembalian

BOOLEAN

Contoh

Kueri berikut menemukan bahwa pengguna TAMU tidak memiliki hak istimewa SELECT pada tabel LISTING.

```
select has_table_privilege('guest', 'listing', 'select');

has_table_privilege
---------------------
false
```

Kueri berikut mencantumkan hak istimewa tabel, termasuk memilih, menyisipkan, memperbarui, dan menghapus, menggunakan output dari tabel katalog pg_tables dan pg_user. Ini adalah sampel saja. Anda mungkin harus menentukan nama skema dan nama tabel dari database Anda. Untuk informasi selengkapnya, lihat Menanyakan tabel katalog.

```
SELECT
     tablename
     ,usename
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'select') AS sel
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'insert') AS ins
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'update') AS upd
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'delete') AS del
FROM
(SELECT * from pg_tables
WHERE schemaname = 'public' and tablename in ('event','listing')) as tables
,(SELECT * FROM pg_user) AS users;

tablename | usename   |  sel   |  ins  |  upd  | del
```

```
----------+-----------+--------+-------+-------+-------
event     |  john     |  true  | true  | true  | true
event     |  sally    |  false | false | false | false
event     |  elsa     |  false | false | false | false
listing   |  john     |  true  | true  | true  | true
listing   |  sally    |  false | false | false | false
listing   |  elsa     |  false | false | false | false
```

Kueri sebelumnya juga berisi cross join. Untuk informasi selengkapnya, lihat Contoh JOIN. Untuk menanyakan tabel yang tidak ada dalam `public` skema, hapus `schemaname` kondisi dari klausa WHERE dan gunakan contoh berikut sebelum kueri Anda.

```
SET SEARCH_PATH to 'schema_name';
```

ID KUERI PENGGUNA TERAKHIR

Mengembalikan ID kueri dari kueri pengguna yang terakhir diselesaikan dalam sesi saat ini. Jika tidak ada kueri yang dijalankan di sesi saat ini, last_user_query_id mengembalikan -1. Fungsi tidak mengembalikan ID kueri untuk kueri yang berjalan secara eksklusif pada node pemimpin. Untuk informasi selengkapnya, lihat Node—hanya fungsi utama.

Sintaks

```
last_user_query_id()
```

Jenis pengembalian

Mengembalikan bilangan bulat.

Contoh

Kueri berikut mengembalikan ID kueri terbaru yang dijalankan oleh pengguna yang diselesaikan dalam sesi saat ini.

```
select last_user_query_id();
```

Hasilnya adalah sebagai berikut.

```
last_user_query_id
----------------------
    5437
```

```
(1 row)
```

Kueri berikut mengembalikan ID kueri dan teks dari kueri yang terakhir diselesaikan yang dijalankan oleh pengguna di sesi saat ini.

```
select query_id, query_text from sys_query_history where query_id =
 last_user_query_id();
```

Hasilnya adalah sebagai berikut.

```
 query_id, query_text
---------
+----------------------------------------------------------------------------------------------------
 5556975 | select last_user_query_id() limit 100 --RequestID=<unique request ID>;
 TraceID=<unique trace ID>
```

PG_BACKEND_PID

Mengembalikan ID proses (PID) dari proses server yang menangani sesi saat ini.

> **Note**
>
> PID tidak unik secara global. Hal ini dapat digunakan kembali dari waktu ke waktu.

Sintaksis

```
pg_backend_pid()
```

Jenis pengembalian

Mengembalikan bilangan bulat.

Contoh

Anda dapat menghubungkan PG_BACKEND_PID dengan tabel log untuk mengambil informasi untuk sesi saat ini. Misalnya, kueri berikut mengembalikan ID kueri dan sebagian teks kueri untuk kueri yang diselesaikan dalam sesi saat ini.

```
select query, substring(text,1,40)
```

```
from stl_querytext
where pid =  PG_BACKEND_PID()
order by query desc;

 query |                substring
-------+-----------------------------------------
 14831 | select query, substring(text,1,40) from
 14827 | select query, substring(path,0,80) as pa
 14826 | copy category from 's3://dw-tickit/manif
 14825 | Count rows in target table
 14824 | unload ('select * from category') to 's3
(5 rows)
```

Anda dapat menghubungkan PG_BACKEND_PID dengan kolom pid dalam tabel log berikut (pengecualian dicatat dalam tanda kurung):

- STL_CONNECTION_LOG
- STL_DDLTEXT
- STL_ERROR
- STL_QUERY
- STL_QUERYTEXT
- STL_SESSIONS(proses)
- STL_TR_KONFLIK
- STL_UTILITYTEXT
- STV_ACTIVE_KURSOR
- STV_INFLIGHT
- STV_LOCKS(lock_owner_pid)
- STV_TERBARU(id_proses)

PG_GET_COLS

Mengembalikan metadata kolom untuk tabel atau definisi tampilan.

Sintaksis

```
pg_get_cols('name')
```

Argumen

name

Nama tabel atau tampilan Amazon Redshift. Untuk informasi selengkapnya, lihat Nama dan pengidentifikasi.

Jenis pengembalian

VARCHAR

Catatan penggunaan

Fungsi PG_GET_COLS mengembalikan satu baris untuk setiap kolom dalam tabel atau definisi tampilan. Baris berisi daftar yang dipisahkan koma dengan nama skema, nama relasi, nama kolom, tipe data, dan nomor kolom. Pemformatan hasil SQL tergantung pada klien SQL yang digunakan.

Contoh

Contoh berikut mengembalikan hasil untuk tampilan yang dinam `SALES_VW` ai dalam skema `public` dan tabel tabel bernama `sales` dalam skema `mytickit1` yang dibuat oleh pengguna dalam database yang terhubung`dev`.

Contoh berikut mengembalikan metadata kolom untuk tampilan bernama`SALES_VW`.

```
select pg_get_cols('sales_vw');

pg_get_cols
---------------------------------------------------------------
(public,sales_vw,salesid,integer,1)
(public,sales_vw,listid,integer,2)
(public,sales_vw,sellerid,integer,3)
(public,sales_vw,buyerid,integer,4)
(public,sales_vw,eventid,integer,5)
(public,sales_vw,dateid,smallint,6)
(public,sales_vw,qtysold,smallint,7)
(public,sales_vw,pricepaid,"numeric(8,2)",8)
(public,sales_vw,commission,"numeric(8,2)",9)
(public,sales_vw,saletime,"timestamp without time zone",10)
```

Contoh berikut mengembalikan metadata kolom untuk `SALES_VW` tampilan dalam format tabel.

```
select * from pg_get_cols('sales_vw')
cols(view_schema name, view_name name, col_name name, col_type varchar, col_num int);

view_schema | view_name | col_name   | col_type                    | col_num
------------+-----------+------------+-----------------------------+--------
public      | sales_vw  | salesid    | integer                     |       1
public      | sales_vw  | listid     | integer                     |       2
public      | sales_vw  | sellerid   | integer                     |       3
public      | sales_vw  | buyerid    | integer                     |       4
public      | sales_vw  | eventid    | integer                     |       5
public      | sales_vw  | dateid     | smallint                    |       6
public      | sales_vw  | qtysold    | smallint                    |       7
public      | sales_vw  | pricepaid  | numeric(8,2)                |       8
public      | sales_vw  | commission | numeric(8,2)                |       9
public      | sales_vw  | saletime   | timestamp without time zone |      10
```

Contoh berikut mengembalikan metadata kolom untuk SALES tabel dalam skema `mytickit1` dalam format tabel.

```
select * from pg_get_cols('"mytickit1"."sales"')
cols(view_schema name, view_name name, col_name name, col_type varchar, col_num int);

view_schema | view_name | col_name   | col_type                    | col_num
------------+-----------+------------+-----------------------------+--------
mytickit1   | sales     | salesid    | integer                     |       1
mytickit1   | sales     | listid     | integer                     |       2
mytickit1   | sales     | sellerid   | integer                     |       3
mytickit1   | sales     | buyerid    | integer                     |       4
mytickit1   | sales     | eventid    | integer                     |       5
mytickit1   | sales     | dateid     | smallint                    |       6
mytickit1   | sales     | qtysold    | smallint                    |       7
mytickit1   | sales     | pricepaid  | numeric(8,2)                |       8
mytickit1   | sales     | commission | numeric(8,2)                |       9
mytickit1   | sales     | saletime   | timestamp without time zone |      10
```

PG_DAPAT_PENERIMA HIBAH OLEH_IAM_ROLE

Mengembalikan semua pengguna dan grup yang diberikan peran IAM tertentu.

Sintaksis

```
pg_get_grantee_by_iam_role('iam_role_arn')
```

Argumen

iam_role_arn

Peran IAM untuk mengembalikan pengguna dan grup yang telah diberikan peran ini.

Jenis pengembalian

VARCHAR

Catatan penggunaan

Fungsi PG_GET_GRANTEE_BY_IAM_ROLE mengembalikan satu baris untuk setiap pengguna atau grup. Setiap baris berisi nama penerima hibah, jenis penerima hibah, dan hak istimewa yang diberikan. Nilai yang mungkin untuk jenis penerima hibah adalah p untuk publik, u untuk pengguna, dan g untuk grup.

Anda harus menjadi superuser untuk menggunakan fungsi ini.

Contoh

Contoh berikut menunjukkan bahwa peran IAM `Redshift-S3-Write` diberikan kepada `group1` dan`reg_user1`. Pengguna `group_1` dapat menentukan peran hanya untuk operasi COPY, dan pengguna `reg_user1` dapat menentukan peran hanya untuk melakukan operasi UNLOAD.

```
select pg_get_grantee_by_iam_role('arn:aws:iam::123456789012:role/Redshift-S3-Write');
```

```
  pg_get_grantee_by_iam_role
-----------------------------
 (group_1,g,COPY)
 (reg_user1,u,UNLOAD)
```

Contoh berikut dari fungsi PG_GET_GRANTEE_BY_IAM_ROLE memformat hasilnya sebagai tabel.

```
select grantee, grantee_type, cmd_type FROM
 pg_get_grantee_by_iam_role('arn:aws:iam::123456789012:role/Redshift-S3-Write')
 res_grantee(grantee text, grantee_type text, cmd_type text) ORDER BY 1,2,3;
```

```
  grantee  | grantee_type | cmd_type
```

```
-----------+--------------+----------
 group_1   | g            | COPY
 reg_user1 | u            | UNLOAD
```

PG_GET_IAM_ROL_OLE_OLE_PENGGUNA

Mengembalikan semua peran IAM dan hak perintah yang diberikan kepada pengguna.

Sintaksis

```
pg_get_iam_role_by_user('name')
```

Argumen

name

Nama pengguna yang akan mengembalikan peran IAM.

Jenis pengembalian

VARCHAR

Catatan penggunaan

Fungsi PG_GET_IAM_ROLE_BY_USER mengembalikan satu baris untuk setiap set peran dan hak istimewa perintah. Baris berisi daftar yang dipisahkan koma dengan nama pengguna, peran IAM, dan perintah.

Nilai `default` dalam hasil menunjukkan bahwa pengguna dapat menentukan peran yang tersedia untuk melakukan perintah yang ditampilkan.

Anda harus menjadi superuser untuk menggunakan fungsi ini.

Contoh

Contoh berikut menunjukkan bahwa pengguna `reg_user1` dapat menentukan peran IAM yang tersedia untuk melakukan operasi COPY. Pengguna juga dapat menentukan `Redshift-S3-Write` peran untuk operasi UNLOAD.

```
select pg_get_iam_role_by_user('reg_user1');
```

```
                            pg_get_iam_role_by_user
------------------------------------------------------------------------------
 (reg_user1,default,COPY)
 (reg_user1,arn:aws:iam::123456789012:role/Redshift-S3-Write,COPY|UNLOAD)
```

Contoh berikut dari fungsi PG_GET_IAM_ROLE_BY_USER memformat hasilnya sebagai tabel.

```
select username, iam_role, cmd FROM pg_get_iam_role_by_user('reg_user1')
 res_iam_role(username text, iam_role text, cmd text);
```

```
 username  |                    iam_role                     | cmd
-----------+-------------------------------------------------+------
 reg_user1 | default                                         | None
 reg_user1 | arn:aws:iam::123456789012:role/Redshift-S3-Read | COPY
```

PG_GET_LATE_BINDING_VIEW_COLS

Mengembalikan metadata kolom untuk semua tampilan pengikatan terlambat dalam database. Untuk informasi selengkapnya, lihat Late-binding pandangan

Sintaksis

```
pg_get_late_binding_view_cols()
```

Jenis pengembalian

VARCHAR

Catatan penggunaan

`PG_GET_LATE_BINDING_VIEW_COLS`Fungsi mengembalikan satu baris untuk setiap kolom dalam tampilan pengikatan terlambat. Baris berisi daftar yang dipisahkan koma dengan nama skema, nama relasi, nama kolom, tipe data, dan nomor kolom.

Contoh

Contoh berikut mengembalikan metadata kolom untuk semua tampilan pengikatan terlambat.

```
select pg_get_late_binding_view_cols();
```

```
pg_get_late_binding_view_cols
------------------------------------------------------------
(public,myevent,eventname,"character varying(200)",1)
(public,sales_lbv,salesid,integer,1)
(public,sales_lbv,listid,integer,2)
(public,sales_lbv,sellerid,integer,3)
(public,sales_lbv,buyerid,integer,4)
(public,sales_lbv,eventid,integer,5)
(public,sales_lbv,dateid,smallint,6)
(public,sales_lbv,qtysold,smallint,7)
(public,sales_lbv,pricepaid,"numeric(8,2)",8)
(public,sales_lbv,commission,"numeric(8,2)",9)
(public,sales_lbv,saletime,"timestamp without time zone",10)
(public,event_lbv,eventid,integer,1)
(public,event_lbv,venueid,smallint,2)
(public,event_lbv,catid,smallint,3)
(public,event_lbv,dateid,smallint,4)
(public,event_lbv,eventname,"character varying(200)",5)
(public,event_lbv,starttime,"timestamp without time zone",6)
```

Contoh berikut mengembalikan metadata kolom untuk semua tampilan pengikatan terlambat dalam format tabel.

```
select * from pg_get_late_binding_view_cols() cols(view_schema name, view_name name,
 col_name name, col_type varchar, col_num int);
view_schema | view_name | col_name   | col_type                    | col_num
------------+-----------+------------+-----------------------------+--------
public      | sales_lbv | salesid    | integer                     |       1
public      | sales_lbv | listid     | integer                     |       2
public      | sales_lbv | sellerid   | integer                     |       3
public      | sales_lbv | buyerid    | integer                     |       4
public      | sales_lbv | eventid    | integer                     |       5
public      | sales_lbv | dateid     | smallint                    |       6
public      | sales_lbv | qtysold    | smallint                    |       7
public      | sales_lbv | pricepaid  | numeric(8,2)                |       8
public      | sales_lbv | commission | numeric(8,2)                |       9
public      | sales_lbv | saletime   | timestamp without time zone |      10
public      | event_lbv | eventid    | integer                     |       1
public      | event_lbv | venueid    | smallint                    |       2
public      | event_lbv | catid      | smallint                    |       3
public      | event_lbv | dateid     | smallint                    |       4
public      | event_lbv | eventname  | character varying(200)      |       5
public      | event_lbv | starttime  | timestamp without time zone |       6
```

PG_GET_SESSION_ROLE

Mengembalikan peran sesi dari pengguna yang saat ini masuk. Peran sesi pengguna adalah grup yang ditentukan oleh penyedia identitas (IdP) untuk pengguna yang masuk. Misalnya, penyedia identitas (IdP) seperti Microsoft Azure Active Directory (Azure AD) memverifikasi identitas pengguna dan menyediakan grup eksternal yang menjadi bagiannya pengguna selama proses login pengguna. Grup eksternal ini diubah menjadi peran Amazon Redshift dan tersedia selama sesi saat ini. Peran-peran ini disebut peran sesi. Administrator dapat memberikan hak istimewa ke peran sesi yang mirip dengan peran Amazon Redshift lainnya. Untuk informasi tentang menggunakan peran, lihatRole-based kontrol akses (RBAC). Untuk informasi tentang mengelola identitas dengan penyedia identitas (IdP), lihat Federasi penyedia identitas asli (IdP) untuk Amazon Redshift di Panduan Manajemen Amazon Redshift.

Untuk melihat peran yang ditentukan dalam katalog Amazon Redshift, kueri tampilan SVV_ROLE sistem.

Sintaksis

```
pg_get_session_roles()
```

Jenis pengembalian

Satu set baris yang terdiri dari dua nilai. Nilai pertama memiliki dua bagian yang dipisahkan oleh titik dua (:) yang berisi`idp-namespace:role-name`. Ini `idp-namespace` adalah namespace penyedia identitas (IdP). Ini `role-name` adalah nama grup eksternal di penyedia identitas (IdP). Nilai kedua berisi a `role-id` yang merupakan pengenal peran.

Catatan penggunaan

`PG_GET_SESSION_ROLES`Fungsi mengembalikan satu baris untuk setiap peran sesi yang dikembalikan.

Contoh

Contoh berikut mengembalikan satu baris untuk setiap peran dari Azure Active Directory IdP. Kolom yang dikembalikan dicetak `sess_roles` dengan kolom `name` dan`roleid`. Masing-masing `name` terdiri dari namespace Azure Active Directory dan nama grup di Azure Active Directory.

```
SELECT * FROM pg_get_session_roles() AS sess_roles(name name, roleid integer);
```

```
name                   roleid
--------------------------------
my_aad:test_group_1    106204
my_aad:test_group_2    106205
my_aad:test_group_3    106206
my_aad:test_group_4    106207
my_aad:test_group_5    106208
```

Contoh berikut mengembalikan satu baris untuk setiap grup IAM yang merupakan anggota pengguna IAM yang saat ini masuk. Kolom yang dikembalikan dicetak `sess_roles` dengan kolom `name` dan`roleid`. Masing-masing `name` terdiri dari namespace IAM dan nama grup IAM.

```
SELECT * FROM pg_get_session_roles() AS sess_roles(name name, roleid integer);

name                   roleid
--------------------------------
IAM:myGroup            110332
```

PG_LAST_COPY_COUNT

Mengembalikan jumlah baris yang dimuat oleh perintah COPY terakhir yang dijalankan dalam sesi saat ini. PG_LAST_COPY_COUNT diperbarui dengan COPY ID terakhir, yang merupakan ID kueri dari COPY terakhir yang memulai proses pemuatan, bahkan jika pemuatan gagal. ID kueri dan COPY ID diperbarui saat perintah COPY memulai proses pemuatan.

Jika COPY gagal karena kesalahan sintaks atau karena hak istimewa yang tidak mencukupi, COPY ID tidak diperbarui dan PG_LAST_COPY_COUNT mengembalikan hitungan untuk COPY sebelumnya. Jika tidak ada perintah COPY yang dijalankan di sesi saat ini, atau jika COPY terakhir gagal selama pemuatan, PG_LAST_COPY_COUNT mengembalikan 0. Untuk informasi selengkapnya, lihat PG_LAST_COPY_ID.

Sintaks

```
pg_last_copy_count()
```

Jenis pengembalian

Mengembalikan BIGINT.

Contoh

Kueri berikut mengembalikan jumlah baris yang dimuat oleh perintah COPY terbaru dalam sesi saat ini.

```
select pg_last_copy_count();

pg_last_copy_count
--------------------
             192497
(1 row)
```

PG_LAST_COPY_ID

Mengembalikan ID kueri dari perintah COPY yang terakhir diselesaikan di sesi saat ini. Jika tidak ada perintah COPY yang dijalankan di sesi saat ini, PG_LAST_COPY_ID mengembalikan -1.

Nilai untuk PG_LAST_COPY_ID diperbarui saat perintah COPY memulai proses pemuatan. Jika COPY gagal karena data pemuatan tidak valid, COPY ID diperbarui, sehingga Anda dapat menggunakan PG_LAST_COPY_ID saat menanyakan tabel STL_LOAD_ERRORS. Jika transaksi COPY dibatalkan, COPY ID tidak diperbarui.

COPY ID tidak diperbarui jika perintah COPY gagal karena kesalahan yang terjadi sebelum proses pemuatan dimulai, seperti kesalahan sintaks, kesalahan akses, kredentif tidak valid, atau hak istimewa yang tidak mencukupi. COPY ID tidak diperbarui jika COPY gagal selama langkah kompresi analisis, yang dimulai setelah koneksi berhasil, tetapi sebelum pemuatan data.

Sintaksis

```
pg_last_copy_id()
```

Jenis pengembalian

Mengembalikan bilangan bulat.

Contoh

Kueri berikut mengembalikan ID kueri dari perintah COPY terbaru dalam sesi saat ini.

```
select pg_last_copy_id();
```

```
pg_last_copy_id
---------------
          5437
(1 row)
```

Kueri berikut menggabungkan STL_LOAD_ERRORS ke STL_LOADERROR_DETAIL untuk melihat detail error yang terjadi selama pemuatan terbaru di sesi saat ini:

```
select d.query, substring(d.filename,14,20),
d.line_number as line,
substring(d.value,1,16) as value,
substring(le.err_reason,1,48) as err_reason
from stl_loaderror_detail d, stl_load_errors le
where d.query = le.query
and d.query = pg_last_copy_id();

 query |    substring      | line |  value   |                    err_reason
-------+-------------------+------+----------
+-----------------------------------------------
    558| allusers_pipe.txt |  251 | 251      | String contains invalid or unsupported
 UTF8 code
    558| allusers_pipe.txt |  251 | ZRU29FGR | String contains invalid or unsupported
 UTF8 code
    558| allusers_pipe.txt |  251 | Kaitlin  | String contains invalid or unsupported
 UTF8 code
    558| allusers_pipe.txt |  251 | Walter   | String contains invalid or unsupported
 UTF8 code
```

PG_LAST_UNLOAD_ID

Mengembalikan ID kueri dari perintah UNLOAD yang terakhir diselesaikan di sesi saat ini. Jika tidak ada perintah UNLOAD yang dijalankan di sesi saat ini, PG_LAST_UNLOAD_ID mengembalikan -1.

Nilai untuk PG_LAST_UNLOAD_ID diperbarui saat perintah UNLOAD memulai proses pemuatan. Jika UNLOAD gagal karena data pemuatan tidak valid, ID UNLOAD diperbarui, sehingga Anda dapat menggunakan ID UNLOAD untuk penyelidikan lebih lanjut. Jika transaksi UNLOAD dibatalkan, ID UNLOAD tidak diperbarui.

ID UNLOAD tidak diperbarui jika perintah UNLOAD gagal karena kesalahan yang terjadi sebelum proses pemuatan dimulai, seperti kesalahan sintaks, kesalahan akses, kredentif tidak valid, atau hak istimewa yang tidak mencukupi.

Sintaksis

```
PG_LAST_UNLOAD_ID()
```

Jenis pengembalian

Mengembalikan bilangan bulat.

Contoh

Kueri berikut mengembalikan ID kueri dari perintah UNLOAD terbaru dalam sesi saat ini.

```
select PG_LAST_UNLOAD_ID();

PG_LAST_UNLOAD_ID
---------------
          5437
(1 row)
```

PG_LAST_QUERY_ID

Mengembalikan ID kueri dari kueri yang terakhir diselesaikan dalam sesi saat ini. Jika tidak ada kueri yang dijalankan di sesi saat ini, PG_LAST_QUERY_ID mengembalikan -1. PG_LAST_QUERY_ID tidak mengembalikan ID kueri untuk kueri yang berjalan secara eksklusif pada node pemimpin. Untuk informasi selengkapnya, lihat Node—hanya fungsi utama.

Sintaks

```
pg_last_query_id()
```

Jenis pengembalian

Mengembalikan bilangan bulat.

Contoh

Kueri berikut mengembalikan ID kueri terbaru yang diselesaikan dalam sesi saat ini.

```
select pg_last_query_id();
```

Hasilnya adalah sebagai berikut.

```
pg_last_query_id
----------------
           5437
(1 row)
```

Kueri berikut mengembalikan ID kueri dan teks dari kueri yang terakhir diselesaikan dalam sesi saat ini.

```
select query, trim(querytxt) as sqlquery
from stl_query
where query = pg_last_query_id();
```

Hasilnya adalah sebagai berikut.

```
query | sqlquery
------+-------------------------------------------------
 5437 | select name, loadtime from stl_file_scan where loadtime > 1000000;
(1 rows)
```

PG_LAST_UNLOAD_COUNT

Mengembalikan jumlah baris yang dibongkar oleh perintah UNLOAD terakhir yang diselesaikan dalam sesi saat ini. PG_LAST_UNLOAD_COUNT diperbarui dengan ID kueri dari UNLOAD terakhir, bahkan jika operasi gagal. ID kueri diperbarui saat UNLOAD selesai. Jika UNLOAD gagal karena kesalahan sintaks atau karena hak istimewa yang tidak mencukupi, PG_LAST_UNLOAD_COUNT mengembalikan hitungan untuk UNLOAD sebelumnya. Jika tidak ada perintah UNLOAD yang diselesaikan dalam sesi saat ini, atau jika UNLOAD terakhir gagal selama operasi pembongkaran, PG_LAST_UNLOAD_COUNT mengembalikan 0.

Sintaksis

```
pg_last_unload_count()
```

Jenis pengembalian

Mengembalikan BIGINT.

Contoh

Kueri berikut mengembalikan jumlah baris yang dibongkar oleh perintah UNLOAD terbaru dalam sesi saat ini.

```
select pg_last_unload_count();

pg_last_unload_count
--------------------
             192497
(1 row)
```

Fungsi SLICE_NUM

Mengembalikan bilangan bulat yang sesuai dengan nomor irisan di cluster tempat data untuk baris berada. SLICE_NUM tidak mengambil parameter.

Sintaksis

```
SLICE_NUM()
```

Jenis pengembalian

Fungsi SLICE_NUM mengembalikan bilangan bulat.

Contoh

Contoh berikut menunjukkan irisan mana yang berisi data untuk sepuluh baris EVENT pertama dalam tabel EVENTS:

```
select distinct eventid, slice_num() from event order by eventid limit 10;

 eventid | slice_num
---------+-----------
       1 |         1
       2 |         2
       3 |         3
       4 |         0
       5 |         1
       6 |         2
       7 |         3
       8 |         0
       9 |         1
      10 |         2
(10 rows)
```

Contoh berikut mengembalikan kode (10000) untuk menunjukkan bahwa kueri tanpa pernyataan FROM berjalan pada node pemimpin:

```
select slice_num();
slice_num
-----------
10000
(1 row)
```

USER

Sinonim untuk CURRENT_USER. Lihat CURRENT_USER.

ROL_IS_ANGGOTA_DARI

Mengembalikan nilai true jika peran tersebut adalah anggota dari peran lain. Superuser dapat memeriksa keanggotaan semua peran. Pengguna reguler yang memiliki izin TABEL SISTEM AKSES dapat memeriksa semua keanggotaan pengguna. Jika tidak, pengguna biasa hanya dapat memeriksa peran yang mereka akses. Amazon Redshift mengalami kesalahan jika peran yang disediakan tidak ada atau pengguna saat ini tidak memiliki akses ke peran tersebut.

Pertimbangan berbagi data

Ketika cluster konsumen menanyakan objek bersama yang mereferensikan fungsi ini, seperti tampilan, kebijakan RLS, atau kebijakan DDM, fungsi mengevaluasi menggunakan konteks keamanan cluster konsumen. Pengguna lokal konsumen, peran, dan keanggotaan grup menentukan hasilnya, bukan yang ditentukan pada cluster produsen. Jika Anda ingin menerapkan konteks izin yang sama yang diterapkan pada produsen, pastikan bahwa nama peran, nama grup, dan keanggotaan pengguna yang sesuai ada di cluster konsumen dan cocok dengan nama yang ada di produsen.

Sintaksis

```
role_is_member_of( role_name,  granted_role_name)
```

Argumen

role_name

Nama peran.

nama_peran yang diberikan

Nama peran yang diberikan.

Jenis pengembalian

Mengembalikan BOOLEAN.

Contoh

Kueri berikut mengonfirmasi bahwa peran tersebut bukan anggota role1 atau role2.

```
SELECT role_is_member_of('role1', 'role2');

 role_is_member_of
-------------------
             False
```

PENGGUNA_ADALAH_ANGGOTA_DARI

> **Important**
>
> Mulai 16 Februari 2026, Amazon Redshift tidak akan lagi mendukung penggunaan `user_is_member_of` dan fungsi terkait yang mengakses pengguna konsumen, peran, atau informasi grup melalui pembagian data.

Mengembalikan true jika pengguna adalah anggota peran atau grup. Superuser dapat memeriksa keanggotaan semua pengguna. Pengguna biasa yang merupakan anggota peran sys:secadmin atau sys:superuser dapat memeriksa semua keanggotaan pengguna. Jika tidak, pengguna biasa hanya dapat memeriksa diri mereka sendiri. Amazon Redshift mengirimkan kesalahan jika identitas yang disediakan tidak ada atau pengguna saat ini tidak memiliki akses ke peran tersebut.

Pertimbangan berbagi data

Ketika cluster konsumen menanyakan objek bersama yang mereferensikan fungsi ini, seperti tampilan, kebijakan RLS, atau kebijakan DDM, fungsi mengevaluasi menggunakan konteks keamanan cluster konsumen. Pengguna lokal konsumen, peran, dan keanggotaan grup menentukan hasilnya, bukan yang ditentukan pada cluster produsen. Jika Anda ingin menerapkan konteks

izin yang sama yang diterapkan pada produsen, pastikan bahwa nama peran, nama grup, dan keanggotaan pengguna yang sesuai ada di cluster konsumen dan cocok dengan nama yang ada di produsen.

Sintaksis

```
user_is_member_of( user_name,  role_name | group_name)
```

Argumen

nama_pengguna

Nama pengguna.

role_name

Nama peran.

group_name

Nama grup.

Jenis pengembalian

Mengembalikan BOOLEAN.

Contoh

Kueri berikut mengonfirmasi bahwa pengguna bukan anggota role1.

```
SELECT user_is_member_of('reguser', 'role1');

 user_is_member_of
-------------------
           False
```

VERSI

Fungsi VERSION mengembalikan detail tentang rilis yang diinstal saat ini, dengan informasi versi Amazon Redshift tertentu di bagian akhir.

> **Note**
>
> Ini adalah fungsi leader-node. Fungsi ini mengembalikan kesalahan jika mereferensikan tabel yang dibuat pengguna, tabel sistem STL atau STV, atau tampilan sistem SVV atau SVL.

Sintaksis

```
VERSION()
```

Jenis pengembalian

Mengembalikan string CHAR atau VARCHAR.

Contoh

Contoh berikut menunjukkan informasi versi cluster dari cluster saat ini:

```
select version();
```

```
 version
-----------------------------------------------------------------------------------------------------------

PostgreSQL 8.0.2 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 3.4.2 20041017 (Red
Hat 3.4.2-6.fc3), Redshift 1.0.12103
```

Di `1.0.12103` mana nomor versi cluster.

> **Note**
>
> Untuk memaksa cluster Anda memperbarui ke versi cluster terbaru, sesuaikan jendela pemeliharaan Anda.

Kata yang dicadangkan

Berikut ini adalah daftar kata-kata yang dicadangkan Amazon Redshift. Anda dapat menggunakan kata-kata yang dicadangkan dengan pengidentifikasi yang dibatasi (tanda kutip ganda).

> **Note**
>
> Meskipun START dan CONNECT bukan kata yang dicadangkan, gunakan pengidentifikasi yang dibatasi atau AS jika Anda menggunakan START dan CONNECT sebagai alias tabel dalam kueri Anda untuk menghindari kegagalan saat runtime.

Lihat informasi yang lebih lengkap di Nama dan pengidentifikasi.

```
AES128
AES256
ALL
ALLOWOVERWRITE
ANALYSE
ANALYZE
AND
ANY
ARRAY
AS
ASC
AUTHORIZATION
AZ64
BACKUP
BETWEEN
BINARY
BLANKSASNULL
BOTH
BYTEDICT
BZIP2
CASE
CAST
CHECK
COLLATE
COLUMN
CONSTRAINT
CREATE
CREDENTIALS
CROSS
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
```

```
CURRENT_USER_ID
DEFAULT
DEFERRABLE
DEFLATE
DEFRAG
DELTA
DELTA32K
DESC
DISABLE
DISTINCT
DO
ELSE
EMPTYASNULL
ENABLE
ENCODE
ENCRYPT
ENCRYPTION
END
EXCEPT
EXPLICIT
FALSE
FOR
FOREIGN
FREEZE
FROM
FULL
GLOBALDICT256
GLOBALDICT64K
GRANT
GROUP
GZIP
HAVING
IDENTITY
IGNORE
ILIKE
IN
INITIALLY
INNER
INTERSECT
INTERVAL
INTO
IS
ISNULL
JOIN
```

```
LEADING
LEFT
LIKE
LIMIT
LOCALTIME
LOCALTIMESTAMP
LUN
LUNS
LZO
LZOP
MINUS
MOSTLY16
MOSTLY32
MOSTLY8
NATURAL
NEW
NOT
NOTNULL
NULL
NULLS
OFF
OFFLINE
OFFSET
OID
OLD
ON
ONLY
OPEN
OR
ORDER
OUTER
OVERLAPS
PARALLEL
PARTITION
PERCENT
PERMISSIONS
PIVOT
PLACING
PRIMARY
RAW
READRATIO
RECOVER
REFERENCES
REJECTLOG
```

```
RESORT
RESPECT
RESTORE
RIGHT
SELECT
SESSION_USER
SIMILAR
SNAPSHOT
SOME
SYSDATE
SYSTEM
TABLE
TAG
TDES
TEXT255
TEXT32K
THEN
TIMESTAMP
TO
TOP
TRAILING
TRUE
TRUNCATECOLUMNS
UNION
UNIQUE
UNNEST
UNPIVOT
USER
USING
VERBOSE
WALLET
WHEN
WHERE
WITH
WITHOUT
```

Tabel sistem dan referensi tampilan

Amazon Redshift memiliki banyak tabel dan tampilan sistem yang berisi informasi tentang bagaimana sistem berfungsi. Anda dapat menanyakan tabel dan tampilan sistem ini dengan cara yang sama seperti Anda akan menanyakan tabel database lainnya. Bagian ini menunjukkan beberapa contoh query tabel sistem dan menjelaskan:

- Bagaimana berbagai jenis tabel dan tampilan sistem dihasilkan
- Jenis informasi apa yang dapat Anda peroleh dari tabel ini
- Cara menggabungkan tabel sistem Amazon Redshift ke tabel katalog
- Cara mengelola pertumbuhan file log tabel sistem

Beberapa tabel sistem hanya dapat digunakan oleh AWS staf untuk tujuan diagnostik. Bagian berikut membahas tabel sistem yang dapat ditanyakan untuk informasi yang berguna oleh administrator sistem atau pengguna database lainnya.

> **Note**
>
> Tabel sistem tidak termasuk dalam backup cluster otomatis atau manual (snapshot). Tampilan sistem STL mempertahankan tujuh hari riwayat log. Mempertahankan log tidak memerlukan tindakan pelanggan apa pun, tetapi jika Anda ingin menyimpan data log selama lebih dari 7 hari, Anda harus menyalinnya secara berkala ke tabel lain atau membongkarnya ke Amazon S3.

Topik

- Jenis tabel dan tampilan sistem
- Visibilitas data dalam tabel dan tampilan sistem
- Memigrasi kueri khusus yang disediakan ke kueri tampilan pemantauan SYS
- Meningkatkan pelacakan pengenal kueri menggunakan tampilan pemantauan SYS
- Kueri tabel sistem, proses, dan id sesi
- Tampilan metadata SVV
- Tampilan pemantauan SYS

- Pemetaan tampilan sistem untuk bermigrasi ke tampilan pemantauan SYS
- Pemantauan sistem (hanya disediakan)
- Tabel katalog sistem

Jenis tabel dan tampilan sistem

Ada beberapa jenis tabel dan tampilan sistem:

- Tampilan SVV berisi informasi tentang objek database dengan referensi ke tabel STV sementara.
- Tampilan SYS digunakan untuk memantau penggunaan kueri dan beban kerja untuk klaster yang disediakan dan grup kerja tanpa server.
- Tampilan STL dihasilkan dari log yang telah disimpan ke disk untuk memberikan riwayat sistem.
- Tabel STV adalah tabel sistem virtual yang berisi snapshot dari data sistem saat ini. Mereka didasarkan pada data dalam memori sementara dan tidak disimpan ke log berbasis disk atau tabel biasa.
- Tampilan SVCS memberikan detail tentang kueri pada kluster penskalaan utama dan konkurensi.
- Tampilan SVL memberikan detail tentang kueri pada cluster utama.

Tabel dan tampilan sistem tidak menggunakan model konsistensi yang sama seperti tabel biasa. Penting untuk mengetahui masalah ini saat menanyakannya, terutama untuk tabel STV dan tampilan SVV. Misalnya, diberikan tabel biasa t1 dengan kolom c1, Anda akan mengharapkan bahwa kueri berikut tidak mengembalikan baris:

```
select * from t1
where c1 > (select max(c1) from t1)
```

Namun, kueri berikut terhadap tabel sistem mungkin mengembalikan baris:

```
select * from stv_exec_state
where currenttime > (select max(currenttime) from stv_exec_state)
```

Alasan kueri ini mungkin mengembalikan baris adalah karena waktu saat ini bersifat sementara dan dua referensi dalam kueri mungkin tidak mengembalikan nilai yang sama saat dievaluasi.

Di sisi lain, kueri berikut mungkin tidak mengembalikan baris:

```
select * from stv_exec_state
where currenttime = (select max(currenttime) from stv_exec_state)
```

Visibilitas data dalam tabel dan tampilan sistem

> **Note**
>
> Amazon Redshift secara otomatis menutupi kolom tabel sistem tertentu saat mencatat informasi tentang kueri yang dibuat ke tampilan Katalog Data untuk mencegah pemaparan metadata sensitif. Untuk informasi selengkapnya, lihat Logging aman di Panduan Manajemen Amazon Redshift.

Ada dua kelas visibilitas untuk data dalam tabel dan tampilan sistem: terlihat oleh pengguna dan terlihat oleh pengguna super.

Hanya pengguna dengan hak superuser yang dapat melihat data dalam tabel yang berada dalam kategori superuser-visible. Pengguna biasa dapat melihat data dalam tabel yang terlihat pengguna. Untuk memberikan akses pengguna reguler ke tabel yang terlihat oleh pengguna super, berikan hak istimewa SELECT pada tabel itu kepada pengguna biasa. Untuk informasi selengkapnya, lihat HIBAH.

Secara default, di sebagian besar tabel yang terlihat pengguna, baris yang dihasilkan oleh pengguna lain tidak terlihat oleh pengguna biasa. Jika pengguna biasa diberikan SYSLOG ACCESS UNRESTRICTED, pengguna tersebut dapat melihat semua baris dalam tabel yang terlihat pengguna, termasuk baris yang dihasilkan oleh pengguna lain. Untuk informasi selengkapnya, lihat ALTER USER atau BUAT PENGGUNA. Semua baris di SVV_TRANSACTIONS dapat dilihat oleh semua pengguna. Untuk informasi selengkapnya tentang visibilitas data, lihat artikel basis AWS re:Post pengetahuan Bagaimana cara mengizinkan izin pengguna reguler database Amazon Redshift untuk melihat data dalam tabel sistem dari pengguna lain untuk klaster saya? .

Untuk tampilan metadata, Amazon Redshift tidak mengizinkan visibilitas ke pengguna yang diberikan SYSLOG ACCESS UNRESTRICTED.

> **Note**
>
> Memberikan pengguna akses tak terbatas ke tabel sistem memberikan visibilitas pengguna ke data yang dihasilkan oleh pengguna lain. Misalnya, STL_QUERY dan STL_QUERY_TEXT

berisi teks lengkap pernyataan INSERT, UPDATE, dan DELETE, yang mungkin berisi data sensitif buatan pengguna.

Superuser dapat melihat semua baris di semua tabel. Untuk memberikan akses pengguna reguler ke tabel superuser-visible, HIBAH PILIH hak istimewa pada tabel itu kepada pengguna biasa.

Memfilter kueri yang dihasilkan sistem

Tabel dan tampilan sistem terkait kueri, seperti SVL_QUERY_SUMMARY, SVL_QLOG, dan lainnya, biasanya berisi sejumlah besar pernyataan yang dibuat secara otomatis yang digunakan Amazon Redshift untuk memantau status database. Kueri yang dihasilkan sistem ini dapat dilihat oleh pengguna super, tetapi jarang berguna. Untuk memfilternya saat memilih dari tabel sistem atau tampilan sistem yang menggunakan `userid` kolom, tambahkan kondisi `userid > 1` ke klausa WHERE. Contoh:

```
select * from svl_query_summary where userid > 1
```

Memigrasi kueri khusus yang disediakan ke kueri tampilan pemantauan SYS

Bermigrasi dari cluster yang disediakan ke Amazon Redshift Tanpa Server

Jika Anda memigrasikan klaster yang disediakan ke Amazon Redshift Tanpa Server, Anda mungkin memiliki kueri menggunakan tampilan sistem berikut, yang hanya menyimpan data dari kluster yang disediakan.

- Semua tampilan STL
- Semua tampilan STV
- Semua tampilan SVCS
- Semua tampilan SVL
- Beberapa tampilan SVV
  - Untuk daftar lengkap tampilan SVV yang tidak didukung di Amazon Redshift Serverless, lihat daftar di bagian bawah Memantau kueri dan beban kerja dengan Amazon Redshift Serverless di Panduan Manajemen Pergeseran Merah Amazon.

Untuk tetap menggunakan kueri Anda, reparasi untuk menggunakan kolom yang ditentukan dalam tampilan pemantauan SYS yang sesuai dengan kolom dalam tampilan khusus yang disediakan. Untuk melihat hubungan pemetaan antara tampilan khusus yang disediakan dan tampilan pemantauan SYS, buka Pemetaan tampilan sistem untuk bermigrasi ke tampilan pemantauan SYS

Memperbarui kueri sambil tetap berada di klaster yang disediakan

Jika Anda tidak bermigrasi ke Amazon Redshift Tanpa Server, Anda mungkin masih ingin memperbarui kueri yang ada. Tampilan pemantauan SYS dirancang untuk kemudahan penggunaan dan mengurangi kompleksitas, menyediakan rangkaian metrik lengkap untuk pemantauan dan pemecahan masalah yang efektif. Menggunakan tampilan SYS seperti SYS_QUERY_HISTORY dan SYS_QUERY_DETAIL yang mengkonsolidasikan informasi dari beberapa tampilan khusus yang disediakan, Anda dapat merampingkan kueri Anda.

Meningkatkan pelacakan pengenal kueri menggunakan tampilan pemantauan SYS

SYS memantau tampilan seperti seperti SYS_QUERY_HISTORY dan SYS_QUERY_DETAIL berisi kolom query_id, yang menyimpan pengenal untuk kueri pengguna. Demikian pula, tampilan hanya disediakan seperti STL_QUERY dan SVL_QLOG berisi kolom kueri, yang juga menyimpan pengidentifikasi kueri. Namun, pengidentifikasi kueri yang direkam dalam tampilan sistem SYS berbeda dari yang direkam dalam tampilan khusus yang disediakan.

Perbedaan antara nilai kolom query_id tampilan SYS dan nilai kolom kueri tampilan khusus yang disediakan adalah sebagai berikut:

- Dalam tampilan SYS, kolom query_id mencatat kueri yang dikirimkan pengguna dalam bentuk aslinya. Pengoptimal Amazon Redshift mungkin memecahnya menjadi kueri turunan untuk meningkatkan kinerja, tetapi satu kueri yang Anda jalankan masih hanya akan memiliki satu baris. SYS_QUERY_HISTORY Jika Anda ingin melihat kueri anak individu, Anda dapat menemukannya diSYS_QUERY_DETAIL.
- Dalam tampilan khusus yang disediakan, kolom kueri merekam kueri pada tingkat kueri anak. Jika pengoptimal Amazon Redshift menulis ulang kueri asli Anda menjadi beberapa kueri turunan, akan ada beberapa baris STL_QUERY dengan nilai pengenal kueri yang berbeda untuk satu kueri yang Anda jalankan.

Saat Anda memigrasikan kueri pemantauan dan diagnostik dari tampilan khusus yang disediakan ke tampilan SYS, pertimbangkan perbedaan ini dan edit kueri Anda sesuai dengan itu. Untuk informasi selengkapnya tentang cara Amazon Redshift memproses kueri, lihat. Perencanaan kueri dan alur kerja pelaksanaan

Contoh

Untuk contoh cara Amazon Redshift merekam kueri secara berbeda dalam tampilan pemantauan khusus yang disediakan dan SYS, lihat contoh kueri berikut. Ini adalah kueri yang ditulis karena Anda akan menjalankannya di Amazon Redshift.

```
SELECT
  s_name
  , COUNT(*) AS numwait
FROM
  supplier,
  lineitem l1,
  orders,
  nation
WHERE    s_suppkey = l1.l_suppkey
         AND o_orderkey = l1.l_orderkey
         AND o_orderstatus = 'F'
         AND l1.l_receiptdate > l1.l_commitdate
         AND EXISTS (SELECT
                       *
                     FROM
                       lineitem l2
                     WHERE  l2.l_orderkey = l1.l_orderkey
                            AND l2.l_suppkey <> l1.l_suppkey )
         AND NOT EXISTS (SELECT
                           *
                         FROM
                           lineitem l3
                         WHERE  l3.l_orderkey = l1.l_orderkey
                                AND l3.l_suppkey <> l1.l_suppkey
                                AND l3.l_receiptdate > l3.l_commitdate )
         AND s_nationkey = n_nationkey
         AND n_name = 'UNITED STATES'
GROUP BY
  s_name
ORDER BY
  numwait DESC
```

```
  , s_name LIMIT 100;
```

Di bawah tenda, pengoptimal kueri Amazon Redshift menulis ulang kueri yang dikirimkan pengguna di atas menjadi 5 kueri turunan.

Query anak pertama membuat tabel sementara untuk mewujudkan subquery.

```
CREATE TEMP TABLE volt_tt_606590308b512(l_orderkey
                                       , l_suppkey
                                       , s_name   ) AS SELECT
                                                     l1.l_orderkey
                                                     , l1.l_suppkey
                                                     , public.supplier.s_name
                                                   FROM
                                                     public.lineitem AS l1,
                                                     public.nation,
                                                     public.orders,
                                                     public.supplier
                                                   WHERE  l1.l_commitdate <
 l1.l_receiptdate
                                                          AND l1.l_orderkey =
 public.orders.o_orderkey
                                                          AND l1.l_suppkey =
 public.supplier.s_suppkey
                                                          AND public.nation.n_name
 = 'UNITED STATES'::CHAR(8)
                                                          AND
 public.nation.n_nationkey = public.supplier.s_nationkey
                                                          AND
 public.orders.o_orderstatus = 'F'::CHAR(1);
```

Kueri anak kedua mengumpulkan statistik dari tabel sementara.

```
padb_fetch_sample: select count(*) from volt_tt_606590308b512;
```

Kueri anak ketiga membuat tabel sementara lain untuk mewujudkan subquery lain, mereferensikan tabel sementara yang dibuat di atas.

```
CREATE TEMP TABLE volt_tt_606590308c2ef(l_orderkey
                                       , l_suppkey) AS (SELECT

 volt_tt_606590308b512.l_orderkey
```

```
                                                              ,
 volt_tt_606590308b512.l_suppkey
                                                            FROM
                                                              public.lineitem AS l2,
                                                              volt_tt_606590308b512
                                                            WHERE  l2.l_suppkey <>
 volt_tt_606590308b512.l_suppkey
                                                                   AND l2.l_orderkey =
 volt_tt_606590308b512.l_orderkey)
                                                                   EXCEPT distinct (SELECT
 volt_tt_606590308b512.l_orderkey, volt_tt_606590308b512.l_suppkey
                                                                   FROM public.lineitem AS
 l3, volt_tt_606590308b512
                                                                   WHERE l3.l_commitdate <
 l3.l_receiptdate
                                                                     AND l3.l_suppkey <>
 volt_tt_606590308b512.l_suppkey
                                                                     AND l3.l_orderkey =
 volt_tt_606590308b512.l_orderkey);
```

Kueri anak keempat kembali mengumpulkan statistik tabel sementara.

```
padb_fetch_sample: select count(*) from volt_tt_606590308c2ef
```

Kueri anak terakhir menggunakan tabel sementara yang dibuat di atas untuk menghasilkan output.

```
SELECT
  volt_tt_606590308b512.s_name AS s_name
  , COUNT(*) AS numwait
FROM
  volt_tt_606590308b512,
  volt_tt_606590308c2ef
WHERE    volt_tt_606590308b512.l_orderkey = volt_tt_606590308c2ef.l_orderkey
         AND volt_tt_606590308b512.l_suppkey = volt_tt_606590308c2ef.l_suppkey
GROUP BY
  1
ORDER BY
  2 DESC
  , 1 ASC LIMIT 100;
```

Dalam tampilan sistem khusus yang disediakan STL_QUERY, Amazon Redshift merekam lima baris pada tingkat kueri turunan, sebagai berikut:

```
SELECT userid, xid, pid, query, querytxt::varchar(100);
FROM stl_query
WHERE xid = 48237350
ORDER BY xid, starttime;

 userid |   xid    |    pid     |  query   |
    querytxt
--------+----------+------------+----------
+-----------------------------------------------------------------------------------------------------------
    101 | 48237350 | 1073840810 | 12058151 | CREATE TEMP TABLE
 volt_tt_606590308b512(l_orderkey, l_suppkey, s_name) AS SELECT l1.l_orderkey, l1.l
    101 | 48237350 | 1073840810 | 12058152 | padb_fetch_sample: select count(*) from
 volt_tt_606590308b512
    101 | 48237350 | 1073840810 | 12058156 | CREATE TEMP TABLE
 volt_tt_606590308c2ef(l_orderkey, l_suppkey) AS (SELECT volt_tt_606590308b512.l_or
    101 | 48237350 | 1073840810 | 12058168 | padb_fetch_sample: select count(*) from
 volt_tt_606590308c2ef
    101 | 48237350 | 1073840810 | 12058170 | SELECT s_name , COUNT(*) AS numwait FROM
 supplier, lineitem l1, orders, nation WHERE s_suppkey = l1.
(5 rows)
```

Dalam tampilan pemantauan SYS SYS_QUERY_HISTORY, Amazon Redshift merekam kueri sebagai berikut:

```
SELECT user_id, transaction_id, session_id, query_id, query_text::varchar(100)
FROM sys_query_history
WHERE transaction_id = 48237350
ORDER BY start_time;

 user_id | transaction_id | session_id | query_id |
          query_text
---------+----------------+------------+----------
+-----------------------------------------------------------------------------------------------------------
     101 |       48237350 | 1073840810 | 12058149 | SELECT s_name , COUNT(*) AS numwait
 FROM supplier, lineitem l1, orders, nation WHERE s_suppkey = l1.
```

Di SYS_QUERY_DETAIL, Anda dapat menemukan detail tingkat kueri anak menggunakan nilai query_id dari SYS_QUERY_HISTORY. Kolom child_query_sequence menunjukkan urutan kueri anak dieksekusi. Untuk informasi selengkapnya tentang kolom di SYS_QUERY_DETAIL, lihat. SYS_QUERY_DETAIL

```
select user_id,
```

```
       query_id,
       child_query_sequence,
       stream_id,
       segment_id,
       step_id,
       start_time,
       end_time,
       duration,
       blocks_read,
       blocks_write,
       local_read_io,
       remote_read_io,
       data_skewness,
       time_skewness,
       is_active,
       spilled_block_local_disk,
       spilled_block_remote_disk
from sys_query_detail
where query_id = 12058149
      and step_id = -1
order by query_id,
         child_query_sequence,
         stream_id,
         segment_id,
         step_id;

 user_id | query_id | child_query_sequence | stream_id | segment_id | step_id |
       start_time         |          end_time          | duration | blocks_read |
 blocks_write | local_read_io | remote_read_io | data_skewness | time_skewness |
 is_active | spilled_block_local_disk | spilled_block_remote_disk
---------+----------+----------------------+-----------+------------+---------
+----------------------------+----------------------------+----------+-------------
+--------------+---------------+----------------+---------------+---------------
+-----------+---------------------------+---------------------------
     101 | 12058149 |                    1 |         0 |          0 |      -1 |
 2023-09-27 15:40:38.512415 | 2023-09-27 15:40:38.533333 |    20918 |           0 |
         0 |             0 |              0 |             0 |            44 | f
 |                         0 |                            0
     101 | 12058149 |                    1 |         1 |          1 |      -1 |
 2023-09-27 15:40:39.931437 | 2023-09-27 15:40:39.972826 |    41389 |          12 |
         0 |            12 |              0 |             0 |            77 | f
 |                         0 |                            0
     101 | 12058149 |                    1 |         2 |          2 |      -1 |
 2023-09-27 15:40:40.584412 | 2023-09-27 15:40:40.613982 |    29570 |          32 |
```

```
        0 |            32 |              0 |             0 |           25 | f
|                        0 |                           0
    101 | 12058149 |                    1 |         2 |          3 |      -1 |
2023-09-27 15:40:40.582038 | 2023-09-27 15:40:40.615758 |    33720 |           0 |
        0 |             0 |              0 |             0 |            1 | f
|                        0 |                           0
    101 | 12058149 |                    1 |         3 |          4 |      -1 |
2023-09-27 15:40:46.668766 | 2023-09-27 15:40:46.705456 |    36690 |          24 |
        0 |            15 |              0 |             0 |           17 | f
|                        0 |                           0
    101 | 12058149 |                    1 |         4 |          5 |      -1 |
2023-09-27 15:40:46.707209 | 2023-09-27 15:40:46.709176 |     1967 |           0 |
        0 |             0 |              0 |             0 |           18 | f
|                        0 |                           0
    101 | 12058149 |                    1 |         4 |          6 |      -1 |
2023-09-27 15:40:46.70656  | 2023-09-27 15:40:46.71289  |     6330 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                        0 |                           0
    101 | 12058149 |                    1 |         5 |          7 |      -1 |
2023-09-27 15:40:46.71405  | 2023-09-27 15:40:46.714343 |      293 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                        0 |                           0
    101 | 12058149 |                    2 |         0 |          0 |      -1 |
2023-09-27 15:40:52.083907 | 2023-09-27 15:40:52.087854 |     3947 |           0 |
        0 |             0 |              0 |             0 |           35 | f
|                        0 |                           0
    101 | 12058149 |                    2 |         1 |          1 |      -1 |
2023-09-27 15:40:52.089632 | 2023-09-27 15:40:52.091129 |     1497 |           0 |
        0 |             0 |              0 |             0 |           11 | f
|                        0 |                           0
    101 | 12058149 |                    2 |         1 |          2 |      -1 |
2023-09-27 15:40:52.089008 | 2023-09-27 15:40:52.091306 |     2298 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         0 |          0 |      -1 |
2023-09-27 15:40:56.882013 | 2023-09-27 15:40:56.897282 |    15269 |           0 |
        0 |             0 |              0 |             0 |           29 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         1 |          1 |      -1 |
2023-09-27 15:40:59.718554 | 2023-09-27 15:40:59.722789 |     4235 |           0 |
        0 |             0 |              0 |             0 |           13 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         2 |          2 |      -1 |
2023-09-27 15:40:59.800382 | 2023-09-27 15:40:59.807388 |     7006 |           0 |
```

```
        0 |             0 |              0 |             0 |           58 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         3 |          3 |      -1 |
2023-09-27 15:41:06.488685 | 2023-09-27 15:41:06.493825 |     5140 |           0 |
        0 |             0 |              0 |             0 |           56 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         3 |          4 |      -1 |
2023-09-27 15:41:06.486206 | 2023-09-27 15:41:06.497756 |    11550 |           0 |
        0 |             0 |              0 |             0 |            2 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         4 |          5 |      -1 |
2023-09-27 15:41:06.499201 | 2023-09-27 15:41:06.500851 |     1650 |           0 |
        0 |             0 |              0 |             0 |           15 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         4 |          6 |      -1 |
2023-09-27 15:41:06.498609 | 2023-09-27 15:41:06.500949 |     2340 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                        0 |                           0
    101 | 12058149 |                    3 |         5 |          7 |      -1 |
2023-09-27 15:41:06.502945 | 2023-09-27 15:41:06.503282 |      337 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                        0 |                           0
    101 | 12058149 |                    4 |         0 |          0 |      -1 |
2023-09-27 15:41:06.62899  | 2023-09-27 15:41:06.631452 |     2462 |           0 |
        0 |             0 |              0 |             0 |           22 | f
|                        0 |                           0
    101 | 12058149 |                    4 |         1 |          1 |      -1 |
2023-09-27 15:41:06.632313 | 2023-09-27 15:41:06.63391  |     1597 |           0 |
        0 |             0 |              0 |             0 |           20 | f
|                        0 |                           0
    101 | 12058149 |                    4 |         1 |          2 |      -1 |
2023-09-27 15:41:06.631726 | 2023-09-27 15:41:06.633813 |     2087 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                        0 |                           0
    101 | 12058149 |                    5 |         0 |          0 |      -1 |
2023-09-27 15:41:12.571974 | 2023-09-27 15:41:12.584234 |    12260 |           0 |
        0 |             0 |              0 |             0 |           39 | f
|                        0 |                           0
    101 | 12058149 |                    5 |         0 |          1 |      -1 |
2023-09-27 15:41:12.569815 | 2023-09-27 15:41:12.585391 |    15576 |           0 |
        0 |             0 |              0 |             0 |            4 | f
|                        0 |                           0
    101 | 12058149 |                    5 |         1 |          2 |      -1 |
2023-09-27 15:41:13.758513 | 2023-09-27 15:41:13.76401  |     5497 |           0 |
```

```
         0 |             0 |              0 |             0 |            39 | f
 |                        0 |                           0
     101 | 12058149 |                    5 |         1 |          3 |      -1 |
 2023-09-27 15:41:13.749    | 2023-09-27 15:41:13.772987 |    23987 |           0 |
         0 |             0 |              0 |             0 |            32 | f
 |                        0 |                           0
     101 | 12058149 |                    5 |         2 |          4 |      -1 |
 2023-09-27 15:41:13.799526 | 2023-09-27 15:41:13.813506 |    13980 |           0 |
         0 |             0 |              0 |             0 |            62 | f
 |                        0 |                           0
     101 | 12058149 |                    5 |         2 |          5 |      -1 |
 2023-09-27 15:41:13.798823 | 2023-09-27 15:41:13.813651 |    14828 |           0 |
         0 |             0 |              0 |             0 |             0 | f
 |                        0 |                           0
(28 rows)
```

Kueri tabel sistem, proses, dan id sesi

Saat menganalisis kueri, proses, dan id sesi yang muncul di tabel sistem, perhatikan hal-hal berikut:

- Nilai id kueri (dalam kolom seperti `query_id` dan`query`) dapat digunakan kembali dari waktu ke waktu.
- Id proses atau nilai id sesi (dalam kolom seperti`process_id`,`pid`, dan`session_id`) dapat digunakan kembali dari waktu ke waktu.
- Nilai id transaksi (dalam kolom seperti `transaction_id` dan`xid`) adalah unik.

Tampilan metadata SVV

Tampilan SVV adalah tampilan sistem di Amazon Redshift yang berisi informasi tentang objek database. Perhatikan bahwa meskipun mereka menyimpan informasi seperti izin pengguna atau nama tabel, mereka tidak dimaksudkan untuk bergabung dengan hubungan yang dibuat pengguna.

> **Note**
>
> Amazon Redshift melaporkan PERINGATAN, bukan KESALAHAN, jika respons database gagal karena alasan apa pun. Amazon Redshift tidak mengirim pesan ERROR saat Anda menanyakan objek di datashare.

Topik

- SVV_ACTIVE_KURSOR
- SVV_ALL_KOLOM
- SVV_ALL_SCHEMAS
- SVV_ALL_TABEL
- SVV_ALTER_TABLE_REKOMENDASI
- KEBIJAKAN PENYAMARAN SVV_TERLAMPIR
- SVV_KOLOM
- SVV_COLUMN_PRIVILEGIA
- SVV_COPY_JOB_INTEGRASI
- SVV_DATABASE_HAK ISTIMEWA
- HAK ISTIMEWA SVV_DATASHARE_
- SVV_DATASHARES
- SVV_DATASHARE_KONSUMEN
- SVV_DATASHARE_OBJECT
- SVV_DEFAULT_HAK ISTIMEWA
- SVV_DISKUSI
- SVV_KOLON_EKSTERNAL_
- SVV_EXTERNAL_DATABASE
- SVV_PARTISI_EKSTERNAL
- SVV_EXTERNAL_SCHEMAS
- SVV_EXTERNAL_TABEL
- SVV_FUNGSI_HAK ISTIMEWA
- SVV_GEOGRAPHY_COLUMN_
- SVV_GEOMETRI_KOLOM
- SVV_IAM_HAK ISTIMEWA
- SVV_IDENTITY_PROVIDER
- SVV_INTEGRASI

- SVV_INTEGRASI_TABEL_PEMETAAN
- SVV_INTEGRASI_TABLE_STATE
- SVV_KOLUMN_INTERLEAVED_
- SVV_LANGUAGE_KEISTIMEWAAN
- SVV_MASKING_POLICY
- SVV_ML_MODEL_INFO
- SVV_ML_MODEL_HAK ISTIMEWA
- SVV_MV_DEPENDENSI
- SVV_MV_INFO
- SVV_QUERY_DALAM PENERBANGAN
- SVV_QUERY_STATE
- SVV_REDSHIFT_KOLOM
- SVV_REDSHIFT_DATABASE
- SVV_REDSHIFT_FUNCTION
- SVV_REDSHIFT_SCHEMA_KUOTA
- SVV_REDSHIFT_SKEMA
- SVV_REDSHIFT_TABEL
- HAK ISTIMEWA SVV_RELASI
- SVV_RLS_APPLICED_POLICY
- SVV_RLS_TERLAMPIR_KEBIJAKAN
- SVV_RLS_POLICY
- SVV_RLS_RELASI
- SVV_ROLE_GRANT
- SVV_ROLE
- SVV_SCHEMA_HAK ISTIMEWA
- SVV_SCHEMA_KUOTA_STATUS
- SVV_SYSTEM_HAK ISTIMEWA
- SVV_TABLE_INFO

- SVV_TABEL
- SVV_TRANSAKSI
- SVV_USER_GRANT
- SVV_USERINFO
- SVV_VACUUM_PROGRES
- SVV_VACUUM_RINGKASAN

SVV_ACTIVE_KURSOR

SVV_ACTIVE_CURSORS menampilkan detail untuk kursor yang sedang terbuka. Untuk informasi selengkapnya, lihat MENYATAKAN.

SVV_ACTIVE_CURSORS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem. Pengguna hanya dapat melihat kursor yang dibuka oleh pengguna tersebut. Seorang superuser dapat melihat semua kursor.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang membuat kursor.
kursor_nama	bangkai (128)	Nama kursor.
id transaksi_	big tits (128)	ID transaksi.
session_id	integer	ID proses dengan kursor aktif.
deklarasi_waktu	timestamp	Waktu kursor dideklarasikan.
total_byte	bigint	Ukuran set hasil kursor, dalam byte.
total_baris	bigint	Jumlah baris dalam set hasil kursor.

Nama kolom	Jenis data	Deskripsi
ditampat_baris	bigint	Jumlah baris yang saat ini diambil dari kumpulan hasil kursor.
kursor_penyimpanan_batas_persentan yang digunakan	integer	Persentase ruang disk yang saat ini digunakan oleh kursor.

SVV_ALL_KOLOM

Gunakan SVV_ALL_COLUMNS untuk melihat penyatuan kolom dari tabel Amazon Redshift seperti yang ditunjukkan pada SVV_REDSHIFT_COLUMNS dan daftar gabungan semua kolom eksternal dari semua tabel eksternal. Untuk informasi tentang kolom Amazon Redshift, lihatSVV_REDSHIFT_KOLOM.

SVV_ALL_COLUMNS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW COLUMNS untuk penemuan kolom. SHOW COLUMNS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama basis data.
schema_name	bangkai (128)	Nama skema.

Nama kolom	Jenis data	Deskripsi
table_name	bangkai (128)	Nama tabel.
column_name	bangkai (128)	Nama kolom.
ordinal_position	integer	Posisi kolom dalam tabel.
column_default	varchar(4000)	Nilai default kolom.
is_nullable	Cucing (3)	Nilai yang menunjukkan apakah kolom tersebut dapat dibatalkan. Nilai yang mungkin adalah ya dan tidak.
data_type	bangkai (128)	Jenis data kolom.
karakter_panjang_maksimum	integer	Jumlah maksimum karakter dalam kolom.
presisi numerik	integer	Presisi numerik.
skala numerik	integer	Skala numerik.
remarks	bangsawan (256)	Keterangan.

Kueri Sampel

Contoh berikut mengembalikan output dari SVV_ALL_COLUMNS.

```
SELECT *
FROM svv_all_columns
WHERE database_name = 'tickit_db'
    AND TABLE_NAME = 'tickit_sales_redshift'
ORDER BY COLUMN_NAME,
    SCHEMA_NAME
LIMIT 5;

 database_name | schema_name |     table_name        | column_name | ordinal_position
 | column_default | is_nullable | data_type | character_maximum_length |
 numeric_precision | numeric_scale | remarks
```

```
 -------------+------------+----------------------+------------+-----------------
+---------------+------------+----------+-------------------------
+-------------------+--------------+---------
   tickit_db   |    public   | tickit_sales_redshift |    buyerid  |        4         |
               |      NO     |  integer  |                          |         32
  |       0       |
   tickit_db   |    public   | tickit_sales_redshift | commission  |        9         |
               |     YES     |  numeric  |                          |          8
  | 2       |
   tickit_db   |    public   | tickit_sales_redshift |    dateid   |        7         |
               |      NO     |  smallint |                          |         16
  |       0       |
   tickit_db   |    public   | tickit_sales_redshift |   eventid   |        5         |
               |      NO     |  integer  |                          |         32
  |       0       |
   tickit_db   |    public   | tickit_sales_redshift |    listid   |        2         |
               |      NO     |  integer  |                          |         32
  |       0       |
```

SVV_ALL_SCHEMAS

Gunakan SVV_ALL_SCHEMAS untuk melihat penyatuan skema Amazon Redshift seperti yang ditunjukkan pada SVV_REDSHIFT_SCHEMAS dan daftar konsolidasi semua skema eksternal dari semua database. Untuk informasi selengkapnya tentang skema Amazon Redshift, lihat. SVV_REDSHIFT_SKEMA

SVV_ALL_SCHEMAS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW SCHEMAS untuk penemuan skema. SHOW SCHEMAS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama database tempat skema ada.
schema_name	bangkai (128)	Nama skema.
pemilik skema	integer	ID pengguna pemilik skema. Untuk informasi tentang ID pengguna, lihatPG_USER_INFO.
tipe_skema	bangkai (128)	Jenis skema. Nilai yang mungkin adalah skema eksternal, lokal, dan bersama.
schema_acl	bangkai (128)	String yang mendefinisikan izin untuk pengguna atau grup pengguna yang ditentukan untuk skema.
sumber_database	bangkai (128)	Nama database sumber untuk skema eksternal.
opsi_skema	bangsawan (256)	Opsi skema. Ini adalah atribut skema eksternal.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_ALL_SCHEMAS.

```
SELECT *
FROM svv_all_schemas
WHERE database_name = 'tickit_db'
ORDER BY database_name,
    SCHEMA_NAME;
```

```
 database_name |    schema_name     | schema_owner | schema_type | schema_acl |
 source_database | schema_option
--------------+--------------------+-------------+------------+------------
+-----------------+---------------
   tickit_db   |       public       |       1      |   shared    |            |
         |
```

SVV_ALL_TABEL

Gunakan SVV_ALL_TABLES untuk melihat penyatuan tabel Amazon Redshift seperti yang ditunjukkan pada SVV_REDSHIFT_TABLES dan daftar gabungan semua tabel eksternal dari semua skema eksternal. Untuk informasi tentang tabel Amazon Redshift, lihatSVV_REDSHIFT_TABEL.

SVV_ALL_TABLES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW TABLES untuk penemuan tabel. SHOW TABLES bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama database tempat tabel itu ada.
schema_name	bangkai (128)	Nama skema untuk tabel.
table_name	bangkai (128)	Nama tabel.

Nama kolom	Jenis data	Deskripsi
table_acl	bangkai (128)	String yang mendefinisikan izin untuk pengguna atau grup pengguna tertentu untuk tabel.
table_type	bangkai (128)	Jenis meja. Nilai yang mungkin adalah tampilan, tabel dasar, tabel eksternal, dan tabel bersama.
remarks	bangsawan (256)	Keterangan.

Kueri Sampel

Contoh berikut mengembalikan output dari SVV_ALL_TABLES.

```
SELECT *
FROM svv_all_tables
WHERE database_name = 'tickit_db'
ORDER BY TABLE_NAME,
    SCHEMA_NAME
LIMIT 5;

 database_name | schema_name |        table_name        | table_type | table_acl |
 remarks
---------------+-------------+--------------------------+------------+-----------
+---------
   tickit_db   |    public   | tickit_category_redshift |    TABLE   |           |
   tickit_db   |    public   |   tickit_date_redshift   |    TABLE   |           |
   tickit_db   |    public   |   tickit_event_redshift  |    TABLE   |           |
   tickit_db   |    public   | tickit_listing_redshift  |    TABLE   |           |
   tickit_db   |    public   |   tickit_sales_redshift  |    TABLE   |           |
```

Jika nilai table_acl adalah null, tidak ada hak akses yang secara eksplisit diberikan ke tabel yang sesuai.

SVV_ALTER_TABLE_REKOMENDASI

Merekam rekomendasi Amazon Redshift Advisor saat ini untuk tabel. Tampilan ini menunjukkan rekomendasi untuk semua tabel, apakah mereka ditentukan untuk pengoptimalan otomatis atau tidak. Untuk melihat apakah tabel didefinisikan untuk pengoptimalan otomatis, lihatSVV_TABLE_INFO. Entri hanya muncul untuk tabel yang terlihat di database sesi saat ini. Setelah rekomendasi diterapkan (baik oleh Amazon Redshift atau oleh Anda), itu tidak lagi muncul di tampilan.

SVV_ALTER_TABLE_RECOMMENDATIONS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
jenis	karakter (30)	Jenis rekomendasi. Nilai yang mungkin adalah distkey dan sortkey.
basis data	karakter (128)	Nama database.
tabel_id	integer	Pengidentifikasi tabel.
grup_id	integer	Nomor kelompok dari serangkaian rekomendasi. Semua rekomenda si dalam kelompok harus diterapkan untuk melihat manfaat maksimal. Nilai yang mungkin adalah -1 untuk rekomendasi kunci pengurutan, dan angka yang lebih besar dari nol untuk rekomendasi kunci distribus i.
ddl	karakter (1024)	Pernyataan SQL yang harus dijalankan untuk menerapkan rekomenda si.
otomatis memenuhi syarat	karakter (1)	Nilai menunjukkan apakah rekomendasi memenuhi syarat untuk menjalankan Amazon Redshift secara otomatis. Jika nilai init, maka indikasinya benar, jika f kemudian salah.

Kueri Sampel

Dalam contoh berikut, baris dalam hasil menunjukkan rekomendasi untuk kunci distribusi dan kunci pengurutan. Baris juga menunjukkan apakah rekomendasi memenuhi syarat untuk Amazon Redshift untuk menerapkannya secara otomatis.

```
select type, database, table_id, group_id, ddl, auto_eligible
from svv_alter_table_recommendations;
```

```
 type      | database | table_id | group_id | ddl

                   | auto_eligible
diststyle | db0      | 117884   | 2        | ALTER TABLE "sch"."dp21235_tbl_1" ALTER
DISTSTYLE KEY DISTKEY "c0"
                     | f
diststyle | db0      | 117892   | 2        | ALTER TABLE "sch"."dp21235_tbl_1" ALTER
DISTSTYLE KEY DISTKEY "c0"
                     | f
diststyle | db0      | 117885   | 1        | ALTER TABLE "sch"."catalog_returns"
ALTER DISTSTYLE KEY DISTKEY "cr_sold_date_sk", ALTER COMPOUND SORTKEY
("cr_sold_date_sk","cr_returned_time_sk") | t
sortkey   | db0      | 117890   | -1       | ALTER TABLE "sch"."customer_addresses"
ALTER COMPOUND SORTKEY ("ca_address_sk")
                      | t
```

KEBIJAKAN PENYAMARAN SVV_TERLAMPIR

Gunakan SVV_ATTACHED_MASKING_POLICY untuk melihat semua hubungan dan roles/users dengan kebijakan yang dilampirkan pada database yang saat ini terhubung.

Hanya pengguna super dan pengguna dengan `sys:secadmin` peran yang dapat melihat SVV_ATTACHED_MASKING_POLICY. Pengguna biasa akan melihat 0 baris.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_kebijakan	text	Nama kebijakan masking yang dilampirkan pada tabel.

Nama kolom	Jenis data	Deskripsi
schema_name	text	Skema tabel tempat kebijakan dilampirkan.
table_name	text	Nama tabel tempat kebijakan dilampirkan.
table_type	text	Jenis tabel tempat kebijakan dilampirkan.
pemberi	text	Nama pengguna yang melampirkan kebijakan.
penerima hibah	text	Nama orang yang user/ role dilampirkan kebijakan tersebut.
tipe penerima hibah	text	Jenis penerima hibah. Ini bisa peran, pengguna, atau publik.
Prioritas	int	Prioritas kebijakan terlampir.
masukan_kolom	text	Atribut kolom input dari kebijakan terlampir.
kolom keluaran	text	Atribut kolom keluaran dari kebijakan terlampir.
is_masking_datashare_on	boolean	Apakah tabel yang dilampirk an kebijakan DDM-protected melebihi data share.

Fungsi internal

SVV_ATTACHED_MASKING_POLICY mendukung fungsi internal berikut:

maske_get_kebijakan_untuk_rol_on_kolom

Dapatkan kebijakan prioritas tertinggi yang berlaku untuk column/role pasangan tertentu.

Sintaksis

```
mask_get_policy_for_role_on_column
                        (relschema,
                        relname,
                        colname,
                        rolename);
```

Parameter

jadwal relasi

Nama skema kebijakan tersebut.

rename

Nama tabel di mana kebijakan berada.

colname

Nama kolom yang dilampirkan kebijakan.

nama peran

Nama peran yang dilampirkan kebijakan.

maske_get_kebijakan_untuk_pengguna_di_kolom

Dapatkan kebijakan prioritas tertinggi yang berlaku untuk column/user pasangan tertentu.

Sintaksis

```
mask_get_policy_for_user_on_column
                        (relschema,
                        relname,
                        colname,
                        username);
```

Parameter

jadwal relasi

Nama skema di mana kebijakan tersebut berada.

rename

Nama tabel di mana kebijakan berada.

colname

Nama kolom yang dilampirkan kebijakan.

nama peran

Nama pengguna yang dilampirkan kebijakan.

SVV_KOLOM

Gunakan SVV_COLUMNS untuk melihat informasi katalog tentang kolom tabel dan tampilan lokal dan eksternal, termasuk tampilan pengikatan terlambat.

SVV_COLUMNS terlihat oleh semua pengguna secara default. Untuk mengontrol akses ke metadata database Anda, aktifkan keamanan metadata untuk cluster yang disediakan atau grup kerja tanpa server. Keamanan metadata memungkinkan Anda memisahkan izin tampilan untuk metadata objek berdasarkan pengguna dan peran. Untuk informasi selengkapnya, lihat Keamanan metadata.

Tampilan SVV_COLUMNS menggabungkan metadata tabel dari Tabel katalog sistem (tabel dengan awalan PG) dan tampilan sistem. SVV_KOLON_EKSTERNAL_ Tabel katalog sistem menjelaskan tabel database Amazon Redshift. SVV_EXTERNAL_COLUMNS menjelaskan tabel eksternal yang digunakan dengan Amazon Redshift Spectrum.

Semua pengguna dapat melihat semua baris dari tabel katalog sistem. Pengguna biasa dapat melihat definisi kolom dari tampilan SVV_EXTERNAL_COLUMNS hanya untuk tabel eksternal yang telah diberikan akses kepada mereka. Meskipun pengguna biasa dapat melihat metadata tabel dalam tabel katalog sistem, mereka hanya dapat memilih data dari tabel yang ditentukan pengguna jika mereka memiliki tabel atau telah diberikan akses.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW COLUMNS untuk penemuan kolom. SHOW COLUMNS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
table_catalog	text	Nama katalog tempat meja berada.
table_schema	text	Nama skema untuk tabel.
table_name	text	Nama tabel.
column_name	text	Nama kolom.
ordinal_position	int	Posisi kolom dalam tabel.
column_default	text	Nilai default kolom.
is_nullable	text	Nilai yang menunjukkan apakah kolom tersebut dapat dibatalkan.
data_type	text	Jenis data kolom.
karakter_panjang_maksimum	int	Jumlah maksimum karakter dalam kolom.
presisi numerik	int	Presisi numerik. Jika kolom data_type adalah numerik, kolom ini mengembalikan jumlah digit signifikan di seluruh nilai.

Nama kolom	Jenis data	Deskripsi
numerik_presision_radix	int	Radix presisi numerik. Jika kolom data_type adalah numerik, kolom ini mengembalikan dasar kolom numeric_precision dan numeric_scale.
skala numerik	int	Skala numerik. Jika kolom data_type adalah numerik, kolom ini mengembalikan jumlah digit signifikan dalam nilai desimal.
ketepatan waktu	int	Presisi datetime.
tipe_interval	text	Jenis interval.
interval_presisi	text	Presisi interval.
Character_set Catalog	text	Katalog set karakter.
Character_Set_Schema	text	Skema set karakter.
nama_set_karakter	text	Nama set karakter.
collation_catalog	text	Katalog kolasi.
skema pengumpulan	text	Skema kolasi.
nama_koleksi	text	Nama kolasi.
domain_name	text	Nama domain.
remarks	text	Keterangan.

SVV_COLUMN_PRIVILEGIA

Gunakan SVV_COLUMN_PRIVILEGES untuk melihat izin kolom yang secara eksplisit diberikan kepada pengguna, peran, dan grup dalam database saat ini.

SVV_COLUMN_PRIVILEGES terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW COLUMN GRANTS untuk penemuan izin. SHOW COLUMN GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_ruang_nama	text	Nama namespace di mana relasi tertentu ada.
nama relasi	text	Nama relasi.
column_name	text	Nama kolom.
keistimewa_tipe	text	Jenis izin. Nilai yang mungkin adalah SELECT atau UPDATE.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.

Nama kolom	Jenis data	Deskripsi
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup atau publik.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_COLUMN_PRIVILEGES.

```
SELECT
 namespace_name,relation_name,COLUMN_NAME,privilege_type,identity_name,identity_type
FROM svv_column_privileges WHERE relation_name = 'lineitem';

 namespace_name | relation_name | column_name | privilege_type | identity_name |
 identity_type
----------------+---------------+-------------+----------------+---------------
+----------------
    public      |   lineitem    | l_orderkey  |     SELECT     |    reguser    |
 user
    public      |   lineitem    | l_orderkey  |     SELECT     |     role1     |
 role
    public      |   lineitem    | l_partkey   |     SELECT     |    reguser    |
 user
    public      |   lineitem    | l_partkey   |     SELECT     |     role1     |
 role
```

SVV_COPY_JOB_INTEGRASI

Gunakan SVV_COPY_JOB_INTEGRATIONS untuk melihat detail integrasi acara S3.

Tampilan ini berisi integrasi peristiwa S3 yang telah dibuat.

SVV_COPY_JOB_INTEGRATIONS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
pemilik pekerjaan	integer	Pengidentifikasi pemilik pekerjaan.
saluran_arn	name	Pengidentifikasi integrasi.
bucket	text	Nama bucket Amazon S3 yang terkait dengan integrasi.
status saluran	name	Keadaan integrasi. Nilai yang valid:`Pending`,`Established` , dan`Inactive`.
db_name	name	Nama database dari objek dependen.
nama_pekerjaan	text	Nama pekerjaan.
status pekerjaan	integer	Keadaan pekerjaan. Nilai yang valid: `0` untuk aktif, 1 untuk tertunda.

Contoh berikut mengembalikan integrasi S3 untuk database saat ini.

```
SELECT * FROM SVV_COPY_JOB_INTEGRATIONS WHERE db_name = pg_catalog.current_database();
```

SVV_DATABASE_HAK ISTIMEWA

Gunakan SVV_DATABASE_PRIVILEGES untuk melihat izin database yang secara eksplisit diberikan kepada pengguna, peran, dan grup di cluster Amazon Redshift Anda.

SVV_DATABASE_PRIVILEGES terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW GRANTS untuk penemuan izin. SHOW GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	text	Nama basis data.
tipe hak istimewa	text	Jenis izin. Untuk izin dengan privilege_scope database, nilai yang mungkin adalah USAGE, CREATE, TEMPORARY, TEMP, dan ALTER. Untuk nilai privilege_scope selain database, nilai yang mungkin termasuk jenis izin apa pun yang tersedia pada cakupan izin.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.

Nama kolom	Jenis data	Deskripsi
cakupan hak istimewa	text	Ruang lingkup izin yang ditentukan dalam privilege_type. Kemungkinan nilainya adalah sebagai berikut: • BASIS DATA • SKEMA • TABEL • FUNGSI • BAHASA Untuk informasi tentang izin cakupan, buka. Izin terlingkup

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_DATABASE_PRIVILEGES.

```
SELECT database_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_database_privileges
WHERE database_name = 'test_db';

 database_name | privilege_type | identity_name | identity_type | admin_option
---------------+----------------+---------------+---------------+--------------
     test_db   |     CREATE     |     reguser   |      user     |     False
     test_db   |     CREATE     |      role1    |      role     |     False
     test_db   |     TEMP       |      public   |      public   |     False
     test_db   |     TEMP       |      role1    |      role     |     False
```

HAK ISTIMEWA SVV_DATASHARE_

Gunakan SVV_DATASHARE_PRIVILEGES untuk melihat izin datashare yang secara eksplisit diberikan kepada pengguna, peran, dan grup di cluster Amazon Redshift Anda.

SVV_DATASHARE_PRIVILEGES terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
datahare_name	text	Nama datashare.
keistimewa_tipe	text	Jenis izin. Nilai yang mungkin adalah ALTER atau SHARE.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_DATASHARE_PRIVILEGES.

```
SELECT datashare_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_datashare_privileges
WHERE datashare_name = 'demo_share';

 datashare_name | privilege_type |  identity_name | identity_type | admin_option
----------------+----------------+----------------+---------------+--------------
   demo_share   |     ALTER      |    superuser   |     user      |   False
   demo_share   |     ALTER      |    reguser     |     user      |   False
```

SVV_DATASHARES

Gunakan SVV_DATASHARES untuk melihat daftar data share yang dibuat di cluster, dan berbagi data dengan cluster.

SVV_DATASHARES terlihat oleh pengguna berikut:

- Superuser
- Pemilik Datashare
- Pengguna dengan izin ALTER atau USAGE pada datashare

Pengguna lain tidak dapat melihat baris apa pun. Untuk informasi tentang izin ALTER dan USAGE, lihatHIBAH.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW DATASHARES untuk penemuan datashare. SHOW DATASHARES diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_berbagi	bangkai (128)	Nama datashare.
share_id	integer	ID datashare.
pemilai_saham	integer	Pemilik datashare.
sumber_database	bangkai (128)	Basis data sumber untuk datashare ini.
Database_konsumen	bangkai (128)	Basis data konsumen yang dibuat dari datashare ini.

Nama kolom	Jenis data	Deskripsi
tipe_bagikan	Washer (8)	Jenis datashare. Nilai yang mungkin adalah INBOUND dan OUTBOUND.
kreatiedate	stempel waktu tanpa zona waktu	Tanggal ketika datashare dibuat.
adalah_dapat diakses publik	boolean	Properti yang menentuka n apakah datashare dapat dibagikan ke cluster yang dapat diakses publik.
share_acl	bangsawan (256)	String yang mendefinisikan izin untuk pengguna atau grup pengguna yang ditentukan untuk datashare.
akun_produsen	varchar(16)	ID untuk akun produsen datashare.
ruang nama produsen	varchar(64)	Pengidentifikasi cluster unik untuk cluster produsen datashare.
dikelola oleh	varchar(64)	Properti yang menentukan AWS layanan yang mengelola datashare.

Catatan penggunaan

Mengambil metadata tambahan — Menggunakan integer yang dikembalikan di `share_owner` kolom, Anda dapat bergabung dengan `usesysid` in SVL_USER_INFO untuk mendapatkan data tentang pemilik datashare. Ini termasuk nama dan properti tambahan.

Contoh kueri

Contoh berikut mengembalikan output untuk SVV_DATASHARES.

```
SELECT share_owner, source_database, share_type, is_publicaccessible
FROM svv_datashares
WHERE share_name LIKE 'tickit_datashare%'
AND source_database = 'dev';

  share_owner | source_database | share_type  | is_publicaccessible
--------------+-----------------+-------------+---------------------
     100      |      dev        |   OUTBOUND  |        True
(1 rows)
```

Contoh berikut mengembalikan output untuk SVV_DATASHARES untuk berbagi data keluar.

```
SELECT share_name, share_owner, btrim(source_database), btrim(consumer_database),
 share_type, is_publicaccessible, share_acl, btrim(producer_account),
 btrim(producer_namespace), btrim(managed_by) FROM svv_datashares WHERE share_type =
 'OUTBOUND';

   share_name   | share_owner | source_database | consumer_database | share_type |
 is_publicaccessible | share_acl | producer_account|         producer_namespace
   | managed_by
----------------+-------------+-----------------+-------------------
+------------+---------------------+-----------+-----------------
+--------------------------------------+------------
    salesshare  |      1      |       dev       |                   |  OUTBOUND  |
   True         |           |   123456789012  | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |

 marketingshare |      1      |       dev       |                   |  OUTBOUND  |
   True         |           |   123456789012  | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
```

Contoh berikut mengembalikan output untuk SVV_DATASHARES untuk inbound datashares.

```
SELECT share_name, share_owner, btrim(source_database), btrim(consumer_database),
 share_type, is_publicaccessible, share_acl, btrim(producer_account),
 btrim(producer_namespace), btrim(managed_by) FROM svv_datashares WHERE share_type =
 'INBOUND';

  share_name    | share_owner | source_database | consumer_database | share_type |
 is_publicaccessible | share_acl | producer_account |         producer_namespace
    | managed_by
```

```
---------------+-------------+-----------------+-------------------
+------------+--------------------+-----------+-----------------
+--------------------------------------+------------
  salesshare    |             |                 |                   |  INBOUND   |
   False         |           |  123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
 |
 marketingshare |             |                 |                   |  INBOUND   |
  False         |           |  123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
 ADX
```

SVV_DATASHARE_KONSUMEN

Gunakan SVV_DATASHARE_CONSUMERS untuk melihat daftar konsumen untuk datashare yang dibuat pada cluster.

SVV_DATASHARE_CONSUMERS terlihat oleh pengguna berikut:

- Superuser
- Pemilik Datashare
- Pengguna dengan izin ALTER atau USAGE pada datashare

Pengguna lain tidak dapat melihat baris apa pun. Untuk informasi tentang izin ALTER dan USAGE, lihatHIBAH.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_berbagi	bangkai (128)	Nama datashare.
akun_konsumen	varchar(16)	ID akun untuk konsumen datashare.
ruang nama konsumen	varchar(64)	Pengidentifikasi cluster unik dari cluster konsumen datashare.
berbagi_tanggal	stempel waktu tanpa zona waktu	Tanggal datashare dibagikan.

Contoh kueri

Contoh berikut mengembalikan output untuk SVV_DATASHARE_CONSUMERS.

```
SELECT count(*)
FROM svv_datashare_consumers
WHERE share_name LIKE 'tickit_datashare%';

1
```

SVV_DATASHARE_OBJECT

Gunakan SVV_DATASHARE_OBJECTS untuk melihat daftar objek di semua data share yang dibuat di cluster atau dibagikan dengan cluster.

SVV_DATASHARE_OBJECTS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang melihat daftar berbagi data, lihat SVV_DATASHARES.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
tipe_bagikan	Washer (8)	Jenis datashare yang ditentukan. Nilai yang mungkin adalah OUTBOUND dan INBOUND.
nama_berbagi	bangkai (128)	Nama datashare.
tipe_objek	varchar(64)	Jenis objek tertentu. Nilai yang mungkin adalah skema, tabel, tampilan, tampilan pengikatan akhir, tampilan terwujud, dan fungsi.
nama_objek	bangkai (512)	Nama objek. Nama objek meluas untuk menyertak

Nama kolom	Jenis data	Deskripsi
		an nama skema, seperti schema1.t1.
akun_produsen	varchar(16)	ID untuk akun produsen datashare.
ruang nama produsen	varchar(64)	Pengidentifikasi cluster unik untuk cluster produsen datashare.
termasukan_baru	boolean	Properti yang menentukan apakah akan menambahk an tabel masa depan, tampilan, atau fungsi yang ditentukan pengguna SQL (UDF) yang dibuat dalam skema yang ditentukan ke datashare. Parameter ini hanya relevan untuk berbagi data OUTBOUND dan hanya untuk tipe skema di datashare.

Contoh kueri

Contoh berikut mengembalikan output untuk SVV_DATASHARE_OBJECTS.

```
SELECT share_type,
    btrim(share_name)::varchar(16) AS share_name,
    object_type,
    object_name
FROM svv_datashare_objects
WHERE share_name LIKE 'tickit_datashare%'
AND object_name LIKE '%tickit%'
ORDER BY object_name
LIMIT 5;

 share_type |     share_name     | object_type |          object_name
```

```
------------+--------------------+-------------+---------------------------------
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_category_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_date_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_event_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_listing_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_sales_redshift
```

```
SELECT * FROM SVV_DATASHARE_OBJECTS WHERE share_name like 'sales%';

share_type | share_name | object_type | object_name  | producer_account |
 producer_namespace          | include_new
-----------+------------+-------------+--------------+-----------------
+--------------------------------------+-------------
 OUTBOUND  | salesshare | schema      | public       | 123456789012     |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |      t
 OUTBOUND  | salesshare | table       | public.sales | 123456789012     |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
```

SVV_DEFAULT_HAK ISTIMEWA

Gunakan SVV_DEFAULT_PRIVILEGES untuk melihat hak istimewa default yang dapat diakses pengguna di cluster Amazon Redshift.

SVV_DEFAULT_PRIVILEGES dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat izin default yang diberikan kepada mereka.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW GRANTS untuk penemuan izin. SHOW GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
schema_name	text	Nama skema.
tipe_objek	text	Jenis objek. Nilai yang mungkin adalah RELATION, FUNCTION, atau PROCEDURE.
id pemiliknya	integer	ID pemilik. Nilai yang mungkin adalah ID pengguna.
nama_pemilik	text	Nama pemiliknya.
tipe_pemilik	text	Jenis pemilik. Nilai yang mungkin adalah pengguna.
tipe hak istimewa	text	Jenis hak istimewa. Nilai yang mungkin adalah INSERT, SELECT, UPDATE, DELETE, RULE, REFERENSI TRIGGER, DROP, dan EXECUTE.
ID penerima hibah	integer	ID penerima hibah. Nilai yang mungkin adalah ID pengguna, ID peran, dan ID grup.
tipe penerima hibah	text	Jenis penerima hibah. Nilai yang mungkin adalah pengguna, peran, dan publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe grup.

Contoh kueri

Contoh berikut mengembalikan output untuk SVV_DEFAULT_PRIVILEGES.

```
SELECT * from svv_default_privileges;
```

```
 schema_name |    object_type    | owner_id | owner_name | owner_type | privilege_type
 | grantee_id | grantee_name | grantee_type | admin_option
-------------+-------------------+--------- +------------+------------+---------------
+------------+--------------+--------------+-------------+
   public    |     RELATION      |    106   |     u1     |    user    |     UPDATE
 |     107    |      u2      |     user     |      f      |
   public    |     RELATION      |    106   |     u1     |    user    |     SELECT
 |     107    |      u2      |     user     |      f      |
```

SVV_DISKUSI

Amazon Redshift membuat tampilan sistem SVV_DISKUSAGE dengan menggabungkan tabel STV_TBL_PERM dan STV_BLOCKLIST. Tampilan SVV_DISKUSAGE berisi informasi tentang alokasi data untuk tabel dalam database.

Gunakan kueri agregat dengan SVV_DISKUSAGE, seperti yang ditunjukkan contoh berikut, untuk menentukan jumlah blok disk yang dialokasikan per database, tabel, irisan, atau kolom. Setiap blok data menggunakan 1 MB. Anda juga dapat menggunakan STV_PARTISI untuk melihat informasi ringkasan tentang pemanfaatan disk.

SVV_DISKUSAGE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> Tampilan ini hanya tersedia saat menanyakan klaster yang disediakan.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
db_id	integer	ID Basis Data.
name	karakter (72)	Nama tabel.
mengiris	integer	Potongan data dialokasikan ke tabel.

Nama kolom	Jenis data	Deskripsi
kol	integer	Zero-based indeks untuk kolom. Setiap tabel yang Anda buat memiliki tiga kolom tersembunyi yang ditambahkan padanya: INSERT_XID, DELETE_XID, dan ROW_ID (OID). Tabel dengan 3 kolom yang ditentukan pengguna berisi 6 kolom aktual, dan kolom yang ditentukan pengguna diberi nomor internal sebagai 0, 1, dan 2. Kolom INSERT_XID, DELETE_XID, dan ROW_ID diberi nomor 3, 4, dan 5, masing-masing, dalam contoh ini.
tbl	integer	ID tabel.
Blocknum	integer	ID untuk blok data.
nilai_angka	integer	Jumlah nilai yang terkandung di blok.
minvalue	bigint	Nilai minimum yang terkandung di blok.
nilai maksimum	bigint	Nilai maksimum yang terkandung di blok.
sb_pos	integer	Pengidentifikasi internal untuk posisi blok super pada disk.
disematkan	integer	Apakah blok disematkan ke dalam memori sebagai bagian dari pra-pemuatan. 0 = salah; 1 = benar. Defaultnya false.
on_disk	integer	Apakah blok disimpan secara otomatis pada disk atau tidak. 0 = salah; 1 = benar. Defaultnya false.
dimodifikasi	integer	Apakah blok telah dimodifikasi atau tidak. 0 = salah; 1 = benar. Defaultnya false.
hdr_dimodifikasi	integer	Apakah header blok telah dimodifikasi atau tidak. 0 = false; 1 = true. Defaultnya false.
tidak disortir	integer	Apakah blok tidak disortir atau tidak. 0 = salah; 1 = benar. Defaultnya benar.
batu nisan	integer	Untuk penggunaan internal.

Nama kolom	Jenis data	Deskripsi
preferred_diskno	integer	Nomor disk yang harus diaktifkan blok, kecuali disk telah gagal. Setelah disk diperbaiki, blok akan pindah kembali ke disk ini.
sementara	integer	Apakah blok berisi data sementara atau tidak, seperti dari tabel sementara atau hasil kueri perantara. 0 = false; 1 = true. Defaultnya false.
blok baru	integer	Menunjukkan apakah blok baru (benar) atau tidak pernah diikat ke disk (false). 0 = false; 1 = true.

Kueri Sampel

SVV_DISKUSAGE berisi satu baris per blok disk yang dialokasikan, sehingga kueri yang memilih semua baris berpotensi mengembalikan sejumlah besar baris. Sebaiknya gunakan hanya kueri agregat dengan SVV_DISKUSAGE.

Kembalikan jumlah blok tertinggi yang pernah dialokasikan ke kolom 6 di tabel USERS (kolom EMAIL):

```
select db_id, trim(name) as tablename, max(blocknum)
from svv_diskusage
where name='users' and col=6
group by db_id, name;

db_id  | tablename | max
--------+-----------+-----
175857 | users     |   2
(1 row)
```

Kueri berikut mengembalikan hasil yang serupa untuk semua kolom dalam tabel 10 kolom besar yang disebut SALESNEW. (Tiga baris terakhir, untuk kolom 10 hingga 12, adalah untuk kolom metadata tersembunyi.)

```
select db_id, trim(name) as tablename, col, tbl, max(blocknum)
from svv_diskusage
where name='salesnew'
```

```
group by db_id, name, col, tbl
order by db_id, name, col, tbl;

db_id  | tablename  | col |  tbl   | max
--------+------------+-----+--------+-----
175857 | salesnew   |   0 | 187605 | 154
175857 | salesnew   |   1 | 187605 | 154
175857 | salesnew   |   2 | 187605 | 154
175857 | salesnew   |   3 | 187605 | 154
175857 | salesnew   |   4 | 187605 | 154
175857 | salesnew   |   5 | 187605 |  79
175857 | salesnew   |   6 | 187605 |  79
175857 | salesnew   |   7 | 187605 | 302
175857 | salesnew   |   8 | 187605 | 302
175857 | salesnew   |   9 | 187605 | 302
175857 | salesnew   |  10 | 187605 |   3
175857 | salesnew   |  11 | 187605 |   2
175857 | salesnew   |  12 | 187605 | 296
(13 rows)
```

SVV_KOLON_EKSTERNAL_

Gunakan SVV_EXTERNAL_COLUMNS untuk melihat detail kolom di tabel eksternal. Gunakan SVV_EXTERNAL_COLUMNS juga untuk kueri lintas database untuk melihat detail pada semua kolom dari tabel pada database yang tidak terhubung yang dapat diakses pengguna.

SVV_EXTERNAL_COLUMNS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW COLUMNS untuk penemuan kolom. SHOW COLUMNS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
redshift_nama_database	text	Nama database Amazon Redshift lokal.
nama skema	text	Nama skema eksternal Amazon Redshift untuk tabel eksternal.
nama tabel	text	Nama tabel eksternal.
nama kolom	text	Nama kolom.
tipe eksternal	text	Jenis data kolom.
Columnnum	integer	Nomor kolom eksternal, mulai dari 1.
bagian_kunci	integer	Jika kolom adalah kunci partisi, urutan kuncinya. Jika kolom bukan partisi, nilainya adalah**0**.
is_nullable	text	Mendefinisikan apakah kolom dapat dibatalkan atau tidak. Beberapa nilai adalah`true`,`false`, atau "" string kosong yang tidak mewakili informasi.

SVV_EXTERNAL_DATABASE

Gunakan SVV_EXTERNAL_DATABASES untuk melihat detail untuk database eksternal.

SVV_EXTERNAL_DATABASES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
eskind	integer	Jenis katalog eksternal untuk database; **1** menunjukkan katalog data, **2** menunjukkan metastor Hive.
esopsi	text	Rincian katalog tempat basis data berada.
nama database	text	Nama database di katalog eksternal.
lokasi	text	Lokasi database.
parameters	text	Parameter database.

SVV_PARTISI_EKSTERNAL

Gunakan SVV_EXTERNAL_PARTITIONS untuk melihat detail partisi di tabel eksternal.

SVV_EXTERNAL_PARTITIONS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama skema	text	Nama skema eksternal Amazon Redshift untuk tabel eksternal dengan partisi yang ditentukan.
nama tabel	text	Nama tabel eksternal.
values	text	Nilai untuk partisi.

Nama kolom	Jenis data	Deskripsi
lokasi	text	Lokasi partisi. Ukuran kolom dibatasi hingga 128 karakter. Nilai yang lebih panjang dipotong.
masukan_format	text	Format input.
format_keluaran	text	Format keluaran.
serialisasi_lib	text	Perpustakaan serialisasi.
serde_parameter	text	SerDe parameter.
terkompresi	integer	Nilai yang menunjukkan apakah partisi dikompresi; **1** menunjukkan terkompresi, **0** menunjukkan tidak terkompresi.
parameters	text	Properti partisi.

SVV_EXTERNAL_SCHEMAS

Gunakan SVV_EXTERNAL_SCHEMAS untuk melihat informasi tentang skema eksternal. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

SVV_EXTERNAL_SCHEMAS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW SCHEMAS untuk penemuan skema. SHOW SCHEMAS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui

saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
esoid	oid	ID skema eksternal.
eskind	smallint	Jenis katalog eksternal untuk skema eksternal: 1 menunjukkan katalog data, 2 menunjukkan metastore Hive, 3 menunjukkan kueri federasi ke Aurora PostgreSQL atau Amazon RDS PostgreSQL, 4 menunjukkan skema untuk database Amazon Redshift lokal, 5 menunjukkan skema untuk database Amazon Redshift jarak jauh, 6 menunjukkan skema untuk tabel sistem, 8 menunjukkan skema untuk database MySQL jarak jauh, 9 menunjukkan skema untuk Amazon Kinesis aliran data, dan 10 menunjukkan Amazon Managed Streaming untuk aliran data Apache Kafka.
nama skema	name	Nama skema eksternal.
esowner	integer	ID pengguna pemilik skema eksternal.
nama database	text	Nama database eksternal.
esopsi	text	Opsi skema eksternal.

Contoh

Contoh berikut menunjukkan detail untuk skema eksternal.

```
select * from svv_external_schemas;
```

```
esoid  | eskind | schemaname | esowner | databasename | esoptions

-------+--------+------------+---------+--------------
+-------------------------------------------------------------
100133 |      1 | spectrum   |     100 | redshift     |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

SVV_EXTERNAL_TABEL

Gunakan SVV_EXTERNAL_TABLES untuk melihat detail tabel eksternal; untuk informasi lebih lanjut, lihat. BUAT SKEMA EKSTERNAL Gunakan SVV_EXTERNAL_TABLES juga untuk kueri lintas database untuk melihat metadata pada semua tabel pada database yang tidak terhubung yang dapat diakses pengguna.

SVV_EXTERNAL_TABLES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW TABLES untuk penemuan tabel. SHOW TABLES bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
redshift_nama_database	text	Nama database Amazon Redshift lokal.
nama skema	text	Nama skema eksternal Amazon Redshift untuk tabel eksternal.

Nama kolom	Jenis data	Deskripsi
nama tabelnya	text	Nama tabel eksternal.
tipe tablet	text	Jenis meja. Beberapa nilai adalah TABLE, VIEW, MATERIALIZED VIEW, atau "" string kosong yang tidak mewakili informasi.
lokasi	text	Lokasi meja.
masukan_format	text	Format masukan
format_keluaran	text	Format keluaran.
serialisasi_lib	text	Perpustakaan serialisasi.
serde_parameter	text	SerDe parameter.
terkompresi	integer	Nilai yang menunjukkan apakah tabel dikompresi; **1** menunjukkan terkompresi, **0** menunjukkan tidak terkompresi.
parameters	text	Properti tabel.

Contoh

Contoh berikut menunjukkan detail svv_external_tables dengan predikat pada skema eksternal yang digunakan oleh kueri federasi.

```
select schemaname, tablename from svv_external_tables where schemaname = 'apg_tpch';
schemaname  | tablename
------------+-----------
apg_tpch    | customer
apg_tpch    | lineitem
apg_tpch    | nation
apg_tpch    | orders
```

```
apg_tpch    | part
apg_tpch    | partsupp
apg_tpch    | region
apg_tpch    | supplier
(8 rows)
```

SVV_FUNGSI_HAK ISTIMEWA

Gunakan SVV_FUNCTION_PRIVILEGES untuk melihat izin fungsi yang secara eksplisit diberikan kepada pengguna, peran, dan grup dalam database saat ini.

SVV_FUNCTION_PRIVILEGES terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW GRANTS untuk penemuan izin. SHOW GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_ruang_nama	text	Nama namespace di mana fungsi tertentu ada.
function_name	text	Nama fungsi.
tipe_argumen	text	String yang mewakili jenis argumen input untuk suatu fungsi.

Nama kolom	Jenis data	Deskripsi
keistimewa_tipe	text	Jenis izin. Nilai yang mungkin adalah EXECUTE.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_FUNCTION_PRIVILEGES.

```
SELECT
 namespace_name,function_name,argument_types,privilege_type,identity_name,identity_type,admin_o
 FROM svv_function_privileges
WHERE identity_name IN ('role1', 'reguser');

 namespace_name | function_name |       argument_types       | privilege_type |
 identity_name | identity_type | admin_option
----------------+---------------+----------------------------+----------------
+----------------+---------------+-------------
    public      | test_func1    | integer                    |    EXECUTE     |
 role1     |     role      |  False
    public      | test_func2    | integer, character varying |    EXECUTE     |
 reguser    |     user      |  False
```

SVV_GEOGRAPHY_COLUMN_

Gunakan SVV_GEOGRAPHY_COLUMNS untuk melihat daftar kolom GEOGRAFI di gudang data Anda. Daftar kolom ini mencakup kolom dari datashares.

SVV_GEOGRAPHY_COLUMNS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
f_table_catalog	bangkai (128)	Nama database tempat tabel dengan kolom GEOGRAFI ada.
f_table_skema	bangkai (128)	Nama skema tempat tabel dengan kolom GEOGRAFI ada.
f_nama_tabel	bangkai (128)	Nama tabel tempat kolom GEOGRAFI ada.
f_geography_kolom	bangkai (128)	Nama kolom GEOGRAFI.
dimensi koordinat	integer	Jumlah dimensi data GEOGRAFI.
menjijikkan	integer	Pengidentifikasi sistem referensi spasial (SRID) dari data GEOGRAFI.
jenis	bangkai (128)	Nama tipe data geografi spasial.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_GEOGRAPHY_COLUMNS.

```
SELECT * FROM svv_geography_columns;

f_table_catalog  | f_table_schema  | f_table_name  | f_geography_column  |
 coord_dimension | srid |  type
-----------------+-----------------+--------------+---------------------
+-----------------+------+-------------
```

```
dev              | public          | spatial_test  | test_geography      | 2
    | 0    | GEOGRAPHY
```

SVV_GEOMETRI_KOLOM

Gunakan SVV_GEOMETRY_COLUMNS untuk melihat daftar kolom GEOMETRY di gudang data Anda. Daftar kolom ini mencakup kolom dari datashares.

SVV_GEOMETRY_COLUMNS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
f_table_catalog	bangkai (128)	Nama database tempat tabel dengan kolom GEOMETRY ada.
f_table_skema	bangkai (128)	Nama skema tempat tabel dengan kolom GEOMETRY ada.
f_nama_tabel	bangkai (128)	Nama tabel di mana kolom GEOMETRY ada.
f_geometri_kolom	bangkai (128)	Nama kolom GEOMETRY.
dimensi koordinat	integer	Jumlah dimensi data GEOMETRI.
menjijikkan	integer	Pengidentifikasi sistem referensi spasial (SRID) dari kolom GEOMETRY.
jenis	bangkai (128)	Nama tipe geometri spasial.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_GEOMETRY_COLUMNS.

```
SELECT * FROM svv_geometry_columns;

f_table_catalog  | f_table_schema  | f_table_name  | f_geometry_column   |
 coord_dimension | srid |  type
-----------------+-----------------+--------------+--------------------
+-----------------+------+-------------
dev              | public          | accomodations | shape               | 2
    | 0    | GEOMETRY
dev              | public          | zipcode       | wkb_geometry        | 2
    | 0    | GEOMETRY
```

SVV_IAM_HAK ISTIMEWA

Gunakan SVV_IAM_PRIVILEGES untuk melihat hak istimewa IAM yang diberikan secara eksplisit pada pengguna, peran, dan grup.

SVV_IAM_PRIVILEGES dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat entri yang dapat mereka akses.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
iam_arn	text	Nama namespace.
tipe_perintah	text	Jenis hak istimewa. Nilai yang mungkin adalah COPY, UNLOAD, CREATE MODEL, atau EXTERNAL FUNCTION.
identitas_id	integer	Identitas ID. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.

Nama kolom	Jenis data	Deskripsi
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.

Kueri Sampel

Contoh berikut menunjukkan hasil SVV_IAM_PRIVILEGES.

```
SELECT * from SVV_IAM_PRIVILEGES ORDER BY IDENTITY_ID;
       iam_arn        | command_type | identity_id | identity_name | identity_type
----------------------+--------------+-------------+---------------+---------------
 default-aws-iam-role | COPY         |           0 | public        | public
 default-aws-iam-role | UNLOAD       |           0 | public        | public
 default-aws-iam-role | CREATE MODEL |           0 | public        | public
 default-aws-iam-role | EXFUNC       |           0 | public        | public
 default-aws-iam-role | COPY         |         106 | u1            | user
 default-aws-iam-role | UNLOAD       |         106 | u1            | user
 default-aws-iam-role | CREATE MODEL |         106 | u1            | user
 default-aws-iam-role | EXFUNC       |         106 | u1            | user
 default-aws-iam-role | COPY         |      118413 | r1            | role
 default-aws-iam-role | UNLOAD       |      118413 | r1            | role
 default-aws-iam-role | CREATE MODEL |      118413 | r1            | role
 default-aws-iam-role | EXFUNC       |      118413 | r1            | role
(12 rows)
```

SVV_IDENTITY_PROVIDER

Tampilan SVV_IDENTITY_PROVIDERS mengembalikan nama dan properti tambahan untuk penyedia identitas. Untuk informasi selengkapnya tentang cara membuat penyedia identitas, lihatBUAT PENYEDIA IDENTITAS.

SVV_IDENTITY_PROVIDERS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
uid	integer	ID unik dari penyedia identitas terdaftar.
name	text	Nama penyedia identitas.
jenis	text	Jenis penyedia identitas.
instancid	text	Pembeda unik antara instance dari tipe yang sama.
namespc	text	Awalan namespace dari penyedia identitas.
params	text	Objek JSON dengan parameter untuk penyedia identitas.
diaktifkan	bool	Menunjukkan jika penyedia identitas diaktifkan.

Kueri Sampel

Untuk melihat properti penyedia identitas, jalankan kueri seperti berikut setelah membuat penyedia identitas.

```
SELECT name, type, instanceid, namespc, params, enabled
FROM svv_identity_providers
ORDER BY 1;
```

Output sampel mencakup deskripsi param untuk Microsoft Entra ID (sebelumnya Azure AD):

```
      name       | type  |             instanceid             | namespc |

                                      params

                    | enabled
```

```
------------------+-------+--------------------------------------+---------
+-----------------------------------------------------------------------------------------------------
+---------
 rs5517_azure_idp | azure | e40d4bb2-7670-44ae-bfb8-5db013221d73 | abc     |
 {"issuer":"https://login.microsoftonline.com/e40d4bb2-7670-44ae-bfb8-5db013221d73/
v2.0", "client_id":"871c010f-5e61-4fb1-83ac-98610a7e9110", "client_secret":,
 "audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift", "https://
analysis.windows.net/powerbi/connector/AWSRDS"]} | t
(1 row)
```

Output sampel mencakup deskripsi param untuk: AWS IAM Identity Center

```
   name   |  type  |                                    instanceid
                |     namespc      |
                        params
                    | enabled
----------+--------
+-----------------------------------------------------------------------------------
+------------------
+-----------------------------------------------------------------------------------------------------
+---------
 idc_name | awsidc | arn:aws:sso::123456789012:application/ssoins-12345f67fe123d4/
apl-a0b0a12dc123b1a4 |   idc_namespc   | {"iam_role":"arn:aws:iam::123456789012:role/
MyRedshiftRole","instance_arn":"arn:aws:sso:::instance/
ssoins-12345f67fe123d4","is_lakehouse_app":"true"}        | t
(1 row)
```

SVV_INTEGRASI

SVV_INTEGRATION menampilkan detail tentang konfigurasi integrasi.

SVV_INTEGRATION hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang integrasi Zero-ETL, lihat Zero-ETL integrasi.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
integration_id	karakter (128)	Identifier yang terkait dengan integrasi.

Nama kolom	Jenis data	Deskripsi
Target_database	karakter (128)	Basis data di Amazon Redshift yang menerima data integrasi.
sumber	karakter (128)	Sumber data untuk integrasi. Jenis yang mungkin termasuk`MySQL`,`PostgreSQL` , dan`S3_EVENT_NOTIFICATIONS` .
status	karakter (128)	Keadaan integrasi. Nilai yang mungkin termasuk `PendingDbConnectState` , `SchemaDiscoveryState` , `CdcRefreshState` , dan `ErrorState` .
kelambatan saat ini	bigint	Waktu jeda saat ini (milidetik) antara sumber dan tujuan integrasi.
titik pemeriksaan terreplikat_terakhir	karakter (128)	Pos pemeriksaan terakhir yang direplikasi.
total_table_direplikasi	integer	Jumlah total tabel saat ini dalam keadaan direplikasi.
total_table_gagal	integer	Jumlah total tabel saat ini dalam keadaan gagal.
waktu penciptaan	timestamp	Waktu (UTC) saat integrasi dibuat. Ini didefinisikan sebagai waktu ketika database target dibuat dari integrasi.
Interval Refresh	integer	Perkiraan interval waktu, dalam detik, untuk menyegarkan data dari sumber Zero-ETL ke database target.
sumber_database	karakter (128)	Nama database sumber.
is_history_mode	boolean	`TRUE`Nilai menunjukkan bahwa mode riwayat aktif. A `FALSE` menunjukkan bahwa mode riwayat mati.

Nama kolom	Jenis data	Deskripsi
waktu_perubahan_terbaru_terdeteksi	timestamp	Waktu (UTC) ketika perubahan sumber terbaru untuk database integrasi ini dipentaskan dalam antrian replikasi.
terbaru_aplikasi_waktu_perubahan	timestamp	Waktu (UTC) dari penyerapan terakhir yang berhasil di Amazon Redshift untuk database target ini.
auto_remediasi	boolean	`TRUE`Nilai menunjukkan bahwa remediasi otomatis baris duplikat diaktifkan. `FALSE`Nilai menunjukkan bahwa itu dinonaktifkan.
waktu_sumber_kiriman_terbaru_sumber_kirim	timestamp	Waktu (UTC), pada jam database sumber, hingga transaksi sumber telah dikirim untuk integrasi ini.

Kueri Sampel

Perintah SQL berikut menampilkan integrasi yang didefinisikan saat ini.

```
select * from svv_integration;

           integration_id              | target_database | source |      state
   | current_lag |      last_replicated_checkpoint     | total_tables_replicated |
 total_tables_failed |       creation_time       |  refresh_interval  | source_database
 | is_history_mode
--------------------------------------+-----------------+--------+----------------
+-------------+--------------------------------------+-------------------------
+---------------------+--------------------------+--------------------
+-----------------+-----------------
  99108e72-1cfd-414f-8cc0-0216acefac77 |     perfdb      |  MySQL | CdcRefreshState |
  56606106  | {"txn_seq":9834,"txn_id":126597515} |            152          |
 0         | 2023-09-19 21:05:27.520299|      720           + mysourceetl     | f
```

SVV_INTEGRASI_TABEL_PEMETAAN

SVV_INTEGRATION_TABLE_MAPPING menampilkan pemetaan database sumber, skema, tabel, kolom, dan tipe data ke target ketika nilai pengenal bidang tersebut berbeda.

> **Note**
>
> Tampilan ini hanya diisi untuk jenis integrasi Zero-ETL berikut:
>
> - AWS Glue aplikasi pihak ketiga untuk Amazon SageMaker Lakehouse
> - Amazon DynamoDB para Amazon Lakehouse SageMaker
>
> Untuk informasi selengkapnya, lihat Zero-ETL integrasi di Panduan Peng AWS Glue embang.

Transformasi nilai pengidentifikasi dari sumber ke target mengikuti aturan berikut:

- Huruf besar diubah menjadi huruf kecil.
- Karakter yang bukan huruf kecil, digit, atau garis bawah (_) diubah menjadi garis bawah (_).
- Jika ada konflik dengan nilai pengenal yang ada, maka Universally Unique Identifier (UUID) ditambahkan ke pengidentifikasi baru.
- Jika nilai pengidentifikasi sumber adalah kata kunci Amazon Redshift, maka akhiran `_redshift` ditambahkan ke pengidentifikasi baru.

Setelah transformasi, karakter harus berupa huruf kecil, digit, atau garis bawah (_) dan cocok dengan pola regex. `[a-z0-9_]` Contoh berikut menunjukkan aturan konversi:

Sumber	Target	Catatan
foo	foo	Tidak ada transformasi
Bar	bar	
FooBar	foobar	
foo1	foo1	Tidak ada transformasi
foo_1	foo_1	Tidak ada transformasi
Bar @1	bar_1	

Sumber	Target	Catatan
foo_bar@	foobar_	
kasus	case_redshift	

SVV_INTEGRATION_TABLE_MAPPING terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang integrasi Zero-ETL, lihat Zero-ETL integrasi di Panduan Manajemen Amazon Redshift.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
integration_id	karakter (128)	Identifier yang terkait dengan integrasi.
sumber_database	karakter (128)	Nama database di sumber.
Target_database	karakter (128)	Basis data di Amazon Redshift yang menerima data integrasi.
nama_skema sumber	karakter (128)	Nama skema di sumber.
nama_skema_target	karakter (128)	Skema di Amazon Redshift yang menerima data integrasi.
nama_tabel_sumber	karakter (128)	Nama tabel di sumber.
nama_tabel_target	karakter (128)	Tabel di Amazon Redshift yang menerima data integrasi.

Nama kolom	Jenis data	Deskripsi
nama_kolom sumber	karakter (128)	Nama kolom di sumber.
nama_kolom target	karakter (128)	Kolom di Amazon Redshift yang menerima data integrasi.
tipe_data sumber	karakter (128)	Jenis data kolom di sumber.
tipe_data target	karakter (128)	Jenis data kolom di Amazon Redshift yang menerima data integrasi.

Kueri Sampel

Perintah SQL berikut menampilkan pemetaan nilai metadata dari sumber ke target.

```
select * from svv_integration_table_mapping;

           integration_id              | source_database | target_database |
 source_schema_name | target_schema_name | source_table_name | target_table_name |
-------------------------------------+-----------------+-----------------
+--------------------+--------------------+--------------------------------------+
  99108e72-1cfd-414f-8cc0-0216acefac77 |     mydatabase  |  mydatabase     |  myschema
         | myschema           | Mytable           | mytable           |


                                      | source_column_name | target_column_name |
 source_data_type | target_data_type |
                                     +--------------------+-------------------
+------------------+------------------+
                                      | Mycolumnname       | mycolumnname       |
 Mydatatype       | mydatatype       |
```

SVV_INTEGRASI_TABLE_STATE

SVV_INTEGRATION_TABLE_STATE menampilkan detail tentang informasi integrasi tingkat tabel.

SVV_INTEGRATION_TABLE_STATE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya, lihat Zero-ETL integrasi.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
integration_id	karakter (128)	Identifier yang terkait dengan integrasi.
Target_database	karakter (128)	Nama database Amazon Redshift.
schema_name	karakter (128)	Nama skema Amazon Redshift.
table_name	karakter (128)	Nama tabel.
tabel_status	karakter (128)	Keadaan meja. Nilai yang mungkin adalah`Synced`,`Failed`,`Deleted`,`ResyncRequired` , `ResyncInitiated` dan`DroppedSource` . `DroppedSource` Status menunjukkan bahwa sumber tabel mode riwayat dijatuhkan di sumbernya.
table_last_replikat_checkpoint	karakter (128)	Koordinat log yang disinkronkan saat ini.
akal budi	karakter (256)	Alasan transisi negara terakhir. Alasan umum dapat berupa tipe data yang tidak didukung dalam tabel, tabel tidak memiliki kunci utama. Untuk mempelajari selengkapnya tentang cara memecahkan masalah umum, lihat Memecahkan Masalah integrasi Zero-ETL di Amazon Redshift.
stempel waktu terakhir_diperbarui	stempel waktu tanpa zona waktu	Waktu (UTC) saat tabel terakhir diperbarui.
tabel_baris	bigint	Jumlah total baris dalam tabel.

Nama kolom	Jenis data	Deskripsi
ukuran tabel	bigint	Ukuran tabel dalam megabyte (MB).
is_history_mode	boolean	TRUENilai menunjukkan bahwa mode riwayat aktif. A FALSE menunjukkan bahwa mode riwayat mati.

Kueri Sampel

Perintah SQL berikut menampilkan kolom log integrasi.

```
select * from svv_integration_table_state;

          integration_id              | target_database | schema_name |     table_name
   | table_state  |table_last_replicated_checkpoint | reason | last_updated_timestamp
   |table_rows  | table_size | is_history_mode
-------------------------------------+-----------------+-------------
+------------------+-------------+---------------------------------+--------
+---------------------------+-----------+-----------+-----------------
 4798e675-8f9f-4686-b05f-92c538e19629 |  sample_test2   |    sample   |
 SampleTestChannel |    Synced    |   {"txn_seq":3,"txn_id":3122}   |        |
 2023-05-12 12:40:30.656625 | 2          |   16       | f
```

SVV_KOLUMN_INTERLEAVED_

Gunakan tampilan SVV_INTERLEAVED_COLUMNS untuk membantu menentukan apakah tabel yang menggunakan kunci pengurutan yang disisipkan harus diindeks ulang menggunakan. VACUUM REINDEX Untuk informasi selengkapnya tentang cara menentukan seberapa sering menjalankan VACUUM dan kapan menjalankan VACUUM REINDEX, lihatMeminimalkan waktu vakum.

SVV_INTERLEAVED_COLUMNS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
tbl	integer	ID tabel.
kol	integer	Zero-based indeks untuk kolom.
selisip_miring	numerik (19,2)	Rasio yang menunjukkan jumlah kemiringan yang ada dalam kolom kunci pengurutan yang disisipkan untuk tabel. Nilai 1,00 menunjukkan tidak ada kemiringan, dan nilai yang lebih besar menunjukkan lebih banyak kemiringan. Tabel dengan kemiringan besar harus diindeks ulang dengan perintah VACUUM REINDEX.
indeks ulang terakhir	timestar	Waktu ketika VACUUM REINDEX terakhir dijalankan untuk tabel yang ditentukan. Nilai ini adalah NULL jika tabel belum pernah diindeks ulang atau jika tabel log sistem yang mendasarinya, STL_VACUUM, telah diputar sejak indeks ulang terakhir.

Kueri Sampel

Untuk mengidentifikasi tabel yang mungkin perlu diindeks ulang, jalankan kueri berikut.

```
select tbl as tbl_id, stv_tbl_perm.name as table_name,
col, interleaved_skew, last_reindex
from svv_interleaved_columns, stv_tbl_perm
where svv_interleaved_columns.tbl = stv_tbl_perm.id
and interleaved_skew is not null;

 tbl_id | table_name | col | interleaved_skew | last_reindex
--------+------------+-----+------------------+---------------------
 100068 | lineorder  |   0 |             3.65 | 2015-04-22 22:05:45
 100068 | lineorder  |   1 |             2.65 | 2015-04-22 22:05:45
 100072 | customer   |   0 |             1.65 | 2015-04-22 22:05:45
 100072 | lineorder  |   1 |             1.00 | 2015-04-22 22:05:45
(4 rows)
```

SVV_LANGUAGE_KEISTIMEWAAN

Gunakan SVV_LANGUAGE_PRIVILEGES untuk melihat izin bahasa yang secara eksplisit diberikan kepada pengguna, peran, dan grup dalam database saat ini.

SVV_LANGUAGE_PRIVILEGES dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_bahasa	text	Nama bahasanya.
tipe hak istimewa	text	Jenis izin. Nilai yang mungkin adalah PEMAKAIAN.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_LANGUAGE_PRIVILEGES.

```
SELECT language_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_language_privileges
WHERE identity_name IN ('role1', 'reguser');

 language_name | privilege_type | identity_name | identity_type | admin_option
---------------+----------------+---------------+---------------+--------------
   exfunc      |     USAGE      |    reguser    |     user      |    False
   exfunc      |     USAGE      |     role1     |     role      |    False
   plpythonu   |     USAGE      |    reguser    |     user      |    False
```

SVV_MASKING_POLICY

Gunakan SVV_MASKING_POLICY untuk melihat semua kebijakan masking yang dibuat di cluster.

Hanya pengguna super dan pengguna dengan `sys:secadmin` peran yang dapat melihat SVV_MASKING_POLICY. Pengguna biasa akan melihat 0 baris.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
politik_database	text	Nama database tempat kebijakan masking dibuat.
nama_kebijakan	text	Nama kebijakan masking.
masukan_kolom	text	Atribut yang disediakan dalam klausa WITH pernyataan CREATE POLICY.
Ekspresi kebijakan	text	Ekspresi masking yang digunakan dalam kebijakan.
Policy_dimodifikasi oleh	text	Nama pengguna yang terakhir mengubah kebijakan.
waktu_kebijakan yang dimodifikasi	timestamp	Stempel waktu kapan kebijakan dibuat atau terakhir diubah.

SVV_ML_MODEL_INFO

Nyatakan informasi tentang keadaan model pembelajaran mesin saat ini.

SVV_ML_MODEL_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	karakter (128)	Basis data model.
schema_name	karakter (128)	Skema model.
user_name	karakter (128)	Pemilik model.
nama_model	karakter (128)	Nama modul.
siklus hidup	arang (20)	Status siklus hidup model.
dapat diperbaharui	integer	Status model apakah dapat disegarkan jika tabel dan kolom asli dalam kueri pelatihan masih ada dan pengguna masih memiliki izin untuk mereka. Nilai yang mungkin adalah: 1 (dapat disegarkan) dan 0 (tidak dapat disegarkan).
model_status	karakter (128)	Keadaan model saat ini.

Contoh kueri

Kueri berikut menampilkan keadaan model pembelajaran mesin saat ini.

```
SELECT schema_name, model_name, model_state
FROM svv_ml_model_info;

 schema_name |        model_name            |             model_state
-------------+------------------------------+-----------------------------------
```

```
 public      | customer_churn_auto_model    | Train Model On SageMaker In Progress
 public      | customer_churn_xgboost_model | Model is Ready
(2 row)
```

SVV_ML_MODEL_HAK ISTIMEWA

Gunakan SVV_ML_MODEL_PRIVILEGES untuk melihat izin model pembelajaran mesin yang secara eksplisit diberikan kepada pengguna, peran, dan grup dalam cluster.

SVV_ML_MODEL_PRIVILEGES dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_ruang_nama	text	Nama namespace tempat model pembelajaran mesin tertentu ada.
nama_model	text	Nama model pembelajaran mesin.
model_versi	integer	Nomor versi model.
tipe hak istimewa	text	Jenis izin. Nilai yang mungkin adalah EXECUTE.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.

Nama kolom	Jenis data	Deskripsi
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_ML_MODEL_PRIVILEGES.

```
SELECT
 namespace_name,model_name,model_version,privilege_type,identity_name,identity_type,admin_optio
 FROM svv_ml_model_privileges
WHERE model_name = 'test_model';

 namespace_name | model_name | model_version | privilege_type |  identity_name |
 identity_type | admin_option
----------------+------------+---------------+----------------+----------------
+---------------+--------------
      public    | test_model |       1       |    EXECUTE     |     reguser    |
 user      |    False
      public    | test_model |       1       |    EXECUTE     |     role1      |
 role      |    False
```

SVV_MV_DEPENDENSI

Tabel SVV_MV_DEPENDENCY menunjukkan dependensi tampilan yang terwujud pada tampilan terwujud lainnya dalam Amazon Redshift.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

SVV_MV_DEPENDENCY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	karakter (128)	Basis data yang berisi tampilan terwujud yang ditentukan.
schema_name	karakter (128)	Skema tampilan yang terwujud.
name	karakter (128)	Nama tampilan yang terwujud.
nama_database dependen	karakter (128)	Basis data tampilan terwujud yang menjadi sandaran tampilan terwujud ini.
nama_skema tergantung	karakter (128)	Skema tampilan terwujud yang menjadi sandaran tampilan terwujud ini.
nama_kete rgantungan	karakter (128)	Nama pandangan terwujud yang menjadi sandaran pandangan terwujud ini.

Contoh kueri

Kueri berikut mengembalikan baris keluaran yang menunjukkan bahwa tampilan terwujud `mv_over_foo` menggunakan tampilan terwujud `mv_foo` dalam definisinya sebagai ketergantungan.

```
CREATE SCHEMA test_ivm_setup;
CREATE TABLE test_ivm_setup.foo(a INT);
CREATE MATERIALIZED VIEW test_ivm_setup.mv_foo AS SELECT * FROM test_ivm_setup.foo;
CREATE MATERIALIZED VIEW test_ivm_setup.mv_over_foo AS SELECT * FROM
 test_ivm_setup.mv_foo;

SELECT * FROM svv_mv_dependency;

 database_name | schema_name          | name        | dependent_database_name |
 dependent_schema_name     | dependent_name
---------------+----------------------+-------------+-------------------------
+---------------------------+----------
 dev           | test_ivm_setup       | mv_over_foo |                     dev |
 test_ivm_setup            | mv_foo
```

SVV_MV_INFO

Tabel SVV_MV_INFO berisi baris untuk setiap tampilan yang terwujud, apakah data basi, dan informasi status.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

SVV_MV_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	karakter (128)	Basis data yang berisi tampilan terwujud.
schema_name	karakter (128)	Skema database.
user_name	karakter (128)	Pengguna yang memiliki tampilan terwujud.
name	karakter (128)	Nama tampilan yang terwujud.
is_basi	harimau (1)	A `t` menunjukkan bahwa pandangan yang terwujud sudah basi. Tampilan terwujud basi adalah tampilan di mana tabel dasar telah diperbarui tetapi tampilan yang terwujud belum diperbarui. An `f` menunjukkan bahwa tampilan yang terwujud diperbarui dengan perubahan tabel dasar. A u menunjukkan kebuntuan yang tidak diketahui (misalnya, dalam kasus MV pada tabel eksternal biasa). Informasi ini mungkin tidak akurat jika penyegaran belum dijalankan sejak restart terakhir.
status	integer	Keadaan pandangan yang terwujud sebagai berikut:

Nama kolom	Jenis data	Deskripsi
		• 0 — Tampilan terwujud sepenuhnya dihitung ulang saat di-refresh. • 1 — Pandangan yang terwujud bersifat inkremental. • 101 - Tampilan yang terwujud tidak dapat disegarkan karena kolom yang dijatuhkan. Kendala ini berlaku bahkan jika kolom tidak digunakan dalam tampilan yang terwujud. • 102 - Tampilan yang terwujud tidak dapat disegarkan karena jenis kolom yang diubah. Kendala ini berlaku bahkan jika kolom tidak digunakan dalam tampilan yang terwujud. • 103 — Tampilan terwujud tidak dapat disegarka n karena tabel berganti nama. • 104 - Tampilan terwujud tidak dapat disegarka n karena kolom berganti nama. Kendala ini berlaku bahkan jika kolom tidak digunakan dalam tampilan yang terwujud. • 105 - Tampilan terwujud tidak dapat disegarkan karena skema berganti nama.
penulisan ulang otomatis	harimau (1)	A `t` menunjukkan bahwa tampilan yang terwujud memenuhi syarat untuk penulisan ulang kueri secara otomatis.
Refresh otomatis	harimau (1)	A `t` menunjukkan bahwa tampilan yang terwujud dapat disegarkan secara otomatis.

Contoh kueri

Untuk melihat keadaan semua tampilan yang terwujud, jalankan kueri berikut.

```
select * from svv_mv_info;
```

Kueri ini mengembalikan output sampel berikut.

```
database_name |       schema_name       | user_name |   name   |  is_stale | state |
 autorefresh | autorewrite
--------------+-------------------------+-----------+---------+-----------+-------
+-------------+---------------
 dev          | test_ivm_setup          | catch-22  | mv       |   f       |     1 |
       1 |           0
 dev          | test_ivm_setup          | lotr      | old_mv   |   t       |     1 |
       0 |           1
```

SVV_QUERY_DALAM PENERBANGAN

Gunakan tampilan SVV_QUERY_INFLIGHT untuk menentukan kueri apa yang sedang berjalan di database. Tampilan ini bergabung STV_INFLIGHT dengan STL_QUERYTEXT. SVV_QUERY_INFLIGHT tidak menampilkan kueri hanya leader-node. Untuk informasi selengkapnya, lihat Node—hanya fungsi utama.

SVV_QUERY_INFLIGHT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> Tampilan ini hanya tersedia saat menanyakan klaster yang disediakan.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
mengiris	integer	Iris tempat kueri berjalan.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
pid	integer	ID proses. Semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama. Anda dapat menggunakan kolom ini untuk bergabung dengan STL_ERROR tabel.
waktu mulai	timestamp	Waktu permintaan dimulai.
tergantung	integer	Apakah kueri ditangguhkan: **0** = false; **1** = true.
text	karakter (200)	Teks kueri, dalam penambahan 200 karakter.
urutan	integer	Nomor urutan untuk segmen pernyataan kueri.

Kueri Sampel

Contoh output di bawah ini menunjukkan dua kueri yang sedang berjalan, kueri SVV_QUERY_INFLIGHT itu sendiri dan query 428, yang dibagi menjadi tiga baris dalam tabel. (Kolom starttime dan pernyataan dipotong dalam output sampel ini.)

```
select slice, query, pid, starttime, suspended, trim(text) as statement, sequence
from svv_query_inflight
order by query, sequence;

slice|query| pid  |      starttime       |suspended| statement | sequence
-----+-----+------+----------------------+---------+-----------+---------
1012 | 428 | 1658 | 2012-04-10 13:53:... |       0 | select ...|    0
1012 | 428 | 1658 | 2012-04-10 13:53:... |       0 | enueid ...|    1
1012 | 428 | 1658 | 2012-04-10 13:53:... |       0 | atname,...|    2
1012 | 429 | 1608 | 2012-04-10 13:53:... |       0 | select ...|    0
(4 rows)
```

SVV_QUERY_STATE

Gunakan SVV_QUERY_STATE untuk melihat informasi tentang runtime kueri yang sedang berjalan.

Tampilan SVV_QUERY_STATE berisi subset data dari tabel STV_EXEC_STATE.

SVV_QUERY_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_DETAIL. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

> **Note**
>
> Tampilan ini hanya tersedia saat menanyakan klaster yang disediakan.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
seg	integer	Jumlah segmen query yang sedang berjalan. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen query dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses.
langkah	integer	Jumlah langkah kueri yang sedang berjalan. Langkah adalah unit terkecil dari runtime query. Setiap langkah mewakili unit kerja diskrit, seperti memindai tabel, mengembalikan hasil, atau menyortir data.

Nama kolom	Jenis data	Deskripsi
waktu maksimal	interval	Jumlah waktu maksimum (dalam mikrodetik) untuk langkah ini berjalan.
avgtime	interval	Waktu rata-rata (dalam mikrodetik) untuk langkah ini berjalan.
baris	bigint	Jumlah baris yang dihasilkan oleh langkah yang sedang berjalan.
bytes	bigint	Jumlah byte yang dihasilkan oleh langkah yang sedang berjalan.
cpu	bigint	Untuk penggunaan internal.
memory	bigint	Untuk penggunaan internal.
rate_baris	double precision	Rows-per-second tingkat sejak kueri dimulai, dihitung dengan menjumlahkan baris dan membaginya dengan jumlah detik dari saat kueri dimulai hingga waktu saat ini.
rate_byte	double precision	Bytes-per-second tingkat sejak kueri dimulai, dihitung dengan menjumlahkan byte dan membaginya dengan jumlah detik dari saat kueri dimulai hingga waktu saat ini.
label	karakter (25)	Label kueri: nama untuk langkah, seperti `scan` atau`sort`.
berbasis-disk	karakter (1)	Apakah langkah kueri ini berjalan sebagai operasi berbasis disk: true (**`t`**) atau false (**`f`**). Hanya langkah-langkah tertentu, seperti langkah hash, sort, dan agregat, yang dapat masuk ke disk. Banyak jenis langkah selalu dilakukan dalam memori.
workmem	bigint	Jumlah memori kerja (dalam byte) yang ditetapkan untuk langkah kueri.
nom_bagi n	integer	Jumlah partisi yang dibagi menjadi tabel hash selama langkah hash. Angka positif dalam kolom ini tidak menyiratkan bahwa langkah hash berjalan sebagai operasi berbasis disk. Periksa nilai di kolom IS_DISKBASED untuk melihat apakah langkah hash berbasis disk.

Nama kolom	Jenis data	Deskripsi
is_rrscan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian terbatas rentang digunakan pada langkah tersebut. Defaultnya adalah false (**f**).
is_tertunda_scan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian tertunda digunakan pada langkah tersebut. Defaultnya adalah false (**f**).

Kueri Sampel

Menentukan waktu pemrosesan kueri demi langkah

Kueri berikut menunjukkan berapa lama waktu yang dibutuhkan setiap langkah kueri dengan ID kueri 279 untuk dijalankan dan berapa banyak baris data yang diproses Amazon Redshift:

```
select query, seg, step, maxtime, avgtime, rows, label
from svv_query_state
where query = 279
order by query, seg, step;
```

Kueri ini mengambil informasi pemrosesan tentang query 279, seperti yang ditunjukkan dalam output sampel berikut:

```
query |   seg   | step | maxtime | avgtime |  rows   | label
------+---------+------+---------+---------+---------+-------------------
  279 |       3 |    0 | 1658054 | 1645711 | 1405360 | scan
  279 |       3 |    1 | 1658072 | 1645809 |       0 | project
  279 |       3 |    2 | 1658074 | 1645812 | 1405434 | insert
  279 |       3 |    3 | 1658080 | 1645816 | 1405437 | distribute
  279 |       4 |    0 | 1677443 | 1666189 | 1268431 | scan
  279 |       4 |    1 | 1677446 | 1666192 | 1268434 | insert
  279 |       4 |    2 | 1677451 | 1666195 |       0 | aggr
(7 rows)
```

Menentukan apakah ada kueri aktif yang sedang berjalan di disk

Kueri berikut menunjukkan apakah ada kueri aktif yang sedang berjalan di disk:

```
select query, label, is_diskbased from svv_query_state
```

```
where is_diskbased = 't';
```

Output sampel ini menunjukkan kueri aktif yang saat ini berjalan di disk:

```
 query | label        | is_diskbased
-------+--------------+--------------
1025   | hash tbl=142 |      t
(1 row)
```

SVV_REDSHIFT_KOLOM

Gunakan SVV_REDSHIFT_COLUMNS untuk melihat daftar semua kolom yang dapat diakses pengguna. Kumpulan kolom ini mencakup kolom pada cluster dan kolom dari berbagi data yang disediakan oleh cluster jarak jauh.

SVV_REDSHIFT_COLUMNS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW COLUMNS untuk penemuan kolom. SHOW COLUMNS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama database tempat tabel yang berisi kolom ada.
schema_name	bangkai (128)	Nama skema untuk tabel.

Nama kolom	Jenis data	Deskripsi
table_name	bangkai (128)	Nama tabel.
column_name	bangkai (128)	Nama sebuah kolom.
ordinal_position	integer	Posisi kolom dalam tabel.
data_type	varchar(32)	Jenis data kolom.
column_default	varchar(4000)	Nilai default kolom.
is_nullable	Cucing (3)	Nilai yang menentukan apakah kolom dapat dibatalkan. Nilai yang mungkin adalah `yes`no,, dan "" (string kosong yang tidak mewakili informasi).
encoding	bangkai (128)	Jenis pengkodean kolom.
distkey	boolean	Nilai yang benar jika kolom ini adalah kunci distribusi untuk tabel, dan false sebaliknya.

Nama kolom	Jenis data	Deskripsi
sortkey	integer	Nilai yang menentukan urutan kolom dalam kunci penguruta n. Jika tabel menggunakan kunci pengurutan majemuk, maka semua kolom yang merupakan bagian dari kunci penguruta n memiliki nilai positif yang menunjukkan posisi kolom dalam kunci pengurutan. Jika tabel menggunakan kunci pengurutan yang disisipka n, maka setiap kolom yang merupakan bagian dari kunci pengurutan memiliki nilai yang secara bergantian positif atau negatif. Di sini, nilai absolut menunjukkan posisi kolom dalam kunci pengurutan. Jika `sortkey` 0, kolom tersebut bukan bagian dari kunci pengurutan.
column_acl	bangkai (128)	String yang mendefinisikan izin untuk pengguna atau grup pengguna yang ditentukan untuk kolom.
komentar	bangsawan (256)	Keterangan.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_REDSHIFT_COLUMNS.

```
SELECT *
FROM svv_redshift_columns
WHERE database_name = 'tickit_db'
    AND TABLE_NAME = 'tickit_sales_redshift'
ORDER BY COLUMN_NAME,
    TABLE_NAME,
    database_name
LIMIT 5;

database_name | schema_name |       table_name      | column_name | ordinal_position |
 data_type | column_default | is_nullable | encoding | distkey | sortkey | column_acl
 | remarks
--------------+-------------+-----------------------+-------------+------------------
+-----------+----------------+-------------+----------+---------+---------
+-------------+--------
   tickit_db  |   public    | tickit_sales_redshift |   buyerid   |        4         |
 integer  |                |      NO     |   az64   |  False  |    0    |             |
   tickit_db  |   public    | tickit_sales_redshift |  commission |        9         |
 numeric  |      (8,2)     |     YES     |   az64   |  False  |    0    |             |
   tickit_db  |   public    | tickit_sales_redshift |    dateid   |        6         |
 smallint |                |      NO     |   none   |  False  |    1    |             |
   tickit_db  |   public    | tickit_sales_redshift |   eventid   |        5         |
 integer  |                |      NO     |   az64   |  False  |    0    |       |
   tickit_db  |   public    | tickit_sales_redshift |   listid    |        2         |
 integer  |                |      NO     |   az64   |  True   |    0    |             |
```

SVV_REDSHIFT_DATABASE

Gunakan SVV_ REDSHIFT_DATABASES untuk melihat daftar semua database yang dapat diakses pengguna. Ini termasuk database di cluster dan database yang dibuat dari berbagi data yang disediakan oleh cluster jarak jauh.

SVV_REDSHIFT_DATABASES terlihat oleh semua pengguna secara default. Untuk mengontrol akses ke metadata database Anda, aktifkan keamanan metadata untuk cluster yang disediakan atau grup kerja tanpa server. Keamanan metadata memungkinkan Anda memisahkan izin tampilan untuk metadata objek berdasarkan pengguna dan peran. Untuk informasi selengkapnya, lihat Keamanan metadata.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW DATABASE untuk penemuan database. SHOW DATABASE bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama basis data.
pemilai_database	integer	ID pengguna pemilik database.
Tipe_database	varchar(32)	Jenis database. Jenis yang mungkin adalah database lokal atau bersama.
database_acl	bangkai (128)	Informasi ini hanya untuk penggunaan internal.
opsi_database	bangkai (128)	Properti database.
tingkat isolasi_basis data	bangkai (128)	Tingkat isolasi database. Nilai yang mungkin termasuk: `Snapshot Isolation` dan`Serializable` .

Contoh kueri

Contoh berikut mengembalikan output untuk SVV_REDSHIFT_DATABASES.

```
select database_name, database_owner, database_type, database_options,
 database_isolation_level
from  svv_redshift_databases;
```

```
database_name | database_owner | database_type | database_options |
 database_isolation_level
--------------+----------------+---------------+------------------+------------------
   dev        |  1             | local         | NULL             | Serializable
```

SVV_REDSHIFT_FUNCTION

Gunakan SVV_REDSHIFT_FUNCTIONS untuk melihat daftar semua fungsi yang dapat diakses pengguna. Kumpulan fungsi ini mencakup fungsi pada cluster dan fungsi dari berbagi data yang disediakan oleh cluster jarak jauh.

SVV_REDSHIFT_FUNCTIONS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW FUNCTIONS untuk penemuan fungsi. SHOW FUNCTIONS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama database tempat cluster yang memiliki fungsi-fungsi ini ada.
schema_name	bangkai (128)	Nama skema yang menentuka n fungsi tertentu.

Nama kolom	Jenis data	Deskripsi
function_name	bangkai (128)	Nama fungsi yang ditentukan.
tipe_fungsi	bangkai (128)	Jenis fungsi. Nilai yang mungkin adalah fungsi reguler, fungsi agregat, dan prosedur tersimpan.
tipe argumen	bangsawan (512)	Sebuah string yang mewakili jenis argumen input fungsi.
tipe_hasil	bangkai (128)	Jenis data dari nilai pengembalian fungsi.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_REDSHIFT_FUNCTIONS.

```
SELECT *
FROM svv_redshift_functions
WHERE database_name = 'tickit_db'
    AND SCHEMA_NAME = 'public'
ORDER BY function_name
LIMIT 5;

database_name | schema_name |      function_name    |  function_type   |
 argument_type  | result_type
--------------+-------------+-----------------------+------------------
+------------------+-------------
   tickit_db  |    public   |     shared_function   | REGULAR FUNCTION | integer,
 integer |   integer
```

SVV_REDSHIFT_SCHEMA_KUOTA

Menampilkan kuota dan penggunaan disk saat ini untuk setiap skema dalam database.

SVV_REDSHIFT_SCHEMA_QUOTA dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Tampilan ini tersedia saat menanyakan klaster yang disediakan atau kelompok kerja Redshift Serverless.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	karakter (128)	Database yang berisi skema.
schema_name	karakter (128)	Nama skema.
pemilik skema	integer	ID pengguna internal pemilik skema.
kuota	integer	Jumlah ruang disk (dalam MB) yang dapat digunakan skema.
penggunaan disk	integer	Ruang disk (dalam MB) yang saat ini digunakan oleh skema.

Contoh kueri

Contoh berikut menampilkan kuota dan penggunaan disk saat ini untuk skema bernama`sales_schema`.

```
SELECT TRIM(SCHEMA_NAME) "schema_name", QUOTA, disk_usage FROM
 svv_redshift_schema_quota
WHERE SCHEMA_NAME = 'sales_schema';


schema_name   | quota | disk_usage
--------------+-------+------------
sales_schema  | 2048  | 30
```

SVV_REDSHIFT_SKEMA

Gunakan SVV_REDSHIFT_SCHEMAS untuk melihat daftar semua skema yang dapat diakses pengguna. Kumpulan skema ini mencakup skema pada cluster dan skema dari berbagi data yang disediakan oleh cluster jarak jauh.

SVV_REDSHIFT_SCHEMAS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW SCHEMAS untuk penemuan skema. SHOW SCHEMAS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama database di mana skema tertentu ada.
schema_name	bangkai (128)	Namespace atau nama skema.
pemilik skema	integer	ID pengguna internal pemilik skema.
tipe_skema	varchar(16)	Jenis skema. Nilai yang mungkin adalah skema bersama dan lokal.
schema_acl	bangkai (128)	String yang mendefinisikan izin untuk pengguna atau grup pengguna yang ditentukan untuk skema.
opsi_skema	bangkai (128)	Opsi skema.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_REDSHIFT_SCHEMAS.

```
SELECT *
FROM svv_redshift_schemas
WHERE database_name = 'tickit_db'
ORDER BY database_name,
    SCHEMA_NAME;

database_name |    schema_name     | schema_owner | schema_type | schema_acl |
 schema_option
--------------+--------------------+--------------+-------------+------------
+---------------
   tickit_db  |       public       |       1      |    shared   |            |
```

SVV_REDSHIFT_TABEL

Gunakan SVV_REDSHIFT_TABLES untuk melihat daftar semua tabel yang dapat diakses pengguna. Kumpulan tabel ini mencakup tabel di cluster dan tabel dari berbagi data yang disediakan oleh cluster jarak jauh.

SVV_REDSHIFT_TABLES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW TABLES untuk penemuan tabel. SHOW TABLES bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	bangkai (128)	Nama database di mana tabel tertentu ada.
schema_name	bangkai (128)	Nama skema untuk tabel.
table_name	bangkai (128)	Nama tabel.
table_type	bangkai (128)	Jenis meja. Nilai yang mungkin adalah tampilan dan tabel.
table_acl	bangkai (128)	String yang mendefinisikan izin untuk pengguna atau grup pengguna yang ditentukan untuk tabel.
komentar	bangkai (128)	Keterangan.
pemilik tabel_	bangkai (128)	Pemilik tabel.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_REDSHIFT_TABLES.

```
SELECT *
FROM svv_redshift_tables
WHERE database_name = 'tickit_db' AND TABLE_NAME LIKE 'tickit_%'
ORDER BY database_name,
TABLE_NAME;

database_name | schema_name |         table_name       | table_type | table_acl |
 remarks | table_owner
--------------+-------------+--------------------------+------------+-----------
+---------+-----------
   tickit_db  |    public   | tickit_category_redshift |    TABLE   |           |
   +
```

```
tickit_db  |    public   |   tickit_date_redshift   |    TABLE   |          |
+
tickit_db  |    public   |   tickit_event_redshift  |    TABLE   |          |
+
tickit_db  |    public   |  tickit_listing_redshift |    TABLE   |          |
+
tickit_db  |    public   |   tickit_sales_redshift  |    TABLE   |          |
+
tickit_db  |    public   |   tickit_users_redshift  |    TABLE   |          |
+
tickit_db  |    public   |   tickit_venue_redshift  |    TABLE   |          |
```

Jika nilai table_acl adalah null, tidak ada hak akses yang secara eksplisit diberikan ke tabel yang sesuai.

HAK ISTIMEWA SVV_RELASI

Gunakan SVV_RELATION_PRIVILEGES untuk melihat izin relasi (tabel dan tampilan) yang secara eksplisit diberikan kepada pengguna, peran, dan grup dalam database saat ini.

SVV_RELATION_PRIVILEGES dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin SYSLOG ACCESS TIDAK TERBATAS

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses. Untuk informasi selengkapnya tentang visibilitas data, lihatVisibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW GRANTS untuk penemuan izin. SHOW GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_ruang_nama	text	Nama namespace di mana relasi tertentu ada.
nama relasi	text	Nama relasi.
keistimewa_tipe	text	Jenis izin. Nilai yang mungkin adalah INSERT, SELECT, UPDATE, DELETE, REFERENSI, atau DROP.
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_RELATION_PRIVILEGES.

```
SELECT
 namespace_name,relation_name,privilege_type,identity_name,identity_type,admin_option
 FROM svv_relation_privileges
WHERE relation_name = 'orders' AND privilege_type = 'SELECT';

 namespace_name | relation_name | privilege_type |  identity_name | identity_type |
 admin_option
----------------+---------------+----------------+----------------+---------------
+--------------
     public     |    orders     |     SELECT     |    reguser     |     user      |
 False
```

```
    public     |    orders     |     SELECT     |     role1      |     role      |
False
```

SVV_RLS_APPLICED_POLICY

Gunakan SVV_RLS_APPLIED_POLICY untuk melacak penerapan kebijakan RLS pada kueri yang mereferensikan hubungan. RLS-protected

SVV_RLS_APPLIED_POLICY terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan `sys:operator` peran
- Pengguna dengan izin TABEL SISTEM AKSES

Perhatikan bahwa sys:secadmin tidak diberikan izin sistem ini.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama pengguna	text	Nama pengguna yang menjalankan query.
kueri	integer	ID kueri.
xid	long	Konteks transaksi.
pid	integer	Proses pemimpin menjalankan kueri.
waktu rekor	Waktu	Waktu ketika kueri direkam.
perintah	harimau (1)	Perintah yang menerapkan kebijakan RLS. Nilai yang mungkin adalah k untuk tidak diketahui, s untuk pilih, u untuk pembaruan, i untuk menyisipkan, y untuk utilitas, dan d untuk menghapus.
nama tanggal	text	Nama database hubungan yang dilampirkan kebijakan keamanan tingkat baris.

Nama kolom	Jenis data	Deskripsi
jadwal relasi	text	Nama skema relasi yang dilampirkan kebijakan keamanan tingkat baris.
rename	text	Nama relasi yang dilampirkan kebijakan keamanan tingkat baris.
polname	text	Nama kebijakan keamanan tingkat baris yang dilampirkan ke relasi.
poldefault	haru (1)	Pengaturan default kebijakan keamanan tingkat baris yang dilampirkan ke relasi. Vaule yang mungkin adalah `f` untuk false jika kebijakan false default telah diterapkan dan `t` untuk true jika kebijakan true default telah diterapkan.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_RLS_APPLIED_POLICY. Untuk menanyakan SVV_RLS_APPLIED_POLICY, Anda harus memiliki izin ACCESS SYSTEM TABLE.

```
-- Check what RLS policies were applied to the run query.
SELECT username, command, datname, relschema, relname, polname, poldefault
FROM svv_rls_applied_policy
WHERE datname = CURRENT_DATABASE() AND query = PG_LAST_QUERY_ID();

 username | command |  datname  | relschema |          relname         |      polname
  | poldefault
----------+---------+-----------+-----------+--------------------------
+-----------------+------------
   molly  |    s    | tickit_db |   public  | tickit_category_redshift |
 policy_concerts |
```

SVV_RLS_TERLAMPIR_KEBIJAKAN

Gunakan SVV_RLS_ATTACHED_POLICY untuk melihat daftar semua relasi dan pengguna yang memiliki satu atau beberapa kebijakan keamanan tingkat baris yang dilampirkan pada database yang saat ini terhubung.

Hanya pengguna dengan peran sys:secadmin yang dapat menanyakan tampilan ini.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
jadwal relasi	text	Nama skema relasi yang dilampirkan kebijakan keamanan tingkat baris.
rename	text	Nama relasi yang dilampirkan kebijakan keamanan tingkat baris.
relkind	text	Jenis objek, seperti tabel.
polname	text	Nama kebijakan keamanan tingkat baris yang dilampirkan ke relasi.
pemberi	text	Nama pengguna yang telah melampirkan kebijakan ini.
penerima hibah	text	Nama pengguna atau peran yang dilampirkan kebijakan ini.
pemberi hibah	text	Jenis penerima hibah. Nilai yang mungkin adalah pengguna atau peran.
is_pol_on	boolean	Parameter yang menunjukkan apakah kebijakan keamanan tingkat baris diaktifkan atau dinonaktifkan pada tabel. Nilai yang mungkin benar dan salah.
is_rls_on	boolean	Parameter yang menunjukkan apakah keamanan tingkat baris dihidupkan atau dimatikan pada tabel. Nilai yang mungkin benar dan salah.
rls_tipe_ konjungsion_	karakter (3)	Parameter yang menunjukkan apakah relasi menggabungkan kebijakan RLS dengan `and` atau`or`.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_RLS_ATTACHED_POLICY.

```
--Inspect the policy in SVV_RLS_ATTACHED_POLICY
SELECT * FROM svv_rls_attached_policy;
```

```
 relschema |        relname           | relkind |     polname     | grantor | grantee
 | granteekind | is_pol_on | is_rls_on | rls_conjuntion_type
-----------+--------------------------+---------+-----------------+---------+----------
+------------+----------+----------+--------------------
 public    | tickit_category_redshift |  table  | policy_concerts |   bob   |  analyst
 |    role     |    True   |    True   |      and
 public    | tickit_category_redshift |  table  | policy_concerts |   bob   |  dbadmin
 |    role     |    True   |    True   |      and
```

SVV_RLS_POLICY

Gunakan SVV_RLS_POLICY untuk melihat daftar semua kebijakan keamanan tingkat baris yang dibuat di cluster Amazon Redshift.

SVV_RLS_POLICY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah TAM PILKAN KEBIJAKAN untuk penemuan kebijakan. SHOW POLICYS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
poldb	text	Nama database tempat kebijakan keamanan tingkat baris dibuat.
polname	text	Nama kebijakan keamanan tingkat baris.
polalia	text	Alias tabel yang digunakan dalam definisi kebijakan.

Nama kolom	Jenis data	Deskripsi
polatt	text	Atribut yang disediakan untuk definisi kebijakan.
polqual	text	Kondisi kebijakan yang disediakan dalam klausa USING dari pernyataan CREATE POLICY.
dipolabel	boolean	Apakah kebijakan diaktifkan secara global.
polmodidi edoleh	text	Nama pengguna yang membuat atau mengubah kebijakan terbaru.
waktu yang dimodifikasi	timestamp	Stempel waktu kapan kebijakan dibuat atau terakhir diubah.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_RLS_POLICY.

```
-- Create some policies.
CREATE RLS POLICY pol1 WITH (a int) AS t USING ( t.a IS NOT NULL );
CREATE RLS POLICY pol2 WITH (c varchar(10)) AS t USING ( c LIKE '%public%');

-- Inspect the policy in SVV_RLS_POLICY
SELECT * FROM svv_rls_policy;

 poldb | polname | polalias |                     polatts                      |
         polqual                | polenabled | polmodifiedby |   polmodifiedtime
-------+---------+----------+--------------------------------------------------
+--------------------------------------+-----------+--------------
+---------------------
 my_db | pol1    | t        | [{"colname":"a","type":"integer"}]               |
 "t"."a" IS NOT NULL                  | t          | policy_admin  | 2022-02-11
 14:40:49
 my_db | pol2    | t        | [{"colname":"c","type":"character varying(10)"}] |
 "t"."c" LIKE CAST('%public%' AS TEXT) | t          | policy_admin  | 2022-02-11
 14:41:28
```

SVV_RLS_RELASI

Gunakan SVV_RLS_RELATION untuk melihat daftar semua relasi yang ada. RLS-protected

SVV_RLS_RELATION terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama tanggal	text	Nama database yang berisi relasi.
jadwal relasi	text	Nama skema yang berisi relasi.
rename	text	Nama relasi.
relkind	text	Jenis relasi, seperti tabel atau tampilan.
is_rls_on	boolean	Parameter yang menunjukkan apakah relasi itu RLS-protected.
is_rls_datashare_on	boolean	Parameter yang menunjukkan apakah relasi tersebut RLS-protected lebih dari berbagi data.
rls_tipe_konjungsion_	karakter (3)	Parameter yang menunjukkan apakah relasi menggabungkan kebijakan RLS dengan `and` atau `or`.
rls_datashare_tipe_konjungsion_	karakter (3)	Parameter yang menunjukkan apakah relasi menggabungkan kebijakan RLS dengan `and` atau `or` melalui data share.

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_RLS_RELATION.

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON FOR DATASHARES;


--Inspect RLS state on the relations using SVV_RLS_RELATION.
SELECT datname, relschema, relname, relkind, is_rls_on, is_rls_datashare_on FROM
 svv_rls_relation ORDER BY relname;
```

```
  datname  | relschema |        relname           | relkind | is_rls_on |
 is_rls_datashare_on | rls_conjunction_type | rls_datashare_conjunction_type
----------+-----------+--------------------------+---------+-----------
+---------------------+----------------------+--------------------------------
 tickit_db |   public  | tickit_category_redshift |  table  |      t    |           t
       |          and         |              and
(1 row)
```

SVV_ROLE_GRANT

Gunakan SVV_ROLE_GRANTS untuk melihat daftar peran yang secara eksplisit diberikan peran dalam cluster.

SVV_ROLE_GRANTS dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
rol_id	integer	ID peran.
role_name	text	Nama perannya.
_rol_id yang diberikan	integer	ID untuk peran yang diberikan.
nama_peran yang diberikan	text	Nama untuk peran yang diberikan.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_ROLE_GRANTS.

```
GRANT ROLE role1 TO ROLE role2;
```

```
GRANT ROLE role2 TO ROLE role3;

SELECT role_name, granted_role_name FROM svv_role_grants;

 role_name |  granted_role_name
-----------+--------------------
   role2   |      role1
   role3   |      role2
(2 rows)
```

SVV_ROLE

Gunakan SVV_ROLES untuk melihat informasi peran.

Tabel ini terlihat oleh semua pengguna.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
rol_id	integer	ID peran.
role_name	text	Nama perannya.
pemilai_peran	text	Nama pemilik peran.
eksternal_id	text	Pengidentifikasi unik peran dalam penyedia identitas pihak ketiga.

Contoh kueri

Contoh berikut mengembalikan output dari SVV_ROLES.

```
SELECT role_name,role_owner FROM svv_roles WHERE role_name IN ('role1', 'role2');

 role_name | role_owner
-----------+------------
   role1   | superuser
   role2   | superuser
```

SVV_SCHEMA_HAK ISTIMEWA

Gunakan SVV_SCHEMA_PRIVILEGES untuk melihat izin skema yang secara eksplisit diberikan kepada pengguna, peran, dan grup dalam database saat ini.

SVV_SCHEMA_PRIVILEGES terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW GRANTS untuk penemuan izin. SHOW GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_ruang_nama	text	Nama namespace tempat skema tertentu ada.
keistimewa_tipe	text	Jenis izin. Untuk izin dengan skema privilege_scope, nilai yang mungkin adalah CREATE, USAGE, dan ALTER. Untuk nilai privilege_scope selain skema, nilai yang mungkin termasuk jenis izin apa pun yang tersedia pada cakupan izin.

Nama kolom	Jenis data	Deskripsi
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna, ID peran, atau ID grup.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna, peran, grup, atau publik.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan izin kepada pengguna dan peran lain. Itu selalu salah untuk peran dan tipe identitas grup.
cakupan hak istimewa	text	Ruang lingkup izin yang ditentukan dalam privilege_type. Kemungkinan nilainya adalah sebagai berikut: • SCHEMA • TABEL • FUNGSI Untuk informasi tentang izin cakupan, buka. Izin terlingkup

Contoh kueri

Contoh berikut menampilkan hasil dari SVV_SCHEMA_PRIVILEGES.

```
SELECT namespace_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_schema_privileges
WHERE namespace_name = 'test_schema1';

 namespace_name | privilege_type |  identity_name | identity_type | admin_option
```

```
---------------+----------------+----------------+---------------+-------------
 test_schema1  |    USAGE       |     reguser    |     user      |   False
 test_schema1  |    USAGE       |     role1      |     role      |   False
```

SVV_SCHEMA_KUOTA_STATUS

Menampilkan kuota dan penggunaan disk saat ini untuk setiap skema.

Pengguna biasa dapat melihat informasi untuk skema yang mereka memiliki izin USAGE. Superuser dapat melihat informasi untuk semua skema dalam database saat ini.

SVV_SCHEMA_QUOTA_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> Tampilan ini hanya tersedia saat menanyakan klaster yang disediakan.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Schema_id	integer	Namespace atau ID skema.
schema_name	karakter (128)	Namespace atau nama skema.
pemilik skema	integer	ID pengguna internal pemilik skema.
kuota	integer	Jumlah ruang disk (dalam MB) yang dapat digunakan skema.
penggunaan disk	integer	Ruang disk (dalam MB) yang saat ini digunakan oleh skema.
disk_peng gunaan_pct	double precision	Persentase ruang disk yang saat ini digunakan oleh skema di luar kuota yang dikonfigurasi.

Contoh kueri

Contoh berikut menampilkan kuota dan penggunaan disk saat ini untuk skema.

```
SELECT TRIM(SCHEMA_NAME) "schema_name", QUOTA, disk_usage, disk_usage_pct FROM
 svv_schema_quota_state
WHERE SCHEMA_NAME = 'sales_schema';
schema_name   | quota | disk_usage | disk_usage_pct
--------------+-------+------------+----------------
sales_schema  | 2048  | 30         | 1.46
(1 row)
```

SVV_SYSTEM_HAK ISTIMEWA

SVV_SYSTEM_PRIVILEGES dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat identitas yang mereka miliki atau akses.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW GRANTS untuk penemuan izin. SHOW GRANTS bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
hak istimewa sistem	text	Nama izin sistem.

Nama kolom	Jenis data	Deskripsi
identitas_id	integer	ID identitas. Nilai yang mungkin adalah ID pengguna atau ID peran.
nama_identitas	text	Nama identitas.
tipe_identitas	text	Jenis identitas. Nilai yang mungkin adalah pengguna atau peran.

Contoh kueri

Contoh berikut menampilkan hasil untuk parameter yang ditentukan.

```
SELECT system_privilege,identity_name,identity_type FROM svv_system_privileges
WHERE system_privilege = 'ALTER TABLE' AND identity_name = 'sys:superuser';

 system_privilege | identity_name | identity_type
------------------+---------------+--------------
   ALTER TABLE    | sys:superuser |     role
```

SVV_TABLE_INFO

Menampilkan informasi ringkasan untuk tabel dan tampilan terwujud dalam database yang terhubung saat ini. Tampilan menyaring tabel sistem, dan hanya menampilkan tabel yang ditentukan pengguna dan tampilan terwujud yang berisi setidaknya 1 baris data.

Anda dapat menggunakan tampilan SVV_TABLE_INFO untuk mendiagnosis dan mengatasi masalah desain tabel yang dapat memengaruhi kinerja kueri. Ini termasuk masalah dengan pengkodean kompresi, kunci distribusi, gaya pengurutan, kemiringan distribusi data, ukuran tabel, dan statistik. Tampilan SVV_TABLE_INFO tidak mengembalikan informasi apa pun untuk tabel kosong.

Tampilan SVV_TABLE_INFO merangkum informasi dari tabel sistem dan tabel katalog berikut:

- STV_NODE_STORAGE_KAPASITAS
- STV_SLICE
- STV_TBL_PERM
- PG_ATRIBUT

- PG_CLASS
- PG_DATABASE
- PG_NAMESPACE
- PG_STATISTIC_INDICATOR

SVV_TABLE_INFO hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem. Untuk mengizinkan pengguna menanyakan tampilan, berikan izin SELECT pada SVV_TABLE_INFO kepada pengguna.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database	text	Nama database.
schema	text	Nama skema.
table_id	oid	ID tabel.
table	text	Nama tabel.
encoded	text	Nilai yang menunjukkan apakah kolom memiliki pengkodean kompresi yang ditentukan.
diststyle	text	Gaya distribusi atau kolom kunci distribusi, jika distribusi kunci ditentuka n. Nilai yang mungkin termasukEVEN,KEY(*column*),ALL,AUT ,AUTO(EVEN) , danAUTO(KEY(*column*)).
sortkey1	text	Kolom pertama dalam kunci pengurutan, jika kunci pengurutan didefinis ikan. Nilai yang mungkin

Nama kolom	Jenis data	Deskripsi
		termasuk*column*,AUTO(SORT KEY) , danAUTO(SORT KEY(*column*)).
max_varchar	integer	Ukuran kolom terbesar yang menggunakan tipe data VARCHAR.
sortkey1_enc	karakter (32)	Pengkodean kompresi kolom pertama dalam kunci pengurutan, jika kunci pengurutan didefinisikan.
sortkey_num	integer	Jumlah kolom didefinisikan sebagai kunci pengurutan.
size	bigint	Ukuran tabel, dalam blok data 1-MB.
pct_used	numerik (10,4)	Persentase ruang yang tersedia yang digunakan oleh tabel.
empty	bigint	Untuk penggunaan internal. Kolom ini tidak lagi digunakan dan akan dihapus dalam rilis mendatang.
unsorted	numerik (5,2)	Persentase baris yang tidak disortir dalam tabel.
stats_off	numerik (5,2)	Angka yang menunjukkan seberapa basi statistik tabel; 0 adalah saat ini, 100 sudah ketinggalan zaman.

Nama kolom	Jenis data	Deskripsi
tbl_rows	numerik (38,0)	Jumlah total baris dalam tabel. Nilai ini termasuk baris yang ditandai untuk dihapus, tetapi belum disedot.
skew_sortkey1	numerik (19,2)	Rasio ukuran kolom kunci non-pengurutan terbesar dengan ukuran kolom pertama dari kunci pengurutan, jika kunci pengurutan ditentuka n. Gunakan nilai ini untuk mengevaluasi efektivitas kunci pengurutan.
skew_rows	numerik (19,2)	Rasio jumlah baris dalam irisan dengan baris terbanyak dengan jumlah baris dalam irisan dengan baris paling sedikit.
estimated_visible_rows	numerik (38,0)	Perkiraan baris dalam tabel. Nilai ini tidak termasuk baris yang ditandai untuk dihapus.

Nama kolom	Jenis data	Deskripsi

risk_event	text	Informasi risiko tentang tabel. Bidang dipisahkan menjadi beberapa bagian:
		
		`risk_type |xid|timestamp`
		
		• Yang`risk_type` , di mana 1 menunjukkan bahwa `COPY command with the EXPLICIT_IDS option` berlari. Amazon Redshift tidak lagi memeriksa keunikan kolom IDENTITY dalam tabel. Untuk informasi selengkap nya, lihat EXPLICIT_IDS.
		• ID transaksi,`xid`, yang memperkenalkan risiko.
		• `timestamp` Saat perintah COPY berjalan.
		
		Contoh berikut menunjukkan nilai-nilai di bidang.
		
		`1|1107|2019-06-22 07:16:11.292952`

vacuum_sort_benefit	numerik (12,2)	Perkiraan persentase maksimum peningkatan kinerja kueri pemindaian saat Anda menjalankan pengurutan vakum.

Nama kolom	Jenis data	Deskripsi
create_time	stempel waktu tanpa zona waktu	Stempel waktu untuk saat tabel dibuat.

Kueri Sampel

Contoh berikut menunjukkan pengkodean, gaya distribusi, penyortiran, dan kemiringan data untuk semua tabel yang ditentukan pengguna dalam database. Di sini, “tabel” harus dilampirkan dalam tanda kutip ganda karena itu adalah kata yang dicadangkan.

```
select "table", encoded, diststyle, sortkey1, skew_sortkey1, skew_rows
from svv_table_info
order by 1;

table          | encoded | diststyle       | sortkey1     | skew_sortkey1 | skew_rows
---------------+---------+-----------------+--------------+---------------+----------
category       | N       | EVEN            |              |               |
date           | N       | ALL             | dateid       |          1.00 |
event          | Y       | KEY(eventid)    | dateid       |          1.00 |      1.02
listing        | Y       | KEY(listid)     | dateid       |          1.00 |      1.01
sales          | Y       | KEY(listid)     | dateid       |          1.00 |      1.02
users          | Y       | KEY(userid)     | userid       |          1.00 |      1.01
venue          | N       | ALL             | venueid      |          1.00 |
(7 rows)
```

SVV_TABEL

Gunakan SVV_TABLES untuk melihat tabel di katalog lokal dan eksternal.

SVV_TABLES terlihat oleh semua pengguna secara default. Untuk mengontrol akses ke metadata database Anda, aktifkan keamanan metadata untuk cluster yang disediakan atau grup kerja tanpa server. Keamanan metadata memungkinkan Anda memisahkan izin tampilan untuk metadata objek berdasarkan pengguna dan peran. Untuk informasi selengkapnya, lihat Keamanan metadata.

Untuk informasi selengkapnya tentang praktik terbaik saat menanyakan tabel sistem dan tampilan, lihat Ku eri Tabel Sistem.

> **Note**
>
> Gunakan perintah SHOW TABLES untuk penemuan tabel. SHOW TABLES bekerja secara konsisten di seluruh konteks katalog lokal, datashare, dan eksternal dan diperbarui saat fitur baru dirilis. Untuk informasi selengkapnya, lihat Praktik terbaik untuk menemukan metadata.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
table_catalog	text	Nama katalog tempat tabel ada.
table_schema	text	Nama skema untuk tabel.
table_name	text	Nama tabel.
table_type	text	Jenis meja. Nilai yang mungkin adalah tampilan, tabel eksternal, dan tabel dasar.
komentar	text	Keterangan.

SVV_TRANSAKSI

Merekam informasi tentang transaksi yang saat ini menyimpan kunci pada tabel dalam database. Gunakan tampilan SVV_TRANSACTIONS untuk mengidentifikasi transaksi terbuka dan mengunci masalah pertentangan. Untuk informasi lebih lanjut tentang kunci, lihat Mengelola operasi penulisan bersamaan danKUNCI.

SVV_TRANSACTIONS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
txn_pemilik	text	Nama pemilik transaksi.
txn_db	text	Nama database yang terkait dengan transaksi.
xid	bigint	ID Transaksi.
pid	integer	ID proses yang terkait dengan kunci.
txn_start	timestamp	Waktu mulai transaksi.
kunci_mode	text	Nama mode kunci yang dipegang atau diminta oleh proses ini. Jika `lock_mode` is `ExclusiveLock` and `granted` is true (`t`), maka ID transaksi ini adalah transaksi terbuka.
tipe_objek yang dapat dikunci	text	Jenis objek yang meminta atau menahan kunci, baik `relation` jika itu adalah meja atau `transactionid` jika itu adalah transaksi.
relasi	integer	ID tabel untuk tabel (relasi) memperoleh kunci. Nilai ini adalah NULL jika `lockable_object_type` ada`transactionid` .

Nama kolom	Jenis data	Deskripsi
diberikan	boolean	Nilai yang menunjukkan apakah kunci telah diberikan (t) atau sedang tertunda (f).

Kueri Sampel

Perintah berikut menunjukkan semua transaksi aktif dan kunci yang diminta oleh setiap transaksi.

```
select * from svv_transactions;

 txn_
 lockable_
 owner | txn_db |  xid   |  pid  |         txn_start          |      lock_mode      |
 object_type    | relation | granted
-------+--------+--------+-------+----------------------------+---------------------
+----------------+----------+---------
 root  | dev    | 438484 | 22223 | 2016-03-02 18:42:18.862254 | AccessShareLock     |
 relation       |   100068 | t
 root  | dev    | 438484 | 22223 | 2016-03-02 18:42:18.862254 | ExclusiveLock       |
 transactionid  |          | t
 root  | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | AccessShareLock     |
 relation       |    50860 | t
 root  | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | AccessShareLock     |
 relation       |    52310 | t
 root  | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | ExclusiveLock       |
 transactionid  |          | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100068 | f
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | RowExclusiveLock    |
 relation       |    16688 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessShareLock     |
 relation       |   100064 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100166 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100171 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100190 | t
```

```
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | ExclusiveLock       |
 transactionid  |          | t
(12 rows)

(12 rows)
```

SVV_USER_GRANT

Gunakan SVV_USER_GRANTS untuk melihat daftar pengguna yang secara eksplisit diberikan peran dalam cluster.

SVV_USER_GRANTS dapat dilihat oleh pengguna berikut:

- Superuser
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna lain hanya dapat melihat peran yang secara eksplisit diberikan kepada mereka.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna untuk pengguna.
user_name	text	Nama pengguna.
rol_id	integer	ID peran untuk peran yang diberikan.
role_name	text	Nama peran untuk peran yang diberikan.
admin_option	boolean	Nilai yang menunjukkan apakah pengguna dapat memberikan peran kepada pengguna dan peran lain.

Kueri Sampel

Kueri berikut memberikan peran kepada pengguna dan menampilkan daftar pengguna yang secara eksplisit diberikan peran.

```
GRANT ROLE role1 TO reguser;
GRANT ROLE role2 TO reguser;
GRANT ROLE role1 TO superuser;
GRANT ROLE role2 TO superuser;

SELECT user_name,role_name,admin_option FROM svv_user_grants;

 user_name | role_name | admin_option
-----------+-----------+--------------
 superuser |  role1    | False
 reguser   |  role1    | False
 superuser |  role2    | False
  reguser  |  role2    | False
```

SVV_USERINFO

Anda dapat mengambil data tentang pengguna database Amazon Redshift dengan tampilan SVV_USER_INFO.

SVV_USER_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_name	text	Nama pengguna untuk peran tersebut.
user_id	integer	ID pengguna untuk pengguna.
dibuatb	boolean	Nilai yang menunjukkan apakah pengguna memiliki izin untuk membuat database.
pengguna super	boolean	Nilai yang menunjukkan apakah pengguna adalah superuser.
katalog_update	boolean	Nilai yang menunjukkan apakah pengguna dapat memperbarui katalog sistem.

Nama kolom	Jenis data	Deskripsi
koneksi_batas	text	Jumlah koneksi yang dapat dibuka pengguna.
syslog_access	text	Nilai yang menunjukkan apakah pengguna memiliki akses ke log sistem. Dua nilai yang mungkin adalah `RESTRICTED` dan`UNRESTRICTED`. `RESTRICTED` berarti bahwa pengguna yang bukan pengguna super dapat melihat catatan mereka sendiri. `UNRESTRICTED` berarti bahwa pengguna yang bukan pengguna super dapat melihat semua catatan dalam tampilan sistem dan tabel yang mereka miliki `SELECT` hak istimewa.
last_ddl_stempel waktu	timestamp	Stempel waktu untuk bahasa definisi data terakhir (DDL) membuat pernyataan yang dijalankan oleh pengguna.
session_timeout	integer	Waktu maksimum dalam detik sesi tetap tidak aktif atau idle sebelum waktu habis. 0 menunjukkan bahwa tidak ada batas waktu yang ditetapkan. Untuk informasi tentang pengaturan batas waktu idle atau tidak aktif cluster, lihat Ku ota dan batas di Amazon Redshift di Panduan Manajemen Amazon Redshift.
user_id eksternal	text	Pengidentifikasi unik pengguna di penyedia identitas pihak ketiga.

Kueri Sampel

Perintah berikut mengambil informasi pengguna dari SVV_USER_INFO.

```
SELECT * FROM SVV_USER_INFO;
```

SVV_VACUUM_PROGRES

Tampilan ini mengembalikan perkiraan berapa banyak waktu yang diperlukan untuk menyelesaikan operasi vakum yang saat ini sedang berlangsung.

SVV_VACUUM_PROGRESS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_VACUUM_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Untuk informasi tentang SVV_VACUUM_SUMMARY, lihat. SVV_VACUUM_RINGKASAN

Untuk informasi tentang SVL_VACUUM_PERCENTAGE, lihat. SVL_VACUUM_PERCENTAGE

> **Note**
>
> Tampilan ini hanya tersedia saat menanyakan klaster yang disediakan.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
table_name	text	Nama tabel yang saat ini sedang disedot debu, atau tabel yang terakhir disedot jika tidak ada operasi yang sedang berlangsung.
status	text	Deskripsi aktivitas saat ini yang dilakukan sebagai bagian dari operasi vakum: • Inisialisasi • Urutkan • Gabungkan • Delete • Pilih • Gagal • Selesai • Dilewati

Nama kolom	Jenis data	Deskripsi
		• Membangun pesanan INTERLEAVED SORTKEY
perkiraan waktu_yang tersisa	text	Perkiraan waktu tersisa untuk menyelesaikan operasi vakum saat ini, dalam menit dan detik:**`5m 10s`**, misalnya. Perkiraan waktu tidak dikembalikan sampai vakum menyelesaikan operasi pengurutan pertamanya. Jika tidak ada vakum yang sedang berlangsung, vakum terakhir yang dilakukan ditampilkan **`Completed`** di kolom STATUS dan kolom TIME_REMAINING_ESTIMATE kosong. Perkiraan biasanya menjadi lebih akurat saat vakum berlangsung.

Kueri Sampel

Kueri berikut, jalankan beberapa menit terpisah, menunjukkan bahwa tabel besar bernama SALESNEW sedang disedot.

```
select * from svv_vacuum_progress;

table_name    |            status             | time_remaining_estimate
--------------+-------------------------------+-------------------------
salesnew      |  Vacuum: initialize salesnew  |
(1 row)
...
select * from svv_vacuum_progress;

table_name   |         status         | time_remaining_estimate
-------------+------------------------+-------------------------
salesnew     |  Vacuum salesnew sort  | 33m 21s
(1 row)
```

Kueri berikut menunjukkan bahwa tidak ada operasi vakum yang sedang berlangsung. Meja terakhir yang akan disedot adalah meja PENJUALAN.

```
select * from svv_vacuum_progress;

table_name   |  status  | time_remaining_estimate
-------------+----------+-------------------------
```

```
  sales      | Complete |
(1 row)
```

SVV_VACUUM_RINGKASAN

Tampilan SVV_VACUUM_SUMMARY bergabung dengan tabel STL_VACUUM, STL_QUERY, dan STV_TBL_PERM untuk meringkas informasi tentang operasi vakum yang dicatat oleh sistem. Tampilan mengembalikan satu baris per tabel per transaksi vakum. Tampilan mencatat waktu operasi yang telah berlalu, jumlah partisi pengurutan yang dibuat, jumlah kenaikan penggabungan yang diperlukan, dan delta dalam jumlah baris dan blok sebelum dan sesudah operasi dilakukan.

SVV_VACUUM_SUMMARY hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_VACUUM_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Untuk informasi tentang SVV_VACUUM_PROGRESS, lihat. SVV_VACUUM_PROGRES

Untuk informasi tentang SVL_VACUUM_PERCENTAGE, lihat. SVL_VACUUM_PERCENTAGE

> **Note**
>
> Tampilan ini hanya tersedia saat menanyakan klaster yang disediakan.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
table_name	text	Nama meja yang disedot debu.
xid	bigint	ID transaksi operasi VACUUM.
urutkan partisi	bigint	Jumlah partisi yang diurutkan dibuat selama fase pengurutan operasi vakum.

Nama kolom	Jenis data	Deskripsi
merge_increments	bigint	Jumlah kenaikan penggabungan yang diperlukan untuk menyelesaikan fase penggabungan operasi vakum.
waktu yang telah berlalu	bigint	Waktu proses operasi vakum yang telah berlalu (dalam mikrodetik).
row_delta	bigint	Perbedaan jumlah total baris tabel sebelum dan sesudah vakum.
sortedrow_delta	bigint	Perbedaan jumlah baris tabel yang diurutkan sebelum dan sesudah vakum.
block_delta	integer	Perbedaan jumlah blok untuk tabel sebelum dan sesudah vakum.
max_merge_partisi	integer	Kolom ini digunakan untuk analisis kinerja dan mewakili jumlah maksimum partisi yang dapat diproses vakum untuk tabel per iterasi fase penggabungan. (Vacuum mengurutk an wilayah yang tidak disortir menjadi satu atau lebih partisi yang diurutkan. Bergantung pada jumlah kolom dalam tabel dan konfigurasi Amazon Redshift saat ini, fase penggabungan dapat memproses jumlah maksimum partisi dalam satu iterasi penggabungan. Fase penggabungan akan tetap berfungsi jika jumlah partisi yang diurutkan melebihi jumlah maksimum partisi penggabungan, tetapi lebih banyak iterasi penggabun gan akan diperlukan.)

Contoh kueri

Kueri berikut mengembalikan statistik untuk operasi vakum pada tiga tabel yang berbeda. Meja PENJUALAN disedot dua kali.

```
select table_name, xid, sort_partitions as parts, merge_increments as merges,
elapsed_time, row_delta, sortedrow_delta as sorted_delta, block_delta
```

```
from svv_vacuum_summary
order by xid;

table_  | xid  |parts|merges| elapsed_ | row_    | sorted_ | block_
name    |      |     |      | time     | delta   | delta   | delta
--------+------+-----+------+----------+---------+---------+--------
users   | 2985 |   1 |    1 | 61919653 |       0 |   49990 |      20
category| 3982 |   1 |    1 | 24136484 |       0 |      11 |       0
sales   | 3992 |   2 |    1 | 71736163 |       0 | 1207192 |      32
sales   | 4000 |   1 |    1 | 15363010 | -851648 | -851648 |    -140
(4 rows)
```

Tampilan pemantauan SYS

Tampilan pemantauan adalah tampilan sistem di Amazon Redshift yang digunakan untuk memantau kueri dan penggunaan sumber daya beban kerja cluster yang disediakan dan kelompok kerja tanpa server. Tampilan ini terletak di `pg_catalog` skema. Untuk menampilkan informasi yang disediakan oleh tampilan ini, jalankan pernyataan SQL SELECT.

Kecuali dinyatakan lain, tampilan ini tersedia untuk cluster Amazon Redshift dan grup kerja Amazon Redshift Serverless.

SYS_SERVERLESS_USAGE mengumpulkan data penggunaan untuk Amazon Redshift Serverless saja.

Topik

- SYS_ANALIS_RIWAYAT KOMPRESI
- SYS_ANALIS_RIWAYAT
- LOG KEBIJAKAN SYS_APLIKASI_MASKING_MASKING_
- SYS_OTOMATIS_OPTIMASI
- SYS_AUTO_TABLE_OPTIMASI
- SYS_CHILD_QUERY_TEXT
- SYS_CONNECTION_LOG
- SYS_COPY_JOB
- SYS_COPY_JOB_DETAIL
- SYS_COPY_JOB_INFO

- SYS_COPY_PENGGANTIAN
- SYS_DATASHARE_CHANGE_LOG
- SYS_DATASHARE_CROSS_REGION_PENGGUNAAN
- SYS_DATASHARE_PENGGUNAAN_KONSUMEN
- SYS_DATASHARE_PENGGUNAAN_PRODUSEN
- SYS_EXTERNAL_QUERY_DETAIL
- SYS_EXTERNAL_QUERY_ERROR
- SYS_EXTRA_COMPUTE_UNTUK_OPTIMASI_OTOMATIS
- SYS_INTEGRASI_AKTIVITAS
- SYS_INTEGRASI_TABEL_AKTIVITAS
- SYS_INTEGRASI_TABEL_STATE_PERUBAHAN
- SYS_LOAD_DETAIL
- SYS_LUDF_DETAIL
- SYS_LOAD_ERROR_DETAIL
- SYS_LOAD_HISTORY
- SYS_MV_REFRESH RIWAYAT
- SYS_MV_STATUS
- SYS_PROSEDUR_PANGGILAN
- SYS_PROSEDUR_PESAN-PESAN
- SYS_QUERY_DETAIL
- SYS_QUERY_JELASKAN
- SYS_QUERY_HISTORY
- SYS_QUERY_TEKS
- SYS_REDSHIFT_TEMPLATE
- SYS_RESTORE_LOG
- SYS_RESTORE_STATE
- SYS_SCHEMA_KUOTA_PELANGGARAN
- PENGGUNAAN_SYS_TANPA SERVER
- SYS_SESSION_RIWAYAT
- SYS_SPATIAL_SIMPLIFIKASI

- SYS_STREAM_SCAN_ERROR
- SYS_STREAM_SCAN_STATE
- SYS_RIWAYAT TRANSAKSI_
- SYS_UDF_LOG
- SYS_UNLOAD_DETAIL
- SYS_UNLOAD_HISTORY
- SYS_USERLOG
- SYS_VACUUM_HISTORY

SYS_ANALIS_RIWAYAT KOMPRESI

Merekam detail untuk operasi analisis kompresi selama perintah COPY atau ANALYSIS COMPRESSION.

SYS_ANALYZE_COMPRESSION_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang membuat entri.
waktu mulai	timestamp	Waktu ketika operasi analisis kompresi dimulai.
id transaksi_	bigint	ID transaksi operasi analisis kompresi.
tabel_id	integer	ID tabel dari tabel yang dianalisis.
table_name	karakter (128)	Nama tabel yang dianalisis.
posisi_kolom	integer	Indeks kolom dalam tabel yang dianalisis untuk menentukan pengkodean kompresi.

Nama kolom	Jenis data	Deskripsi
penyandian lama	karakter (15)	Jenis pengkodean sebelum analisis kompresi.
penyandian baru	karakter (15)	Jenis pengkodean setelah analisis kompresi.
Mode	karakter (14)	Nilai yang mungkin adalah: PRASETEL Menentukan bahwa `new_encoding` ditentukan oleh perintah Amazon Redshift COPY berdasarkan tipe data kolom. Tidak ada data yang diambil sampelnya. PADA Menentukan bahwa `new_encoding` ditentukan oleh perintah Amazon Redshift COPY berdasarkan analisis data sampel. HANYA MENGANALISIS Menentukan bahwa `new_encoding` ditentukan oleh perintah Amazon Redshift ANALYZE COMPRESSION berdasarkan analisis data sampel. Namun, jenis pengkodean kolom yang dianalisis tidak berubah.

Kueri Sampel

Contoh berikut memeriksa rincian analisis kompresi pada `lineitem` tabel dengan perintah COPY terakhir yang dijalankan dalam sesi yang sama.

```
select transaction_id, table_id, btrim(table_name) as table_name, column_position,
 old_encoding, new_encoding, mode
from sys_analyze_compression_history
where transaction_id = (select transaction_id from sys_query_history where query_id =
 pg_last_copy_id()) order by column_position;
```

```
 transaction_id  |  table_id   | table_name | column_position |  old_encoding   |
 new_encoding   |      mode
-----------------+-------------+------------+-----------------+-----------------
+-----------------+-------------
      8196       |   248126    | lineitem   |        0        | mostly32        |
 mostly32        | ON
      8196       |   248126    | lineitem   |        1        | mostly32        | lzo
           | ON
      8196       |   248126    | lineitem   |        2        | lzo             |
 delta32k        | ON
      8196       |   248126    | lineitem   |        3        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |        4        | bytedict        |
 bytedict        | ON
      8196       |   248126    | lineitem   |        5        | mostly32        |
 mostly32        | ON
      8196       |   248126    | lineitem   |        6        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |        7        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |        8        | lzo             | zstd
           | ON
      8196       |   248126    | lineitem   |        9        | runlength       | zstd
           | ON
      8196       |   248126    | lineitem   |       10        | delta           | lzo
           | ON
      8196       |   248126    | lineitem   |       11        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |       12        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |       13        | bytedict        | zstd
           | ON
      8196       |   248126    | lineitem   |       14        | bytedict        | zstd
           | ON
      8196       |   248126    | lineitem   |       15        | text255         | zstd
           | ON
(16 rows)
```

SYS_ANALIS_RIWAYAT

Rincian log untuk operasi ANALYSIS.

SYS_ANALYZE_HISTORY hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang membuat entri.
id transaksi _	long	ID transaksi.
database_ name	karakter (30)	Nama basis data.
table_name	karakter (30)	Nama tabel.
tabel_id	integer	ID tabel.
is_otomatis	harimau (1)	Nilai true (t) jika operasi menyertakan operasi Amazon Redshift ANALYZE secara default. Nilai false (f) jika perintah ANALYZE dijalankan secara eksplisit.
status	karakter (15)	Hasil dari perintah analisis. Nilai yang mungkin adalah Penuh, Dilewati, dan PredicateColumn.
waktu mulai	timestamp	Waktu dalam UTC saat operasi ANALYZE mulai berjalan.
akhir_waktu	timestamp	Waktu dalam UTC saat operasi ANALYZE selesai berjalan.
baris	double	Jumlah total baris dalam tabel
baris-bar is yang dimodifikasi	double	Jumlah total baris yang dimodifikasi sejak operasi ANALYZE terakhir.
analisis_ ambang_pe rsen	integer	Nilai parameter analise_praghold_persen.

Nama kolom	Jenis data	Deskripsi
waktu analisa terakhir	timestamp	Waktu dalam UTC saat tabel dianalisis sebelumnya.

Kueri Sampel

```
 user_id | transaction_id | database_name | schema_name |      table_name     |
 table_id | is_automatic | Status |         start_time         |          end_time
     | rows | modified_rows | analyze_threshold_percent |  last_analyze_time
---------+----------------+---------------+-------------+---------------------
+----------+--------------+--------+----------------------------+----------
+-----------------+------+--------------+--------------------------
+---------------------
     101 |           8006 |           dev |      public | test_table_562bf8dc
 |   110427 |            f |   Full | 2023-09-21 18:33:08.504646 | 2023-09-21
 18:33:24.296498 |    5 |             5 |                          0 | 2000-01-01
 00:00:00
```

LOG KEBIJAKAN SYS_APLIKASI_MASKING_MASKING_

Gunakan SYS_APPLIED_MASKING_POLICY_LOG untuk melacak penerapan kebijakan penyembunyian data dinamis pada kueri yang mereferensikan hubungan. DDM-protected

SYS_APPLIED_MASKING_POLICY_LOG terlihat oleh pengguna berikut:

- Superuser
- Pengguna dengan `sys:operator` peran
- Pengguna dengan izin TABEL SISTEM AKSES

Pengguna biasa akan melihat 0 baris.

Perhatikan bahwa SYS_APPLIED_MASKING_POLICY_LOG tidak terlihat oleh pengguna dengan peran tersebut. `sys:secadmin`

Untuk informasi selengkapnya tentang penyamaran data dinamis, bukaPenyembunyian data dinamis.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_kebijakan	text	Nama kebijakan masking.
user_id	text	ID pengguna yang menjalankan kueri.
rekor_waktu	timestamp	Waktu entri tampilan sistem direkam.
session_id	int	ID proses.
id transaksi_	long	ID transaksi.
Query_id	int	ID kueri.
database_name	text	Nama database tempat query dijalankan.
nama relasi	text	Nama tabel yang diterapkan kebijakan masking.
schema_name	text	Nama skema tempat tabel berada.
Id lampiran	long	ID kebijakan masking terlampir.
Relation_type	text	Jenis relasi yang diterapkan kebijakan masking. Nilai yang mungkin adalah `TABLE`, `VIEW`, `LATE BINDING VIEW`, dan `MATERIALIZED VIEW` .
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Contoh berikut menunjukkan bahwa kebijakan `mask_credit_card_full` masking dilampirkan ke `credit_db.public.credit_cards` tabel.

```
select policy_name, database_name, relation_name, schema_name, relation_kind
from sys_applied_masking_policy_log;

policy_name           | database_name | relation_name | schema_name | relation_kind
----------------------+---------------+---------------+-------------+---------------
mask_credit_card_full | credit_db     | credit_cards  | public      | table

(1 row)
```

SYS_OTOMATIS_OPTIMASI

Gunakan SYS_AUTOMATIC_OPTIMIZATION untuk melihat detail tugas yang dijalankan Amazon Redshift untuk pengoptimalan otomatis, juga dikenal sebagai otonom. Untuk informasi lebih lanjut tentang pengoptimalan otomatis, lihatOptimasi database otomatis.

SYS_AUTOMATIC_OPTIMIZATION hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
session_id	integer	ID proses mengeksekusi kueri tugas. Awalnya disetel ke 0 saat tugas dibuat, nilai ini menjadi bukan nol hanya ketika `event` disetel sebagai Dimulai.
database_name	karakter (128)	Nama database tempat tugas dijalankan.
tipe_objek	karakter (30)	Jenis objek yang dioperasikan oleh tugas otonom. Nilai yang mungkin adalah sebagai berikut: • tabel • mV • tabel_kolumn_pair

Nama kolom	Jenis data	Deskripsi
id_objek	karakter (512)	Pengidentifikasi objek tempat tugas otonom beroperasi. Bidang ini dapat menampung lebih dari satu objek ketika tugas berjalan pada beberapa objek database. Dalam hal ini, pengidentifikasi dipisahkan oleh koma.
task_type	karakter (100)	Jenis tugas otonom yang dijalankan. Tugas yang mungkin adalah sebagai berikut: • AutoAlterTableTaskSortkey • AutoAlterTableTaskDistkey • VacuumSort • VacuumDelete • Analisis
kejadian	karakter (50)	Jenis peristiwa transisi keadaan untuk tugas otonom. Nilai yang mungkin adalah sebagai berikut: • Antre • Dimulai • Ditangguhkan • Selesai • Failed (Gagal)
waktu_acara	timestamp	Waktu transisi negara terjadi.
status	karakter (512)	Status eksekusi tugas pengoptimalan. Kosong jika tugas tidak mencoba dijalankan.

Nama kolom	Jenis data	Deskripsi
tipe komputasi	karakter (100)	Hitung jenis sumber daya yang digunakan oleh tugas. Nilai yang mungkin untuk disediakan adalah sebagai berikut: • utama • skala primer Nilai juga bisa kosong saat disediakan jika tugas tidak menggunakan komputasi apa pun. Untuk informasi selengkapnya tentang berbagai jenis sumber daya komputasi yang dapat Anda gunakan untuk otonom, lihat. Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis
detail_tugas	karakter (512)	Detail tugas tambahan jika ada. Bidang ini bisa kosong.

Catatan penggunaan

Kolom compute_type akan kosong untuk cluster tanpa server karena Amazon Redshift tidak membedakan sumber daya komputasi skala primer atau primer. Sumber daya komputasi cluster tanpa server diukur dengan penggunaan Redshift Processing Units (RPUs). Untuk informasi selengkapnya, lihat Kapasitas komputasi untuk Amazon Redshift Serverless.

Contoh

Kueri berikut menampilkan optimasi otomatis yang dilakukan pada tabel 155259.

```
SELECT pid, trim(task_type) as task_type,
  trim(database) as database,
  trim(status) as status,
  trim(event) as event,
  event_time
from SYS_AUTOMATIC_OPTIMIZATION
WHERE object_ids like '%155259%'
AND status = 'Task completed successfully';
```

```
 task_type  |    database    |           status            |   event   |
 event_time
------------+----------------+-----------------------------+-----------
+---------------------------
 VacuumSort | tpcds_100g_oob | Task completed successfully | Completed | 2025-12-22
 07:27:15.943018
```

Kueri berikut menunjukkan semua pengoptimalan otomatis yang dijalankan VacuumSort "". Untuk informasi lebih lanjut tentang "VacuumSort“, lihatPengurutan tabel otomatis.

```
SELECT trim(task_type) as task_type,
  trim(database) as database,
  trim(object_type) as object_type,
  trim(object_ids) as object_ids,
  trim(status) as status,
  trim(event) as event,
  event_time
from SYS_AUTOMATIC_OPTIMIZATION
WHERE task_type like '%VacuumSort%'
AND status = 'Task completed successfully';

task_type  |    database    | object_type | object_ids |           status            |
  event   |         event_time
------------+----------------+-------------+------------+-----------------------------
+-----------+----------------------------
 VacuumSort | tpcds_100g_oob | table       | 155301     | Task completed successfully |
 Completed | 2025-12-22 07:14:00.065391
 VacuumSort | tpcds_100g_oob | table       | 155303     | Task completed successfully |
 Completed | 2025-12-22 07:14:09.158251
 VacuumSort | tpcds_100g_oob | table       | 155291     | Task completed successfully |
 Completed | 2025-12-22 07:17:06.61164
 VacuumSort | tpcds_100g_oob | table       | 155293     | Task completed successfully |
 Completed | 2025-12-22 07:17:37.015069
 VacuumSort | tpcds_100g_oob | table       | 155281     | Task completed successfully |
 Completed | 2025-12-22 07:18:54.903935
 VacuumSort | tpcds_100g_oob | table       | 155279     | Task completed successfully |
 Completed | 2025-12-22 07:20:13.960002
 VacuumSort | tpcds_100g_oob | table       | 155271     | Task completed successfully |
 Completed | 2025-12-22 07:21:26.095549
 VacuumSort | tpcds_100g_oob | table       | 155267     | Task completed successfully |
 Completed | 2025-12-22 07:22:48.119249
 VacuumSort | tpcds_100g_oob | table       | 155269     | Task completed successfully |
 Completed | 2025-12-22 07:24:12.010424
```

```
 VacuumSort | tpcds_100g_oob | table       | 155263     | Task completed successfully |
 Completed | 2025-12-22 07:25:35.958388
 VacuumSort | tpcds_100g_oob | table       | 155265     | Task completed successfully |
 Completed | 2025-12-22 07:26:40.580395
 VacuumSort | tpcds_100g_oob | table       | 155259     | Task completed successfully |
 Completed | 2025-12-22 07:27:15.943018
(12 rows)
```

SYS_AUTO_TABLE_OPTIMASI

Merekam tindakan otomatis yang dilakukan oleh Amazon Redshift pada tabel yang ditentukan untuk pengoptimalan otomatis.

SYS_AUTO_TABLE_OPTIMIZATION hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
id transaksi_	long	Pengidentifikasi transaksi.
session_id	int	Pengidentifikasi sesi dari proses yang menjalankan perintah alter.
tabel_id	int	Pengidentifikasi tabel.
alter_tipe_tabel	karakter (32)	Jenis rekomendasi. Nilai yang mungkin adalah `distkey`, `sortkey`, dan `encode`.
status	karakter (128)	Status penyelesaian rekomendasi. Peluang nilai adalah `Start`, `Complete`, `Skipped`, `Abort`, `Checkpoint` , dan `Failed`.
waktu_acara	timestamp	Stempel waktu kolom status.
alter_dari	karakter (200)	Gaya distribusi sebelumnya dan kunci pengurutan tabel sebelum menerapkan rekomendasi. Nilai dipotong menjadi penambahan 200 karakter.

Kueri Sampel

Dalam contoh berikut, baris dalam hasil menunjukkan tindakan yang diambil oleh Amazon Redshift.

```
SELECT table_id, alter_table_type, status, event_time, alter_from
FROM SYS_AUTO_TABLE_OPTIMIZATION;

 table_id |  alter_table_type   |                        status
 |         event_time          |  alter_from
----------+---------------------+-----------------------------------------------------
+-----------------------------+-----------------
   118082 | sortkey             | Start
 | 2020-08-22 19:42:20.727049  |
   118078 | sortkey             | Start
 | 2020-08-22 19:43:54.728819  |
   118082 | sortkey             | Start
 | 2020-08-22 19:42:52.690264  |
   118072 | sortkey             | Start
 | 2020-08-22 19:44:14.793572  |
   118082 | sortkey             | Failed
 | 2020-08-22 19:42:20.728917  |
   118078 | sortkey             | Complete
 | 2020-08-22 19:43:54.792705  |  SORTKEY: None;
   118086 | sortkey             | Complete
 | 2020-08-22 19:42:00.72635   |  SORTKEY: None;
   118082 | sortkey             | Complete
 | 2020-08-22 19:43:34.728144  |  SORTKEY: None;
   118072 | sortkey             | Skipped:Retry exceeds the maximum limit for a table.
 | 2020-08-22 19:44:46.706155  |
   118086 | sortkey             | Start
 | 2020-08-22 19:42:00.685255  |
   118082 | sortkey             | Start
 | 2020-08-22 19:43:34.69531   |
   118072 | sortkey             | Start
 | 2020-08-22 19:44:46.703331  |
   118082 | sortkey             | Checkpoint: progress 14.755079%
 | 2020-08-22 19:42:52.692828  |
   118072 | sortkey             | Failed
 | 2020-08-22 19:44:14.796071  |
   116723 | sortkey             | Abort:This table is not AUTO.
 | 2020-10-28 05:12:58.479233  |
   110203 | distkey             | Abort:This table is not AUTO.
 | 2020-10-28 05:45:54.67259   |
```

SYS_CHILD_QUERY_TEXT

Mengembalikan teks SQL dari kueri anak.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
Query_id	bigint	ID kueri pengguna
urutan kueri anak	integer	Urutan permintaan pengguna yang ditulis ulang, dimulai dengan 1.
urutan	integer	Nomor urutan untuk bagian kueri ini.
text	karakter (200)	200 karakter pertama dari teks query SQL.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Dalam contoh berikut, baris dalam hasil menunjukkan tindakan yang diambil oleh Amazon Redshift.

```
SELECT * from sys_child_query_text where query_id = '34487366' order by
 child_query_sequence asc, sequence asc;

user_id | query_id | child_query_sequence | sequence | text
--------|----------|----------------------|----------|-----------------------------------------
100     | 34899339 |   1                  |    0     |  /* RQEV2-aY6ZZ1ZpQK */\nwith
 venue as (\n    select venueid,\n            venuename,\n            venuestate\n
 from venue\n), event as (\n    select eventid,\n            venueid,\n            date

100     | 34899339 |   1                  |    1     |  id,\n            eventname\n
  from event\n    where eventname like '3 Doors Down'\n), users as (\n    select userid
\n    from users\n), sales as (\n    select salesid,\n            pricepaid,
```

```
100     | 34899339 |   1                 |    2     |  \n            eventid,\n
       buyerid\n    from sales\n)\nselect e.eventname,\n        v.venuename,\n
 count(distinct(u.userid)) as unique_customers,\n        sum(s.pricepaid) as total_sal

100     | 34899339 |   1                 |    3     |  es\nfrom venue as v inner join
 event e on v.venueid = e.venueid\ninner join sales s on e.eventid = s.eventid inner
 join users u on s.buyerid = u.userid\ngroup by 1,2\norder by 4 desc limit 100
```

SYS_CONNECTION_LOG

Mencatat upaya otentikasi dan koneksi serta pemutusan.

SYS_CONNECTION_LOG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kejadian	karakter (50)	Acara koneksi atau otentikasi.
rekor_waktu	timestamp	Waktu peristiwa itu terjadi.
host jarak jauh	karakter (45)	Nama atau alamat IP host jarak jauh.
pelabuhan jarak jauh	karakter (32)	Nomor port untuk host jarak jauh.
session_id	integer	ID proses yang terkait dengan pernyataan.
database_name	karakter (50)	Nama database.
user_name	karakter (50)	Nama pengguna.
auth_method	karakter (32)	Metode otentikasi.

Nama kolom	Jenis data	Deskripsi
durasi	integer	Durasi koneksi dalam mikrodetik.
ssl_versi	karakter (50)	Versi Secure Sockets Layer (SSL). Kolom ini kosong saat SSL dinonaktifkan.
ssl_cipher	karakter (128)	Sandi SSL.
mtu	integer	Unit transmisi maksimum (MTU).
ssl_kompresi	karakter (64)	Jenis kompresi SSL.
ssl_ekspansi	karakter (64)	Jenis ekspansi SSL.
iam_auth_guide	karakter (36)	ID otentikasi IAM untuk CloudTrail permintaan.
application_name	karakter (250)	Nama awal atau yang diperbarui dari aplikasi untuk sesi.
versi_driver	karakter (64)	Versi driver ODBC atau JDBC yang terhubung ke cluster Amazon Redshift Anda dari alat klien SQL pihak ketiga Anda.
os_version	karakter (64)	Versi sistem operasi yang ada di mesin klien yang terhubung ke cluster Amazon Redshift Anda.
nama plugin	karakter (32)	Nama plugin yang digunakan untuk terhubung ke cluster Amazon Redshift Anda.

Nama kolom	Jenis data	Deskripsi
versi_protokol	integer	Versi protokol internal yang digunakan driver Amazon Redshift saat membuat koneksi dengan server. Versi protokol dinegosiasikan antara driver dan server. Versi ini menjelaskan fitur yang tersedia. Nilai yang valid meliputi: • 0 (VERSI_PROTOKOL_SERVER DASAR) • 1 (EXTENDED_RESULT_METADATA_SERVER_PROTOCOL_VERSION) — Untuk menyimpan perjalanan pulang pergi per kueri, server mengirimkan informasi metadata set hasil tambahan. • 2 (BINARY_PROTOCOL_VERSION) - Tergantung pada tipe data dari kumpulan hasil, server mengirimkan data dalam format biner. • 3 (EXTENDED2_RESULT_METADATA_SERVER_PROTOCOL_VERSION) - Server mengirimkan informasi sensitivitas huruf besar (kolasi) dari kolom.
global_session_id	karakter (36)	Pengidentifikasi unik global untuk sesi saat ini. ID sesi tetap ada melalui kegagalan node restart.

Kueri Sampel

Untuk melihat detail untuk koneksi terbuka, jalankan kueri berikut.

```
select record_time, user_name, database_name, remote_host, remote_port
from sys_connection_log
where event = 'initiating session'
and session_id not in
(select session_id from sys_connection_log
where event = 'disconnecting session')
order by 1 desc;

record_time         | user_name   | database_name   | remote_host   | remote_port
```

```
--------------------+-------------+-----------------+---------------
+--------------------------------
2014-11-06 20:30:06 | rdsdb       | dev             | [local]       |

2014-11-06 20:29:37 | test001     | test            | 10.49.42.138  | 11111

2014-11-05 20:30:29 | rdsdb       | dev             | 10.49.42.138  | 33333

2014-11-05 20:28:35 | rdsdb       | dev             | [local]       |
(4 rows)
```

Contoh berikut mencerminkan upaya otentikasi yang gagal dan koneksi dan pemutusan yang berhasil.

```
select event, record_time, remote_host, user_name
from sys_connection_log order by record_time;

            event      |         record_time        |  remote_host  | user_name

-----------------------+----------------------------+---------------+---------
authentication failure | 2012-10-25 14:41:56.96391  | 10.49.42.138  | john

authenticated          | 2012-10-25 14:42:10.87613  | 10.49.42.138  | john

initiating session     | 2012-10-25 14:42:10.87638  | 10.49.42.138  | john

disconnecting session  | 2012-10-25 14:42:19.95992  | 10.49.42.138  | john

(4 rows)
```

Contoh berikut menunjukkan versi driver ODBC, sistem operasi pada mesin klien, dan plugin yang digunakan untuk terhubung ke cluster Amazon Redshift. Dalam contoh ini, plugin yang digunakan adalah untuk otentikasi driver ODBC standar menggunakan nama login dan kata sandi.

```
select driver_version, os_version, plugin_name from sys_connection_log;

driver_version                          |  os_version                       |
 plugin_name
----------------------------------------+-----------------------------------
+--------------------
Amazon Redshift ODBC Driver 1.4.15.0001 | Darwin 18.7.0 x86_64              | none
Amazon Redshift ODBC Driver 1.4.15.0001 | Linux 4.15.0-101-generic x86_64   | none
```

Contoh berikut menunjukkan versi sistem operasi pada mesin klien, versi driver, dan versi protokol.

```
select os_version, driver_version, protocol_version from sys_connection_log;

os_version                     |  driver_version              | protocol_version
-------------------------------+------------------------------+-----------------
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
```

SYS_COPY_JOB

Gunakan SYS_COPY_JOB untuk melihat detail perintah COPY JOB.

Tampilan ini berisi perintah COPY JOB yang telah dibuat.

SYS_COPY_JOB terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
id pekerjaan	bigint	Pengidentifikasi pekerjaan menyalin.
nama_pekerjaan	karakter (128)	Nama pekerjaan penyalinan.
iam_role	karakter (128)	Peran IAM yang ditentukan dalam pernyataan COPY.
teks_pekerjaan	karakter (256)	Parameter pernyataan COPY.
is_auto	integer	Menunjukkan apakah COPY JOB dijalankan secara otomatis oleh Amazon Redshift. A 1 menunjukkan benar, 0 menunjukkan salah.

Nama kolom	Jenis data	Deskripsi
on_error_suspensi	integer	Informasi ini hanya untuk penggunaan internal.

SYS_COPY_JOB_DETAIL

Gunakan SYS_COPY_JOB_DETAIL untuk melihat detail perintah COPY JOB.

Tampilan ini berisi perintah COPY JOB yang telah dibuat. Jika COPY JOB mencoba memuat file dan gagal dimuat, file tersebut dilewati pada upaya COPY JOB otomatis di masa mendatang.

SYS_COPY_JOB_DETAIL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang memiliki pekerjaan.
database_name	text	Nama database yang berisi pekerjaan.
id pekerjaan	integer	Pengidentifikasi pekerjaan menyalin.
nama_pekerjaan	text	Nama pekerjaan penyalinan.
file_location	text	Nama bucket Amazon S3 untuk entri tersebut.
file_name	text	Nama kunci Amazon S3 untuk entri.
ukuran file	bigint	Ukuran entri dalam byte.

Nama kolom	Jenis data	Deskripsi
tag file	text	Tag entitas Amazon S3 (ETag) untuk entri.
teks_pekerjaan	karakter (256)	Parameter pernyataan COPY.
waktu modifikasi	timestamp	Waktu (UTC) di mana objek S3 dibuat atau dimodifikasi memulai konsumsi. Dalam mode daftar S3, ini adalah waktu modifikasi objek. Untuk mode notifikasi, ini adalah waktu peristiwa notifikasi terjadi.
enqueue_time	timestamp	Waktu (UTC) di mana entri ditambahkan ke antrian penyerapan.
status	haru (1)	Status entri. Nilai yang valid: P untuk konsumsi yang tertun I da, E untuk dicerna, untuk kesalahan menelan, dan U untuk tidak diketahui (menunjukkan semacam keadaan tak terduga).

Contoh berikut mengembalikan satu baris tempat untuk entri yang dicerna.

```
SELECT * FROM SYS_COPY_JOB_DETAIL WHERE status ilike '%ingested%' limit 1;


user_id | 100
database_name | dev
job_name | many_job_4_3
job_id | 110702
file_location | saral-sqs-system4623202051-0
file_name | frenzy-9/4623202051/file_0_107
```

```
file_size | 11302
file_etag | 51b2d78ac5b5aecf4ee6f8374815ad19
modification_time | 2024-07-15 20:43:14
enqueue_time | 2024-07-15 20:44:24
status | Ingested
```

SYS_COPY_JOB_INFO

Gunakan SYS_COPY_JOB_INFO untuk melihat pesan yang dicatat tentang COPY JOB.

Tampilan ini berisi informasi tentang kesalahan dalam COPY JOB yang telah dijalankan.

SYS_COPY_JOB_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
id pekerjaan	bigint	Pengidentifikasi pekerjaan menyalin.
nama_pekerjaan	karakter (128)	Nama pekerjaan penyalinan.
database_name	karakter (128)	Nama basis data.
rekor_waktu	timestamp	Waktu (UTC) saat pesan dicatat.
pesan	kacamata (512)	Pesan peristiwa yang dicatat untuk COPY JOB ini.

SYS_COPY_PENGGANTIAN

Menampilkan log yang mencatat ketika UTF-8 karakter tidak valid diganti oleh MENYALIN perintah dengan opsi ACCEPTINVCHARS. Entri log ditambahkan ke SYS_COPY_REPLACEMENTS untuk masing-masing dari 100 baris pertama pada setiap irisan node yang memerlukan setidaknya satu penggantian.

Anda dapat menggunakan tampilan ini untuk melihat informasi tentang grup kerja tanpa server dan cluster yang disediakan.

SYS_COPY_REPLACEMENTS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang menghasilkan kueri.
Query_id	bigint	ID kueri. Kolom yang digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
tabel_id	integer	ID tabel.
file_name	karakter (256)	Jalur lengkap ke file input untuk perintah COPY.
column_name	karakter (127)	Bidang pertama yang berisi UTF-8 karakter yang tidak valid.
bilangan garis	bigint	Nomor baris dalam file data input yang berisi UTF-8 karakter yang tidak valid. -1 menunjukkan bahwa nomor baris tidak tersedia, seperti saat menyalin dari file data kolumnar.
raw_line	karakter (1024)	Data pemuatan mentah yang berisi UTF-8 karakter yang tidak valid.

Kueri Sampel

Contoh berikut mengembalikan pengganti untuk operasi COPY terbaru.

```
select query_idp, table_id, file_name, line_number, colname
from sys_copy_replacements
where query = pg_last_copy_id();
```

```
 query_id | table_id |   file_name                                                |
 line_number | column_name
 ---------+----------+------------------------------------------------------------
+-------------+--------
    96    |    26    | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt             |
 123 | city
    96    |    26    | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt             |
 456 | city
    96    |    26    | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt             |
 789 | city
    96    |    26    | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt             |
 012 | city
    96    |    26    | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt             |
 119 | city
...
```

SYS_DATASHARE_CHANGE_LOG

Merekam tampilan konsolidasi untuk melacak perubahan data pada cluster produsen dan konsumen.

SYS_DATASHARE_CHANGE_LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang mengambil tindakan.
user_name	bangkai (128)	Nama pengguna yang mengambil tindakan.
session_id	integer	ID sesi.
id transaksi_	bigint	ID transaksi.
share_id	integer	ID datashare terpengaruh.

Nama kolom	Jenis data	Deskripsi
nama_berbagi	bangkai (128)	Nama datashare.
id database_sumber	integer	ID database tempat datashare milik.
nama_database_sumber	bangkai (128)	Nama database tempat datashare milik.
id database_konsumen	integer	ID database yang diimpor dari datashare.
nama_database_konsumen	bangkai (128)	Nama database yang diimpor dari datashare.
arn	bangkai (192)	ARN sumber daya yang mendukung database yang diimpor.
rekor_waktu	timestamp	Stempel waktu tindakan.
tindakan	bangkai (128)	Tindakan yang sedang dijalankan. Nilai yang mungkin adalah CREATE DATASHARE, DROP DATASHARE, GRANT ALTER, REVOKE ALTER, GRANT SHARE, REVOKE SHARE, ALTER ADD, ALTER REMOVE, ALTER SET, GRANT USAGE, REVOKE USAGE, CREATE DATABASE, GRANT, atau REVOKE USAGE pada database bersama, DROP SHARED DATABASE, ALTER SHARED DATABASE.
status	integer	Status tindakan. Nilai yang mungkin adalah SUKSES dan ERROR-ERROR KODE.
berbagi_tipe_objek	varchar(64)	Jenis objek database yang ditambahkan atau dihapus dari datashare. Nilai yang mungkin adalah skema, tabel, kolom, fungsi, dan tampilan. Ini adalah bidang untuk cluster produsen.

Nama kolom	Jenis data	Deskripsi
share_object_id	integer	ID objek database yang ditambahkan atau dihapus dari datashare. Ini adalah bidang untuk cluster produsen.
bagian_nama_objek	bangkai (128)	Nama objek database yang ditambahkan atau dihapus dari datashare. Ini adalah bidang untuk cluster produsen.
tipe_pengguna target	varchar(16)	Jenis pengguna atau grup yang diberikan hak istimewa. Ini adalah bidang untuk cluster produsen dan konsumen.
target_user_id	integer	ID pengguna atau grup yang diberikan hak istimewa. Ini adalah bidang untuk cluster produsen dan konsumen.
target_nama_pengguna	bangkai (128)	Nama pengguna atau grup yang diberikan hak istimewa. Ini adalah bidang untuk cluster produsen dan konsumen.
akun_konsumen	varchar(16)	ID akun konsumen data. Ini adalah bidang untuk cluster produsen.
ruang nama konsumen	varchar(64)	Namespace akun konsumen data. Ini adalah bidang untuk cluster produsen.
akun_produsen	varchar(16)	ID akun dari akun produsen milik datashare. Ini adalah bidang untuk cluster konsumen.
ruang nama produsen	varchar(64)	Namespace akun produk milik datashare. Ini adalah bidang untuk cluster konsumen.
nama atribut	varchar(64)	Nama atribut datashare atau database bersama.
atribut_nilai	bangkai (128)	Nilai atribut datashare atau database bersama.
pesan	bangkai (512)	Pesan kesalahan saat tindakan gagal.

Kueri Sampel

Contoh berikut menunjukkan tampilan SYS_DATASHARE_CHANGE_LOG.

```
SELECT DISTINCT action
FROM sys_datashare_change_log
WHERE share_object_name LIKE 'tickit%';

         action
 ----------------------
  "ALTER DATASHARE ADD"
```

SYS_DATASHARE_CROSS_REGION_PENGGUNAAN

Gunakan tampilan SYS_DATASHARE_CROSS_REGION_USAGE untuk mendapatkan ringkasan penggunaan data lintas wilayah yang ditransfer yang disebabkan oleh kueri pembagian data lintas wilayah. SYS_DATASHARE_CROSS_REGION_USAGE mengumpulkan detail di tingkat segmen.

SYS_DATASHARE_CROSS_REGION_USAGE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Query_id	integer	ID kueri. Gunakan nilai ini untuk bergabung dengan tabel dan tampilan sistem lainnya.
urutan kueri anak	integer	Urutan permintaan pengguna yang ditulis ulang, dimulai dengan 1.
segmen_id	bigint	Jumlah segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
waktu mulai	Waktu	Waktu di UTC dimana transfer data dimulai.
akhir_waktu	Waktu	Waktu dalam UTC saat transfer data berakhir.

Nama kolom	Jenis data	Deskripsi
data yang ditransfer	bigint	Jumlah byte data yang ditransfer dari Wilayah produsen ke Wilayah konsumen.
sumber_wilayah	arang (25)	Wilayah produsen tempat kueri mentransfer data.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Contoh berikut menunjukkan tampilan SYS_DATASHARE_CROSS_REGION_USAGE.

```
SELECT query_id, segment_id, transferred_data, source_region
from sys_datashare_cross_region_usage
where query_id = pg_last_query_id()
order by query_id, segment_id;

  query_id | segment_id | transferred_data | source_region
-----------+------------+------------------+--------------
    200048 |          2 |          4194304 |    us-west-1
    200048 |          2 |          4194304 |    us-east-2
```

SYS_DATASHARE_PENGGUNAAN_KONSUMEN

Merekam aktivitas dan penggunaan datashares. Pandangan ini hanya relevan pada cluster konsumen.

SYS_DATASHARE_USAGE_CONSUMER dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang mengeluarkan permintaan.
session_id	integer	ID proses pemimpin yang menjalankan kueri.
id transaksi_	bigint	Konteks transaksi saat ini.
id permintaan	Washer (50)	ID unik dari panggilan API yang diminta.
tipe_permintaan	bangkai (25)	Jenis permintaan yang dibuat ke cluster produsen.
transaksi_uid	Washer (50)	ID unik dari transaksi.
waktu_rekor	timestamp	Waktu ketika tindakan direkam.
status	integer	Status panggilan API yang diminta.
kesalahan	bangsawa (512)	Pesan untuk kesalahan.

Kueri Sampel

Contoh berikut menunjukkan tampilan SYS_DATASHARE_USAGE_CONSUMER.

```
SELECT request_type, status, trim(error) AS error
FROM sys_datashare_usage_consumer

  request_type  | status | error_message
----------------+--------+---------------
 "GET RELATION" |   0    |
```

SYS_DATASHARE_PENGGUNAAN_PRODUSEN

Merekam aktivitas dan penggunaan datashares. Pandangan ini hanya relevan pada cluster produsen.

SYS_DATASHARE_USAGE_PRODUCER terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
share_id	integer	ID objek (OID) dari datashare.
nama_berbagi	bangkai (128)	Nama datashare.
id permintaan	Washer (50)	ID unik dari panggilan API yang diminta.
tipe_permintaan	bangkai (25)	Jenis permintaan yang dibuat ke cluster produsen.
tipe_objek	varchar(64)	Jenis objek yang dibagikan dari datashare. Nilai yang mungkin adalah skema, tabel, kolom, fungsi, dan tampilan.
objek_oid	integer	ID objek yang dibagikan dari datashare.
nama_objek	bangkai (128)	Nama objek yang dibagikan dari datashare.
akun_konsumen	varchar(16)	Akun akun konsumen tempat datashare dibagikan.
ruang nama konsumen	varchar(64)	Namespace akun konsumen tempat datashare dibagikan.

Nama kolom	Jenis data	Deskripsi
uid transaksi_konsumen	Washer (50)	ID transaksi unik dari pernyataan pada cluster konsumen.
waktu_rekor	timestamp	Waktu ketika tindakan direkam.
status	integer	Status datashare.
kesalahan	bangsawa (512)	Pesan untuk kesalahan.
wilayah konsumen	karakter (64)	Wilayah tempat cluster konsumen berada.

Kueri Sampel

Contoh berikut menunjukkan tampilan SYS_DATASHARE_USAGE_PRODUCER.

```
SELECT DISTINCT
FROM sys_datashare_usage_producer
WHERE object_name LIKE 'tickit%';

   request_type
 ------------------
   "GET RELATION"
```

SYS_EXTERNAL_QUERY_DETAIL

Gunakan SYS_EXTERNAL_QUERY_DETAIL untuk melihat detail untuk kueri pada tingkat segmen. Setiap baris mewakili segmen dari kueri WLM tertentu dengan detail seperti jumlah baris yang diproses, jumlah byte yang diproses, dan info partisi tabel eksternal di Amazon S3. Setiap baris dalam tampilan ini juga akan memiliki entri yang sesuai dalam tampilan SYS_QUERY_DETAIL, kecuali tampilan ini memiliki informasi lebih detail terkait dengan pemrosesan kueri eksternal.

SYS_EXTERNAL_QUERY_DETAIL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
Query_id	bigint	Pengidentifikasi kueri dari kueri eksternal.
id transaksi_	bigint	Pengidentifikasi transaksi.
urutan kueri anak	integer	Urutan permintaan pengguna yang ditulis ulang. Dimulai dengan 0, mirip dengan segment_id.
segmen_id	integer	Pengidentifikasi segmen dari segmen kueri.
tipe_sumber	karakter (32)	Jenis sumber data kueri, bisa `S3` untuk Redshift Spectrum, `PG` untuk kueri federasi.
waktu mulai	timestamp	Waktu ketika kueri dimulai.
akhir_waktu	timestamp	Waktu ketika kueri selesai.
durasi	bigint	Jumlah waktu (mikrodetik) yang dihabiskan untuk kueri.
total_partisi	integer	Jumlah partisi yang diperlukan kueri Amazon S3.
partisi_berkualifikasi	integer	Jumlah partisi yang dipindai kueri Amazon S3.
memindai file-file	bigint	Jumlah file Amazon S3 yang dipindai.

Nama kolom	Jenis data	Deskripsi
baris yang dikembalikan	bigint	Jumlah baris yang dipindai untuk kueri Amazon S3, atau jumlah baris yang dikembali kan untuk kueri federasi.
Byte_dikembalikan	bigint	Jumlah byte yang dipindai untuk kueri Amazon S3, atau jumlah byte yang dikembalikan untuk kueri federasi.
file_format	text	Format file dari file Amazon S3.
file_location	text	Lokasi Amazon S3 dari tabel eksternal.
teks_kueri eksternal	text	Teks kueri tingkat segmen untuk kueri federasi.
warning_message	karakter (4000)	Pesan peringatan ditampilkan saat kueri berjalan.
table_name	karakter (136)	Nama tabel langkah yang sedang dioperasikan.
is_rekursif	karakter (1)	Menunjukkan apakah ada pemindaian rekursif untuk subfolder.
is_bersarang	karakter (1)	Menunjukkan apakah tipe data kolom bersarang diakses.
s3list_time	bigint	Durasi daftar file dalam milidetik.

Nama kolom	Jenis data	Deskripsi
dapatkan_partisi_waktu_	long	Waktu yang dihabiskan untuk membuat daftar dan memenuhi syarat partisi untuk objek eksternal tertentu dari AWS Glue Data Catalog dan Apache Hive.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Kueri berikut menunjukkan detail kueri eksternal.

```
SELECT query_id,
       segment_id,
       start_time,
       end_time,
       total_partitions,
       qualified_partitions,
       scanned_files,
       returned_rows,
       returned_bytes,
       trim(external_query_text) query_text,
       trim(file_location) file_location
FROM sys_external_query_detail
ORDER BY query_id, start_time DESC
LIMIT 2;
```

Sampel keluaran.

```
 query_id | segment_id |         start_time         |          end_time
   | total_partitions | qualified_partitions | scanned_files | returned_rows |
 returned_bytes | query_text | file_location
----------+------------+----------------------------+----------------------------
+------------------+----------------------+---------------+---------------
+----------------+------------+---------------
```

```
 763251 |          0 | 2022-02-15 22:32:23.312448 | 2022-02-15 22:32:24.036023 |
        3 |                   3 |             3 |         38203 |        2683414 |
          |
 763254 |          0 | 2022-02-15 22:32:40.17103  | 2022-02-15 22:32:40.839313 |
        3 |                   3 |             3 |         38203 |        2683414 |
          |
```

SYS_EXTERNAL_QUERY_ERROR

Anda dapat menanyakan tampilan sistem SYS_EXTERNAL_QUERY_ERROR untuk mendapatkan informasi tentang kesalahan pemindaian Redshift Spectrum. SYS_EXTERNAL_QUERY_ERROR menampilkan contoh kesalahan yang dicatat. Defaultnya adalah 10 entri per kueri.

SYS_EXTERNAL_QUERY_ERROR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang menghasilkan baris ini.
Query_id	bigint	Pengidentifikasi kueri dari kueri yang menghasilkan baris ini.
file_location	karakter (256)	Lokasi data yang ditanyakan.
jijik	char (2100)	Lokasi kesalahan dalam file. Bag `rowid` ian-bagian dipisahkan dengan `:` (titik dua) dan bagian tambahan mungkin ditambahkan di masa mendatang. `row_offset :row_group :row_id` Row_offset adalah offset (dalam byte) dari baris dalam file dan diatur `-1` untuk format file yang tidak didukung.

Nama kolom	Jenis data	Deskripsi
		Sebuah tabel dibagi menjadi row_groups, dan setiap grup memiliki baris dengan row_ids yang berbeda.
column_name	karakter (127)	Nama kolom yang dikembalikan oleh query.
nilai_asli	karakter (1024)	Nilai asli ditanyakan.
nilai_dimodifikasi	karakter (1024)	Nilai yang dimodifikasi dikembalikan berdasarkan opsi konfigurasi penanganan data yang ditentukan dalam kueri.
pemicu	karakter (128)	Opsi penanganan data ditentukan dalam kueri.
tindakan	karakter (128)	Tindakan terkait dengan opsi penanganan data yang ditentukan dalam kueri.
nilai_tindakan	karakter (128)	Nilai parameter tindakan yang terkait dengan opsi penanganan data yang ditentukan dalam kueri.
kode_kesalahan	integer	Kode hasil dari opsi penanganan data yang ditentukan dalam kueri.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Contoh kueri

Kueri berikut mengembalikan daftar baris yang operasi penanganan data dilakukan.

```
SELECT * FROM sys_external_query_error;
```

Kueri mengembalikan hasil yang mirip dengan berikut ini.

```
   user_id   query_id  file_location                                rowid
column_name           original_value            modified_value       trigger
action               action_value                error_code
```

```
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0
league_name           Barclays Premier League    Barclays Premier Lea UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0
league_nspi           34595                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:1
league_nspi           34151                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2
league_name           Barclays Premier League    Barclays Premier Lea UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2
league_nspi           33223                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3
league_name           Barclays Premier League    Barclays Premier Lea UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3
league_nspi           32808                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:4
league_nspi           32790                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:5
league_name           Spanish Primera Division   Spanish Primera Divi UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:6
league_name           Spanish Primera Division   Spanish Primera Divi UNSPECIFIED
TRUNCATE                                          156
```

SYS_EXTRA_COMPUTE_UNTUK_OPTIMASI_OTOMATIS

Gunakan SYS_EXTRA_COMPUTE_FOR_AUTOMATIC_OPTIMIZATION untuk melihat periode penggunaan di mana Amazon Redshift menjalankan tugas pengoptimalan otomatis menggunakan sumber daya komputasi tambahan. Untuk informasi selengkapnya tentang pengoptimalan otomatis, lihatOptimasi database otomatis. Untuk informasi selengkapnya tentang pengoptimalan otomatis yang dijalankan menggunakan sumber daya komputasi tambahan, lihat. Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis

SYS_EXTRA_COMPUTE_FOR_AUTOMATIC_OPTIMIZATION hanya tersedia untuk cluster yang disediakan.

SYS_EXTRA_COMPUTE_FOR_AUTOMATIC_OPTIMIZATION hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu mulai	stempel waktu tanpa zona waktu	Waktu periode optimasi komputasi ekstra dimulai.
akhir_waktu	stempel waktu tanpa zona waktu	Waktu periode optimasi komputasi ekstra berakhir.
jumlah kueri	bigint	Jumlah kueri pengoptimalan otomatis yang dijalankan selama periode penggunaan.
komput_detik	numerik (27,0)	Jumlah detik dalam periode penggunaan yang dihabiskan Amazon Redshift untuk menjalankan tugas pengoptimalan otomatis menggunakan sumber daya komputasi tambahan.

Contoh

Berikut ini adalah contoh kueri mencari pengoptimalan otomatis yang dilakukan pada 16 September 2025.

```
SELECT *
FROM sys_extra_compute_for_automatic_optimization
WHERE start_time BETWEEN '2025-09-16 00:00:00' AND '2025-09-16 23:59:59';

start_time           | end_time            | query_count | compute_seconds
---------------------+---------------------+-------------+-----------------
 2025-09-16 00:00:00  | 2025-09-16 00:00:59 | 1           | 59
 2025-09-16 00:01:05  | 2025-09-16 00:01:58 | 2           | 53
```

SYS_INTEGRASI_AKTIVITAS

SYS_INTEGRATION_ACTIVITY menampilkan detail tentang proses integrasi yang telah selesai.

SYS_INTEGRATION_ACTIVITY hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang integrasi Zero-ETL, lihat Bek erja dengan integrasi Zero-ETL di Panduan Manajemen Amazon Redshift.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
integration_id	karakter (128)	Identifier yang terkait dengan integrasi.
Target_database	karakter (128)	Basis data di Amazon Redshift yang menerima data integrasi.
sumber	karakter (128)	Sumber data untuk integrasi. Jenis yang mungkin termasuk `MySQL` dan`PostgreSQL` .
nama_pos pemeriksaan	karakter (128)	Nama pos pemeriksaan mereplikasi koordinat binlog.
tipe_pos pemeriksaan	karakter (16)	Jenis pos pemeriksaan. Nilai yang mungkin termasuk:`snapshot`,`cdc`.
checkpoint_byte	bigint	Jumlah byte di pos pemeriksaan ini.
stempel waktu terakhir_ komit	timestamp	Stempel waktu saat terakhir dikomit di pos pemeriksa an ini.
tabel_yang dimodifikasi	integer	Jumlah tabel yang dimodifikasi di pos pemeriksaan.
integrasi_mulai_waktu	Waktu	Waktu (UTC) ketika integrasi dimulai untuk pos pemeriksaan ini.

Nama kolom	Jenis data	Deskripsi
integrasi_akhir_waktu	Waktu	Waktu (UTC) ketika integrasi berakhir untuk pos pemeriksaan ini.

Kueri Sampel

Perintah SQL berikut menampilkan log integrasi.

```
select * from sys_integration_activity;

          integration_id              | target_database | source |
   checkpoint_name                  | checkpoint_type  | checkpoint_bytes |
 last_commit_timestamp   | modified_tables |   integration_start_time   |
 integration_end_time
-------------------------------------+-----------------+--------
+--------------------------------------------+------------------+------------------
+------------------------+-----------------+---------------------------
+---------------------------
 76b15917-afae-4447-b7fd-08e2a5acce7b |   demo1         | MySQL  | checkpoints/
checkpoint_3_241_3_510.json     |        cdc       |        762       | 2023-05-10
 23:00:14.201 |         1       | 2023-05-10 23:00:45.054265 | 2023-05-10
 23:00:46.339826
 76b15917-afae-4447-b7fd-08e2a5acce7b |   demo1         | MySQL  | checkpoints/
checkpoint_3_16329_3_17839.json |        cdc       |       13488      | 2023-05-11
 01:33:57.411 |         2       | 2023-05-11 02:19:09.440121 | 2023-05-11
 02:19:16.090492
 76b15917-afae-4447-b7fd-08e2a5acce7b |   demo1         | MySQL  | checkpoints/
checkpoint_3_5103_3_5532.json   |        cdc       |        1657      | 2023-05-10
 23:13:14.205 |         2       | 2023-05-10 23:13:23.545487 | 2023-05-10
 23:13:25.652144
```

SYS_INTEGRASI_TABEL_AKTIVITAS

SYS_INTEGRATION_TABLE_ACTIVITY menampilkan detail aktivitas menyisipkan, menghapus, dan memperbarui integrasi Zero-ETL. Ada satu baris ditambahkan untuk setiap konsumsi yang selesai.

Seorang superuser dapat melihat semua baris dalam tabel ini.

Untuk informasi selengkapnya, lihat Integr asi Zero-ETL.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
integration_id	karakter (128)	Identifier yang terkait dengan integrasi.
nama_pos pemeriksaan	karakter (128)	Nama pos pemeriksaan.
Target_database	karakter (128)	Nama database Amazon Redshift.
schema_name	karakter (128)	Nama skema Amazon Redshift.
table_name	karakter (128)	Nama tabel.
tabel_id	integer	Pengidentifikasi tabel.
waktu_rekor	timestamp	Waktu (UTC) saat perubahan ini selesai.
id transaksi_	bigint	Pengidentifikasi transaksi.
baris yang disisipkan	bigint	Jumlah baris yang dimasukkan oleh konsumsi.
baris yang dihapus	bigint	Jumlah baris yang dihapus oleh konsumsi.
baris yang diperbarui	bigint	Jumlah baris yang diperbarui oleh konsumsi.
bytes_tertelan	bigint	Jumlah byte yang tertelan.

Kueri Sampel

Perintah SQL berikut menampilkan aktivitas integrasi.

```
select * from sys_integration_table_activity;
```

```
          integration_id              | checkpoint_name | target_database | schema_name
 |     table_name    | table_id     | record_time                | transaction_id  |
 inserted_rows  | deleted_rows | updated_rows | bytes_ingested
-------------------------------------+-----------------+----------------
+-------------+------------------+-------------+---------------------------
+-----------------+---------------+-------------+-------------+--------------
 4798e675-8f9f-4686-b05f-92c538e19629 |                 | sample_test2    |    sample
 | SampleTestChannel |  111276      | 2023-05-12 12:40:30.656625 | 7736            |  2
             | 0            | 0            | 125
```

SYS_INTEGRASI_TABEL_STATE_PERUBAHAN

SYS_INTEGRATION_TABLE_STATE_CHANGE menampilkan detail tentang log perubahan status tabel untuk integrasi.

Seorang superuser dapat melihat semua baris dalam tabel ini.

Untuk informasi selengkapnya, lihat Bekerja dengan Zero-ETL integrasi.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
integration_id	karakter (128)	Identifier yang terkait dengan integrasi.
database_name	karakter (128)	Nama database Amazon Redshift.
schema_name	karakter (128)	Nama skema Amazon Redshift.
table_name	karakter (128)	Nama tabel.
negara baru_	karakter (128)	Keadaan meja. Peluang nilai adalah `Synced`, `ResyncRequired` , `ResyncInitiated` , `Deleted`, `Failed`, dan `ResyncDeleted` .
table_last_replika t_checkpoint	karakter (128)	Koordinat log yang disinkronkan saat ini.

Nama kolom	Jenis data	Deskripsi
negara_perubahan_alasan	karakter (256)	Alasan transisi negara terakhir.
waktu_rekor	timestamp	Waktu (UTC) saat catatan ini diperbarui.

Kueri Sampel

Perintah SQL berikut menampilkan log integrasi.

```
select * from sys_integration_table_state_change;

            integration_id            | database_name | schema_name | table_name
 | new_state |  table_last_replicated_checkpoint   | state_change_reason |
 record_time
--------------------------------------+---------------+-------------+------------
+-----------+--------------------------------------+---------------------
+----------------------------
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest79   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:39:50.087868
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest56   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:39:45.54005
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest50   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:40:20.362504
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest18   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:40:32.544084
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest40t3s | sbtest23   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 15:49:05.186209
```

SYS_LOAD_DETAIL

Mengembalikan informasi untuk melacak atau memecahkan masalah pemuatan data.

Tampilan ini mencatat kemajuan setiap file data saat dimuat ke dalam tabel database.

Tampilan ini terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang membuat entri.
Query_id	integer	ID kueri.
file_name	karakter (256)	Nama file yang akan dimuat.
bytes_dipindai	integer	Jumlah byte yang dipindai dari file di Amazon S3.
garis_dipindai	integer	Jumlah baris yang dipindai dari file muat. Nomor ini mungkin tidak cocok dengan jumlah baris yang sebenarnya dimuat. Misalnya, beban dapat memindai tetapi mentolerir sejumlah catatan buruk, berdasarkan opsi MAXERROR dalam perintah COPY.
waktu_rekor	timestamp	Waktu entri ini terakhir diperbarui.
splits_scan	Jumlah pemisahan file ini.	Jumlah pemisahan file ini.
waktu mulai	timestamp	Waktu pemrosesan file ini dimulai.
akhir_waktu	timestamp	Waktu pemrosesan file ini selesai.
tag file	karakter (256)	ETag file di Amazon S3.
file_terakhir_dimodifikasi	timestamp	Stempel waktu terakhir yang dimodifikasi dari file di Amazon S3.

Kueri Sampel

Contoh berikut mengembalikan detail untuk operasi COPY terakhir.

```
select query_id, trim(file_name) as file, record_time
from sys_load_detail
where query_id = pg_last_copy_id();

 query_id |               file               |          record_time
----------+----------------------------------+----------------------------
 28554    | s3://dw-tickit/category_pipe.txt | 2013-11-01 17:14:52.648486
(1 row)
```

Kueri berikut berisi entri untuk memuat tabel baru di database TICKIT:

```
select query_id, trim(file_name), record_time
from sys_load_detail
where file_name like '%tickit%' order by query_id;

 query_id |           btrim           |          record_time
----------+---------------------------+----------------------------
 22475    | tickit/allusers_pipe.txt | 2013-02-08 20:58:23.274186
 22478    | tickit/venue_pipe.txt    | 2013-02-08 20:58:25.070604
 22480    | tickit/category_pipe.txt | 2013-02-08 20:58:27.333472
 22482    | tickit/date2008_pipe.txt | 2013-02-08 20:58:28.608305
 22485    | tickit/allevents_pipe.txt| 2013-02-08 20:58:29.99489
 22487    | tickit/listings_pipe.txt | 2013-02-08 20:58:37.632939
 22593    | tickit/allusers_pipe.txt | 2013-02-08 21:04:08.400491
 22596    | tickit/venue_pipe.txt    | 2013-02-08 21:04:10.056055
 22598    | tickit/category_pipe.txt | 2013-02-08 21:04:11.465049
 22600    | tickit/date2008_pipe.txt | 2013-02-08 21:04:12.461502
 22603    | tickit/allevents_pipe.txt| 2013-02-08 21:04:14.785124
 22605    | tickit/listings_pipe.txt | 2013-02-08 21:04:20.170594

(12 rows)
```

Fakta bahwa catatan ditulis ke file log untuk tampilan sistem ini tidak berarti bahwa beban berhasil dilakukan sebagai bagian dari transaksi yang berisi. Untuk memverifikasi load commit, kueri tampilan STL_UTILITYTEXT dan cari catatan COMMIT yang sesuai dengan transaksi COPY. Misalnya, kueri ini bergabung dengan SYS_LOAD_DETAIL dan STL_QUERY berdasarkan subquery terhadap STL_UTILITYTEXT:

```
select l.query_id,rtrim(l.file_name),q.transaction_id
from sys_load_detail l, sys_query_text q
where l.query_id=q.query_id
and exists
(select xid from stl_utilitytext where xid=q.transaction_id and
 rtrim("text")='COMMIT');

 query_id |           rtrim           |  transaction_id
----------+---------------------------+-----------------
 22600    | tickit/date2008_pipe.txt  | 68311
 22480    | tickit/category_pipe.txt  | 68066
  7508    | allusers_pipe.txt         | 23365
  7552    | category_pipe.txt         | 23415
  7576    | allevents_pipe.txt        | 23429
  7516    | venue_pipe.txt            | 23390
  7604    | listings_pipe.txt         | 23445
 22596    | tickit/venue_pipe.txt     | 68309
 22605    | tickit/listings_pipe.txt  | 68316
 22593    | tickit/allusers_pipe.txt  | 68305
 22485    | tickit/allevents_pipe.txt | 68071
  7561    | allevents_pipe.txt        | 23429
  7541    | category_pipe.txt         | 23415
  7558    | date2008_pipe.txt         | 23428
 22478    | tickit/venue_pipe.txt     | 68065
   526    | date2008_pipe.txt         |  2572
  7466    | allusers_pipe.txt         | 23365
 22482    | tickit/date2008_pipe.txt  | 68067
 22598    | tickit/category_pipe.txt  | 68310
 22603    | tickit/allevents_pipe.txt | 68315
 22475    | tickit/allusers_pipe.txt  | 68061
   547    | date2008_pipe.txt         |  2572
 22487    | tickit/listings_pipe.txt  | 68072
  7531    | venue_pipe.txt            | 23390
  7583    | listings_pipe.txt         | 23445
(25 rows)
```

SYS_LUDF_DETAIL

SYS_LUDF_DETAIL merekam informasi dan metrik untuk Lambda User Defined Functions (LUDFs) yang digunakan dalam kueri tertentu.

SYS_LUDF_DETAIL hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri memanggil fungsi.
id transaksi _	bigint	Pengidentifikasi transaksi.
Query_id	bigint	Pengidentifikasi kueri pengguna.
fungsi_oid	bigint	ID objek dari fungsi dalam katalog.
fungsi_po sisi	integer	Posisi numerik fungsi ini dalam langkah. Misalnya, jika ada beberapa panggilan dari fungsi yang sama dalam daftar SELECT kueri, mereka dapat diidentifikasi menggunakan nomor posisi ini.
Stream_id	integer	Pengidentifikasi aliran tempat panggilan fungsi dilakukan.
segmen_id	integer	Pengidentifikasi segmen tempat panggilan fungsi dilakukan.
Step_id	integer	Pengidentifikasi langkah di mana panggilan fungsi dilakukan.
nama_fung si lambda	karakter (256)	Nama fungsi lambda.
waktu mulai	timestamp	Waktu panggilan dimulai pada.
akhir_waktu	timestamp	Waktu panggilan berakhir pada.
total_durasi	bigint	Total durasi panggilan dalam mikrodetik.
Invokasi	integer	Jumlah pemanggilan bersamaan atau eksternal.
total_baris	bigint	Jumlah baris passed/returned to/from panggilan.

Nama kolom	Jenis data	Deskripsi
input_byte	bigint	Jumlah byte yang diteruskan ke panggilan.
output_byte	bigint	Jumlah byte panggilan yang dihasilkan.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Contoh berikut menggunakan Lambda UDF dalam kueri, dan kemudian menunjukkan cara menanyakan tampilan SYS_LUDF_DETAIL untuk melihat detail eksekusi fungsi.

```
SET SESSION AUTHORIZATION regular_user;

CREATE EXTERNAL FUNCTION exfunc_sum(INT,INT) RETURNS INT STABLE
LAMBDA 'lambda_sum'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';

CREATE TABLE t_sum(c1 int, c2 int);
INSERT INTO t_sum VALUES (4,5), (6,7);
SELECT exfunc_sum(c1,c2) FROM t_sum;

-- Switch to super user in order to inspect records in the LUDF SYS view.
SET SESSION AUTHORIZATION super_user;
select * from sys_ludf_detail;
```

Contoh output:

```
 user_id | transaction_id | query_id | function_oid | function_position | stream_id
 | segment_id | step_id | lambda_function_name |         start_time         |
     end_time          | total_duration | invocations | total_rows | input_bytes |
 output_bytes
---------+----------------+----------+--------------+-------------------+-----------
+------------+---------+----------------------+---------------------------
+---------------------------+---------------+-------------+------------+-------------
+--------------
     100 |           1463 |     1544 |       111055 |                 0 |         0 |
         0 |       2 | lambda_sum           | 2026-01-06 17:23:25.165898 | 2026-01-06
```

```
 17:23:25.165898 |           414 |           1 |          2 |         277 |
 18
(1 row)
```

SYS_LOAD_ERROR_DETAIL

Gunakan SYS_LOAD_ERROR_DETAIL untuk melihat detail kesalahan perintah COPY. Setiap baris mewakili perintah COPY. Ini berisi perintah COPY yang sedang berjalan dan selesai.

SYS_LOAD_ERROR_DETAIL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan salinan.
Query_id	bigint	Pengidentifikasi kueri salinan.
id transaksi_	bigint	Pengidentifikasi transaksi.
session_id	integer	Pengidentifikasi proses dari proses yang menjalankan salinan.
database_name	karakter (64)	Nama database yang terhubung ke pengguna saat salinan dikeluarkan.
tabel_id	integer	Pengidentifikasi tabel.
waktu mulai	timestamp	Waktu (UTC) saat salinan dimulai.
file_name	karakter (256)	Jalur lengkap ke file input untuk dimuat.

Nama kolom	Jenis data	Deskripsi
bilang_baris	bigint	Nomor baris dalam file muat dengan kesalahan. Saat Anda memuat file JSON, nomor baris baris terakhir objek JSON dengan kesalahan.
column_name	karakter (127)	Bidang dengan kesalahan.
tipe_kolom	karakter (10)	Jenis data bidang dengan kesalahan.
panjang_kolom	karakter (10)	Panjang kolom, jika berlaku. Bidang ini diisi ketika tipe data memiliki batas panjang. Misalnya, untuk kolom dengan tipe data “karakter (3)”, kolom ini berisi nilai “3.”
posisi	integer	Posisi kesalahan di lapangan.
kode kesalahan	integer	Kode kesalahan.
pesan kesalahan	karakter (512)	Penjelasan kesalahan.

Kueri Sampel

Kueri berikut menunjukkan rincian kesalahan pemuatan perintah salin untuk kueri tertentu.

```
SELECT query_id,
       table_id,
       start_time,
       trim(file_name) AS file_name,
       trim(column_name) AS column_name,
       trim(column_type) AS column_type,
       trim(error_message) AS error_message
FROM sys_load_error_detail
WHERE query_id = 762949
ORDER BY start_time
```

```
LIMIT 10;
```

Sampel keluaran.

```
 query_id | table_id |         start_time         |               file_name
      | column_name | column_type |                 error_message
----------+----------+----------------------------
+-----------------------------------------+-------------+-------------
+-----------------------------------------------
   762949 |   137885 | 2022-02-15 22:14:46.759151 | s3://load-test/copyfail/
wrong_format_000 | id          | int4        | Invalid digit, Value 'a', Pos 0, Type:
 Integer
   762949 |   137885 | 2022-02-15 22:14:46.759151 | s3://load-test/copyfail/
wrong_format_001 | id          | int4        | Invalid digit, Value 'a', Pos 0, Type:
 Integer
```

SYS_LOAD_HISTORY

Gunakan SYS_LOAD_HISTORY untuk melihat detail perintah COPY. Setiap baris mewakili perintah COPY dengan statistik akumulasi untuk beberapa bidang. Ini berisi perintah COPY yang sedang berjalan dan selesai.

SYS_LOAD_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan salinan.
Query_id	bigint	Pengidentifikasi kueri salinan.
id transaksi_	bigint	Pengidentifikasi transaksi.
session_id	integer	Pengidentifikasi proses dari proses yang menjalankan salinan.

Nama kolom	Jenis data	Deskripsi
database_name	text	Nama database yang terhubung ke pengguna saat operasi dikeluarkan.
status	text	Status salinan. Nilai yang valid adalah`running`,`completed`,`aborted`.
table_name	text	Nama tabel menyalin ke.
waktu mulai	timestamp	Waktu ketika salinan dimulai.
akhir_waktu	timestamp	Waktu ketika salinan selesai.
durasi	bigint	Jumlah waktu (mikrodetik) yang dihabiskan dalam perintah COPY.
sumber data	text	Lokasi Amazon S3 dari masukan file untuk disalin.
file_format	text	Format file sumber. Format termasuk csv, txt, json, avro, orc, atau parket.
baris yang dimuat	bigint	Jumlah baris yang disalin ke tabel.
dimuat_byte	bigint	Jumlah byte yang disalin ke tabel.
sumber_file_count	integer	Jumlah file dihitung dalam file sumber.
source_file_byte	bigint	Jumlah byte dalam file sumber.

Nama kolom	Jenis data	Deskripsi
file_count_dipindai	integer	Jumlah file yang dipindai dari Amazon S3.
file_bytes_dipindai	bigint	Jumlah byte yang dipindai dari file di Amazon S3.
hitung_kesalahan	bigint	Jumlah kesalahan dihitung.
salin_job_id	bigint	Pengidentifikasi pekerjaan menyalin. A `0` menunjukkan tidak ada pengidentifikasi pekerjaan.

Kueri Sampel

Kueri berikut menunjukkan baris, byte, tabel, dan sumber data yang dimuat dari perintah salinan tertentu.

```
SELECT query_id,
       table_name,
       data_source,
       loaded_rows,
       loaded_bytes
FROM sys_load_history
WHERE query_id IN (6389,490791,441663,74374,72297)
ORDER BY query_id,
         data_source DESC;
```

Sampel keluaran.

```
 query_id |    table_name    |                                  data_source
              | loaded_rows | loaded_bytes
----------+------------------
+-------------------------------------------------------------------------+-------------
+---------------
     6389 | store_returns    | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
store_returns/    |   287999764 | 1196240296158
```

```
    72297 | web_site         | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
web_site/         |          54 |         43808
    74374 | ship_mode        | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
ship_mode/        |          20 |          1320
   441663 | income_band      | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
income_band/      |          20 |          2152
   490791 | customer_address | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
customer_address/ |     6000000 |     722924305
```

Kueri berikut menunjukkan baris, byte, tabel, dan sumber data yang dimuat dari perintah salin.

```
SELECT query_id,
       table_name,
       data_source,
       loaded_rows,
       loaded_bytes
FROM sys_load_history
ORDER BY query_id DESC
LIMIT 10;
```

Sampel keluaran.

```
 query_id |       table_name       |                                data_source
                          | loaded_rows |  loaded_bytes
----------+------------------------
+-------------------------------------------------------------------------------
+-------------+-----------------
   491058 | web_site               | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_site/               |          54 |           43808
   490947 | web_sales              | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_sales/              |   720000376 |  22971988122819
   490923 | web_returns            | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_returns/            |    71997522 |     96597496325
   490918 | web_page               | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_page/               |        3000 |            1320
   490907 | warehouse              | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/warehouse/              |          20 |            1320
   490902 | time_dim               | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/time_dim/               |       86400 |            1320
   490876 | store_sales            | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/store_sales/            |  2879987999 | 151666241887933
   490870 | store_returns          | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/store_returns/          |   287999764 |   1196405607941
```

```
   490865 | store                  | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/store/                 |        1002 |          365507
```

Kueri berikut menunjukkan baris dan byte harian yang dimuat dari perintah salin.

```
SELECT date_trunc('day',start_time) AS exec_day,
       SUM(loaded_rows) AS loaded_rows,
       SUM(loaded_bytes) AS loaded_bytes
FROM sys_load_history
GROUP BY exec_day
ORDER BY exec_day DESC;
```

Sampel keluaran.

```
      exec_day       | loaded_rows |   loaded_bytes
---------------------+-------------+------------------
 2022-01-20 00:00:00 |  6347386005 |  258329473070606
 2022-01-19 00:00:00 | 19042158015 |  775198502204572
 2022-01-18 00:00:00 | 38084316030 | 1550294469446883
 2022-01-17 00:00:00 | 25389544020 | 1033271084791724
 2022-01-16 00:00:00 | 19042158015 |  775222736252792
 2022-01-15 00:00:00 | 19834245387 |  798122849155598
 2022-01-14 00:00:00 | 75376544688 | 3077040926571384
```

SYS_MV_REFRESH RIWAYAT

Hasilnya mencakup informasi tentang riwayat penyegaran semua tampilan yang terwujud. Hasilnya termasuk jenis refresh, seperti manual atau otomatis, dan status penyegaran terbaru.

SYS_MV_REFRESH_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan penyegaran.

Nama kolom	Jenis data	Deskripsi
session_id	integer	Pengidentifikasi proses untuk proses yang menjalankan penyegaran tampilan terwujud.
id transaksi_	bigint	Pengidentifikasi transaksi.
database_name	karakter (128)	Basis data yang berisi tampilan terwujud.
schema_name	karakter (128)	Skema tampilan yang terwujud.
mv_id	bigint	ID objek dari tampilan terwujud.
mv_name	karakter (128)	Nama tampilan yang terwujud.
jenis penyegaran	karakter (32)	Jenis penyegaran, seperti manual atau otomatis.
status	text	Status penyegaran. Untuk informasi terperinci tentang status, lihat kolom status untukSVL_MV_REFRESH_STATUS.
waktu mulai	timestamp	Waktu mulai penyegaran.
akhir_waktu	timestamp	Waktu akhir penyegaran.
durasi	bigint	Jumlah waktu dalam mikrodetik yang dibutuhkan untuk menyegarkan tampilan yang terwujud.

Nama kolom	Jenis data	Deskripsi
akun_konsumen	arang (12)	ID AWS akun dari cluster konsumen yang memulai penyegaran. Bidang ini diisi saat refresh dimulai dari cluster konsumen dalam pengaturan berbagi data.
wilayah konsumen	karakter (32)	Wil AWS ayah cluster konsumen yang memulai penyegaran. Bidang ini diisi saat refresh dimulai dari cluster konsumen dalam pengaturan berbagi data.
ruang nama konsumen	karakter (36)	Pengidentifikasi namespace dari cluster konsumen yang memulai penyegaran. Bidang ini diisi saat refresh dimulai dari cluster konsumen dalam pengaturan berbagi data.

Kueri Sampel

Kueri berikut menunjukkan riwayat penyegaran untuk tampilan yang terwujud.

```
SELECT user_id,
     session_id,
     transaction_id,
     database_name,
     schema_name,
     mv_id,
     mv_name,
     refresh_type,
     status,
     start_time,
     end_time,
     duration,
```

```
    consumer_account,
    consumer_region,
    consumer_namespace
    from sys_mv_refresh_history;
```

Kueri mengembalikan output sampel berikut:

```
 user_id | session_id | transaction_id | database_name | schema_name                |
 mv_id  |  mv_name           |  refresh_type  |  status
                                                                  |  start_time
      |  end_time                  |  duration | consumer_account | consumer_region |
 consumer_namespace
---------+------------+----------------+--------------
+----------------------------+--------+--------------------+----------------
+-------------------------------------------------------------------------------------------------
+----------------------------+---------------------------+-----------
+------------------+-----------------+-----------------------------------
       1 | 1073815659 |          15066 | dev           | test_stl_mv_refresh_schema
 | 203762 | mv_incremental     | Manual         | MV was already updated
                                                                  | 2023-10-26
 15:59:20.952179 | 2023-10-26 15:59:20.952866 |      687 |                  |
       |
       1 | 1073815659 |          15068 | dev           | test_stl_mv_refresh_schema
 | 203771 | mv_nonincremental  | Manual         | MV was already updated
                                                                  | 2023-10-26
 15:59:21.008049 | 2023-10-26 15:59:21.008658 |      609 |                  |
       |
       1 | 1073815659 |          15070 | ext_db        | producer_schema
  | 203779 | producer_mv        | Manual         | Refresh successfully updated MV
 incrementally                                                    | 2023-10-26
 15:59:21.064252 | 2023-10-26 15:59:21.064885 |      633 | 0123456789       | us-east-1
       | 623d8ff2-4391-4381-83d7-177caa6767af
       1 | 1073815659 |          15074 | dev           | test_stl_mv_refresh_schema
 | 203762 | mv_incremental     | Manual         | Refresh successfully updated MV
 incrementally                                                    | 2023-10-26
 15:59:29.693329 | 2023-10-26 15:59:43.482842 | 13789513 |                  |
       |
       1 | 1073815659 |          15076 | dev           | test_stl_mv_refresh_schema
 | 203771 | mv_nonincremental  | Manual         | Refresh successfully recomputed
 MV from scratch                                                  | 2023-10-26
 15:59:43.550184 | 2023-10-26 15:59:47.880833 |  4330649 |                  |
       |
```

```
      1 | 1073815659 |          15078 | dev           | test_stl_mv_refresh_schema
 | 203779 | mv_refresh_error   | Manual         | Refresh failed due to an internal
 error                                                              | 2023-10-26
 15:59:47.949052 | 2023-10-26 15:59:52.494681 |  4545629 |                  |
      |
      1 | 1073815659 |          15071 | dev           | test_stl_mv_refresh_schema
 | 203778 | mv_test            | Manual         | Cascade refresh failed because
 materialized view test_stl_mv_refresh_schema.child was not refreshed. | 2023-10-26
 15:30:21.432252 | 2023-10-26 15:30:21.432252 |      532 |                  |
      |
      1 | 1073815659 |          15071 | dev           | test_stl_mv_refresh_schema
 | 203761 | child              | Manual         | Refresh failed due to an internal
 error.                                                             | 2023-10-26
 15:30:21.432252 | 2023-10-26 15:30:21.432252 |      532 |                  |
      |
      1 | 1073815659 |          15069 | dev           | test_stl_mv_refresh_schema
 | 203778 | mv_test            | Manual         | Cascade refresh skipped because
 materialized view test_stl_mv_refresh_schema.child was not refreshed.| 2023-10-26
 15:21:43.550369 | 2023-10-26 15:21:43.550369 |      633
      1 | 1073815659 |          15069 | dev           | test_stl_mv_refresh_schema
 | 203761 | child              | Manual         | Refresh failed due to an internal
 error.                                                             | 2023-10-26
 15:21:43.550369 | 2023-10-26 15:21:43.550369 |      633
(10 rows)
```

SYS_MV_STATUS

Hasilnya mencakup informasi tentang keadaan semua pandangan yang terwujud. Ini mencakup informasi tabel dasar, properti skema, dan informasi tentang peristiwa terbaru, seperti menjatuhkan kolom.

SYS_MV_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	bigint	ID pengguna yang membuat acara.

Nama kolom	Jenis data	Deskripsi
id transaksi_	bigint	ID transaksi acara.
database_name	karakter (128)	Basis data yang berisi tampilan terwujud.
event_desc	karakter (500)	Peristiwa yang mendorong perubahan negara. Contoh nilai meliputi yang berikut: • Jenis kolom diubah • Kolom dijatuhkan • Kolom diganti namanya • Nama skema diubah • Small-table konversi • MEMOTONG • Vakum Perhatikan bahwa ada kemungkinan nilai lain untuk kolom ini.
waktu mulai	timestamp	Waktu mulai acara.
nama_database_tabel_dasar	karakter (128)	Nama database untuk tabel dasar.
skema tabel_dasar	karakter (128)	Skema tabel dasar.
nama_tabel_dasar	karakter (128)	Nama tabel dasar.
mv_schema	karakter (128)	Skema tampilan yang terwujud.
mv_name	karakter (128)	Nama tampilan yang terwujud.

Nama kolom	Jenis data	Deskripsi
status	karakter (32)	Keadaan yang berubah dari pandangan yang terwujud, yaitu sebagai berikut: • Hitung ulang • Tidak bisa disegarkan

Kueri Sampel

Kueri berikut menunjukkan status tampilan yang terwujud.

```
select * from sys_mv_state;
```

Kueri mengembalikan output sampel berikut:

```
 user_id | transaction_id | database_name | event_desc                  | start_time
               | base_table_database_name | base_table_schema | base_table_name     |
 mv_schema  | mv_name                    | state
---------+----------------+---------------+-----------------------------
+--------------------------+--------------------------+------------------
+---------------------+-------------+---------------------------+--------------
 106     | 12720          | tickit_db     | TRUNCATE                    | 2023-07-26
 14:59:12.788268 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_a1f3f862 | Recompute
 106     | 12724          | tickit_db     | ALTER TABLE ALTER DISTSTYLE | 2023-07-26
 14:59:51.409014 | tickit_db                | mv_schema         | test_table_58102435 |
 mv_schema   | materialized_view_ca746631 | Recompute
 106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
 14:59:12.822928 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Unrefreshable
 106     | 12727          | tickit_db     | Table was renamed           | 2023-07-26
 15:00:08.051244 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Unrefreshable
 106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
 14:59:12.857755 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Unrefreshable
 106     | 12727          | tickit_db     | Table was renamed           | 2023-07-26
 15:00:08.051358 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5ef0d754 | Unrefreshable
```

```
106     | 12720          | tickit_db     | TRUNCATE                    | 2023-07-26
14:59:12.788159 | tickit_db                 | mv_schema         | test_table_95d6d861 |
mv_schema   | materialized_view_5750a8d4 | Recompute
106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
14:59:12.857799 | tickit_db                 | mv_schema         | test_table_95d6d861 |
mv_schema   | materialized_view_a1f3f862 | Unrefreshable
106     | 12720          | tickit_db     | TRUNCATE                    | 2023-07-26
14:59:12.788327 | tickit_db                 | mv_schema         | test_table_95d6d861 |
mv_schema   | materialized_view_5ef0d754 | Recompute
106     | 12727          | tickit_db     | ALTER TABLE ALTER SORTKEY   | 2023-07-26
15:00:08.006235 | tickit_db                 | mv_schema         | test_table_58102435 |
mv_schema   | materialized_view_ca746631 | Recompute
106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
14:59:12.82297  | tickit_db                 | mv_schema         | test_table_95d6d861 |
mv_schema   | materialized_view_a1f3f862 | Unrefreshable
106     | 12727          | tickit_db     | Table was renamed           | 2023-07-26
15:00:08.051321 | tickit_db                 | mv_schema         | test_table_95d6d861 |
mv_schema   | materialized_view_a1f3f862 | Unrefreshable
```

SYS_PROSEDUR_PANGGILAN

Gunakan tampilan SYS_PROCEDURE_CALL untuk mendapatkan informasi tentang panggilan prosedur yang disimpan, termasuk waktu mulai, waktu akhir, status panggilan prosedur tersimpan, dan hierarki panggilan untuk panggilan prosedur tersimpan bersarang. Setiap panggilan prosedur tersimpan menerima ID kueri.

SYS_PROCEDURE_CALL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
session_user_id	integer	Pengidentifikasi pengguna yang membuat sesi dan merupakan pemanggil panggilan prosedur tersimpan tingkat atas.

Nama kolom	Jenis data	Deskripsi
security_user_id	integer	Pengidentifikasi pengguna yang hak-haknya digunakan untuk menjalankan pernyataan dalam prosedur tersimpan. Jika prosedur tersimpan dipanggil dari prosedur tersimpan DEFINER luar, maka ini akan menjadi user_id pemilik dari prosedur tersimpan luar itu.
Query_id	integer	Pengidentifikasi kueri dari panggilan prosedur tersimpan.
kueri_teks	char (4000)	Teks kueri panggilan prosedur tersimpan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri mulai berjalan. Stempel waktu menggunakan enam digit presisi untuk detik pecahan, misalnya. 2009-06-12 11:29:19.131 358.
akhir_waktu	timestamp	Waktu dalam UTC saat kueri selesai berjalan. Stempel waktu menggunakan enam digit presisi untuk detik pecahan, misalnya: 2009-06-12 11:29:19.131 358.

Nama kolom	Jenis data	Deskripsi
status	haru (10)	Status panggilan prosedur tersimpan. Ketika prosedur yang disimpan dihentikan oleh sistem atau dibatalkan oleh pengguna, nilainya dibatalka n. Jika panggilan prosedur tersimpan berjalan hingga selesai, nilainya berhasil.
id kueri prosedur pemanggil	integer	Jika panggilan prosedur tersimpan dipanggil oleh panggilan prosedur tersimpan lainnya, maka kolom ini berisi ID kueri dari panggilan luar. Jika tidak, bidang tersebut adalah NULL.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Kueri berikut mengembalikan hierarki panggilan prosedur tersimpan bersarang.

```
select query_id, datediff(seconds, start_time, end_time) as elapsed_time, status,
 trim(query_text) as call, caller_procedure_query_id from sys_procedure_call;
```

Sampel keluaran.

```
 query_id | elapsed_time | status  |                      call                      |
 caller_procedure_query_id
----------+--------------+---------+------------------------------------------------
+---------------------------
     3087 |           18 | success | CALL proc_bd906c98c45443ffa165e9552056902d(1)    |
          3085
     3085 |           18 | success | CALL proc_bd906c98c45443ffa165e9552056902d_2(1); |
```

```
(2 rows)
```

SYS_PROSEDUR_PESAN-PESAN

SYS_PROCEDURE_MESSAGES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
id transaksi_	bigint	Pengidentifikasi transaksi.
Query_id	integer	Pengidentifikasi kueri dari panggilan prosedur tersimpan.
waktu_rekor	timestamp	Waktu dalam UTC saat pesan dibuat.
logar_level	haru (10)	Tingkat log dari pesan yang dihasilkan. Nilai yang mungkin adalah LOG, INFO, PEMBERITA HUAN, PERINGATAN, dan PENGECUALIAN.
pesan	karakter (1024)	Teks pesan yang dihasilkan.
bilang_baris	integer	Nomor baris pesan yang dihasilkan.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Kueri berikut menunjukkan contoh output dari SYS_PROCEDURE_MESSAGES.

```
select transaction_id, query_id, record_time, log_level, trim(message), line_number
 from sys_procedure_messages;
```

```
transaction_id | query_id |        record_time         | log_level |           btrim
        | line_number
---------------+----------+----------------------------+-----------
+--------------------------+-------------
     25267     |   80562  | 2023-07-17 14:38:31.910136 |   NOTICE  |
 test_notice_msg_b9f1e749  |     8
     25267     |   80562  | 2023-07-17 14:38:31.910002 |    LOG    |
 test_log_msg_833c7420     |     6
     25267     |   80562  | 2023-07-17 14:38:31.910111 |    INFO   |
 test_info_msg_651373d9    |     7
     25267     |   80562  | 2023-07-17 14:38:31.910154 |   WARNING |
 test_warning_msg_831c5747 |     9
(4 rows)
```

SYS_QUERY_DETAIL

Gunakan SYS_QUERY_DETAIL untuk melihat detail untuk kueri di berbagai tingkat metrik, dengan setiap baris mewakili detail tentang kueri WLM tertentu pada tingkat metrik tertentu. Tampilan ini berisi banyak jenis kueri seperti perintah DDL, DML, dan utilitas (misalnya, menyalin dan membongkar). Beberapa kolom mungkin tidak relevan tergantung pada jenis kueri. Misalnya, external_scanned_bytes tidak relevan dengan tabel internal.

SYS_QUERY_DETAIL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> Untuk memverifikasi apakah transaksi yang berisi kueri yang dieksekusi berhasil dilakukan, Anda perlu melakukan operasi gabungan antara tabel sistem dan `sys_transaction_history` tabel. Contoh:
>
> ```
> SELECT
>     th.transaction_id,
>     qd.query_id,
>     th.status AS transaction_status
> FROM
> ```

```
    sys_query_detail qd
LEFT JOIN sys_query_history qh ON qd.query_id = qh.query_id
LEFT JOIN sys_transaction_history th on qh.transaction_id = th.transaction_id;
```

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
Query_id	bigint	Pengidentifikasi kueri.
urutan kueri anak	integer	Urutan permintaan pengguna yang ditulis ulang, dimulai dengan 1.
Stream_id	integer	Pengidentifikasi aliran dari aliran kueri.
segmen_id	integer	Pengidentifikasi segmen dari segmen yang berjalan kueri.
Step_id	integer	Pengidentifikasi langkah dalam segmen.
nama_langkah	text	Nama langkah dalam segmen. Nilai yang mungkin adalah `aggregate broadcast ,delete,distribut e ,hash,hashjoin,insert,limit,,m ,unique`, dan`window`.
tabel_id	integer	Pengidentifikasi tabel untuk pemindaian tabel permanen.
table_name	karakter (136)	Nama tabel langkah yang sedang dioperasikan.

Nama kolom	Jenis data	Deskripsi
is_rrscan	karakter	Nilai yang menunjukkan apakah suatu langkah adalah langkah pemindaian. True (t) menunjukkan bahwa pemindaian terbatas rentang digunakan.
waktu mulai	timestamp	Waktu ketika langkah query dimulai. Bidang ini dicatat pada tingkat segmen, terlepas dari nilai di `metrics_level` kolom.
akhir_waktu	timestamp	Waktu ketika langkah query selesai. Bidang ini dicatat pada tingkat segmen, terlepas dari nilai di `metrics_level` kolom.
durasi	bigint	Jumlah waktu (mikrodetik) yang dihabiskan untuk langkah. Bidang ini dicatat pada tingkat segmen, terlepas dari nilai di `metrics_level` kolom.
pemberitahuan	text	Deskripsi peristiwa peringatan.
input_byte	bigint	Byte input untuk langkah saat ini.
masukan_baris	bigint	Baris input untuk langkah saat ini.
output_byte	bigint	Byte keluaran untuk langkah saat ini.

Nama kolom	Jenis data	Deskripsi
keluaran_baris	bigint	Baris keluaran untuk langkah saat ini.
blokir_baca	bigint	Jumlah blok yang dibaca langkah.
blok_tulis	bigint	Jumlah blok yang ditulis langkah itu.
Lokal_baca_io	bigint	Jumlah blok yang dibaca dari cache disk lokal.
Jarak jauh_baca_io	bigint	Jumlah blok yang dibaca dari remote.
sumber	text	Jenis objek database yang dipindai. Kolom ini hanya memiliki nilai jika nilai step_name baris adalah. `scan`
kemiringan data	integer	Kemiringan distribusi baris keluaran di antara semua langkah. Ini adalah angka dalam kisaran 0% hingga 100%. Semakin besar jumlahnya, semakin tidak seimbang distribusinya.
kemiringan waktu	integer	Kemiringan distribusi waktu eksekusi di antara semua langkah. Ini adalah angka dalam kisaran 0% hingga 100%. Semakin besar jumlahnya, semakin tidak seimbang distribusinya.

Nama kolom	Jenis data	Deskripsi
adalah_aktif	karakter	Keadaan kueri pada tingkat langkah. Nilai yang mungkin adalah 't' yang menunjukk an langkah sedang berjalan secara aktif atau 'f' yang menunjukkan langkah selesai berjalan.
dipilled_block_local_disk	bigint	Jumlah blok tumpah ke disk lokal.
dipilled_block_remote_disk	bigint	Jumlah blok yang tumpah ke Amazon Simple Storage Service.
atribut langkah_langkah	karakter (64)	Berisi informasi tentang langkah terkait. Nilai yang mungkin untuk langkah pemindaian:`multi-dim ensional` .
metrik_level	karakter (64)	Tingkat metrik dari query. Kemungkinan nilainya adalah sebagai berikut: • Kueri anak • Streaming • Segment • Langkah

Nama kolom	Jenis data	Deskripsi
id_orangtua rencananya	integer	Pengidentifikasi node induk node rencana. Node induk dapat memiliki beberapa node anak. Misalnya, gabungan gabungan adalah simpul induk dari pemindaian pada tabel yang digabungkan.
plan_node_id	integer	Pengidentifikasi simpul rencana yang dipetakan ke satu atau lebih langkah dalam kueri.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Catatan penggunaan

SYS_QUERY_DETAIL dapat berisi metrik pada tingkat kueri langkah, steam, segmen, dan anak. Selain mereferensikan kolom metrics_level, Anda dapat melihat level metrik mana yang ditampilkan baris tertentu dengan mereferensikan bidang step_id, segment_id, dan stream_id sesuai dengan tabel berikut.

Tingkat metrik	nilai stream_id	nilai segment_id	nilai step_id
kueri anak	-1	-1	-1
aliran	Nilai langkah yang valid	-1	-1
segmen	Nilai langkah yang valid	Nilai langkah yang valid	-1
langkah	Nilai langkah yang valid	Nilai langkah yang valid	Nilai langkah yang valid

Kueri Sampel

Contoh berikut mengembalikan output dari SYS_QUERY_DETAIL.

Kueri berikut menunjukkan detail metadata kueri pada tingkat langkah, termasuk nama langkah, input_bytes, output_bytes, input_rows, output_rows.

```
SELECT query_id,
       child_query_sequence,
       stream_id,
       segment_id,
       step_id,
       trim(step_name) AS step_name,
       duration,
       input_bytes,
       output_bytes,
       input_rows,
       output_rows
FROM sys_query_detail
WHERE query_id IN (193929)
ORDER BY query_id,
         stream_id,
         segment_id,
         step_id DESC;
```

Sampel keluaran.

```
 query_id | child_query_sequence | stream_id | segment_id | step_id | step_name  |
 duration     | input_bytes | output_bytes | input_rows | output_rows
----------+----------------------+-----------+------------+---------+------------
+-----------------+-------------+--------------+------------+-------------
   193929 |                    2 |         0 |          0 |       3 | hash       |
      37144 |           0 |      9350272 |          0 |      292196
   193929 |                    5 |         0 |          0 |       3 | hash       |
       9492 |           0 |        23360 |          0 |        1460
   193929 |                    1 |         0 |          0 |       3 | hash       |
      46809 |           0 |      9350272 |          0 |      292196
   193929 |                    4 |         0 |          0 |       2 | return     |
       7685 |           0 |          896 |          0 |         112
   193929 |                    1 |         0 |          0 |       2 | project    |
      46809 |           0 |            0 |          0 |      292196
   193929 |                    2 |         0 |          0 |       2 | project    |
      37144 |           0 |            0 |          0 |      292196
```

```
193929 |                   5 |         0 |          0 |       2 | project    |
    9492 |           0 |           0 |          0 |        1460
193929 |                   3 |         0 |          0 |       2 | return     |
   11033 |           0 |       14336 |          0 |         112
193929 |                   2 |         0 |          0 |       1 | project    |
   37144 |           0 |           0 |          0 |      292196
193929 |                   1 |         0 |          0 |       1 | project    |
   46809 |           0 |           0 |          0 |      292196
193929 |                   5 |         0 |          0 |       1 | project    |
    9492 |           0 |           0 |          0 |        1460
193929 |                   3 |         0 |          0 |       1 | aggregate  |
   11033 |           0 |      201488 |          0 |          14
193929 |                   4 |         0 |          0 |       1 | aggregate  |
    7685 |           0 |       28784 |          0 |          14
193929 |                   5 |         0 |          0 |       0 | scan       |
    9492 |           0 |       23360 |     292196 |        1460
193929 |                   4 |         0 |          0 |       0 | scan       |
    7685 |           0 |        1344 |        112 |         112
193929 |                   2 |         0 |          0 |       0 | scan       |
   37144 |           0 |     7304900 |     292196 |      292196
193929 |                   3 |         0 |          0 |       0 | scan       |
   11033 |           0 |       13440 |        112 |         112
193929 |                   1 |         0 |          0 |       0 | scan       |
   46809 |           0 |     7304900 |     292196 |      292196
193929 |                   5 |         0 |          0 |      -1 |            |
    9492 |       12288 |           0 |          0 |         0
193929 |                   1 |         0 |          0 |      -1 |            |
   46809 |       16384 |           0 |          0 |         0
193929 |                   2 |         0 |          0 |      -1 |            |
   37144 |       16384 |           0 |          0 |         0
193929 |                   4 |         0 |          0 |      -1 |            |
    7685 |       28672 |           0 |          0 |         0
193929 |                   3 |         0 |          0 |      -1 |            |
   11033 |      114688 |           0 |          0 |         0
```

Untuk melihat tabel dalam database Anda dalam urutan dari yang paling sering digunakan hingga yang paling jarang digunakan, gunakan contoh berikut. Ganti *sample_data_dev* dengan database Anda sendiri. Perhatikan bahwa kueri ini akan menghitung kueri mulai saat cluster Anda dibuat, tetapi data tampilan sistem Anda tidak disimpan ketika gudang data Anda kekurangan ruang.

```
SELECT table_name, COUNT (DISTINCT query_id)
FROM SYS_QUERY_DETAIL
WHERE table_name LIKE 'sample_data_dev%'
```

```
GROUP BY table_name
ORDER BY COUNT(*) DESC;

+--------------------------------+-------+
|           table_name           | count |
+--------------------------------+-------+
| sample_data_dev.tickit.venue   |     4 |
| sample_data_dev.myunload1.venue |     3 |
| sample_data_dev.tickit.listing  |     1 |
| sample_data_dev.tickit.category |     1 |
| sample_data_dev.tickit.users    |     1 |
| sample_data_dev.tickit.date     |     1 |
| sample_data_dev.tickit.sales    |     1 |
| sample_data_dev.tickit.event    |     1 |
+--------------------------------+-------+
```

Contoh berikut menunjukkan berbagai tingkat metrik untuk kueri WLM tunggal.

```
SELECT query_id, child_query_sequence, stream_id, segment_id, step_id, step_name,
 start_time, end_time, metrics_level
FROM sys_query_detail
WHERE query_id = 1553 AND step_id = -1
ORDER BY stream_id, segment_id, step_id;

 query_id | child_query_sequence | stream_id | segment_id | step_id | step_name |
   start_time         |          end_time          | metrics_level
----------+----------------------+-----------+------------+---------+-----------
+----------------------------+----------------------------+---------------
     1553 |                    1 |        -1 |         -1 |      -1 |           |
 2024-10-17 02:28:49.814721 | 2024-10-17 02:28:49.847838 | child query
     1553 |                    1 |         0 |         -1 |      -1 |           |
 2024-10-17 02:28:49.814721 | 2024-10-17 02:28:49.835609 | stream
     1553 |                    1 |         0 |          0 |      -1 |           |
 2024-10-17 02:28:49.824677 | 2024-10-17 02:28:49.830372 | segment
     1553 |                    1 |         1 |         -1 |      -1 |           |
 2024-10-17 02:28:49.835624 | 2024-10-17 02:28:49.845773 | stream
     1553 |                    1 |         1 |          1 |      -1 |           |
 2024-10-17 02:28:49.84088  | 2024-10-17 02:28:49.842388 | segment
     1553 |                    1 |         1 |          2 |      -1 |           |
 2024-10-17 02:28:49.835926 | 2024-10-17 02:28:49.844396 | segment
     1553 |                    1 |         2 |         -1 |      -1 |           |
 2024-10-17 02:28:49.846949 | 2024-10-17 02:28:49.847838 | stream
```

```
     1553 |                    1 |         2 |          3 |      -1 |           |
 2024-10-17 02:28:49.847013 | 2024-10-17 02:28:49.847485 | segment
(8 rows)
```

SYS_QUERY_JELASKAN

Menampilkan rencana EXPLAIN untuk kueri yang telah dikirimkan untuk dieksekusi.

SYS_QUERY_EXPLAIN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
Query_id	bigint	Pengidentifikasi kueri. Informasi kueri terperinci disimpan diSYS_QUERY_HISTORY.
urutan kueri anak	integer	Urutan permintaan pengguna yang ditulis ulang, dimulai dengan 1.
plan_node_id	integer	Pengidentifikasi simpul rencana yang dipetakan ke satu atau lebih langkah dalam kueri.
id_orangtua rencananya	integer	Pengidentifikasi node induk node rencana. Node induk dapat memiliki beberapa node anak. Misalnya, gabungan gabungan adalah simpul induk

Nama kolom	Jenis data	Deskripsi
		dari pemindaian pada tabel yang digabungkan.
plan_node	karakter (400)	Teks simpul dari output EXPLAIN. Node rencana yang mengacu pada eksekusi pada node komputasi diawali dengan XN dalam output EXPLAIN.
rencana-info	karakter (400)	Informasi kualifikasi dan filter untuk simpul rencana. Misalnya, kondisi bergabung dan batasan klausa WHERE disertakan dalam kolom ini.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Contoh berikut adalah rencana EXPLAIN dari satu kueri.

```
SELECT * FROM sys_query_explain WHERE query_id = 612635 ORDER BY plan_node_id;

 userid | query_id | child_query_sequence | plan_node_id | plan_parent_id |


          plan_node

                                          |

                                                                plan_info
```

```
--------+----------+----------------------+-------------+---------------
+-----------------------------------------------------------------------------------------------
+-----------------------------------------------------------------------------------------------
    100 |   612635 |                    1 |           1 |             0 | XN Limit
 (cost=3604047533041.00..3604047533041.25 rows=100 width=20)

                                          |

    100 |   612635 |                    1 |           2 |             1 |   ->  XN
 Merge  (cost=3604047533041.00..3604047533148.02 rows=42809 width=20)

                                           | Merge Key: sum(b.totalprice)

    100 |   612635 |                    1 |           3 |             2 |         ->
 XN Network  (cost=3604047533041.00..3604047533148.02 rows=42809 width=20)

                                         |   Send to leader

    100 |   612635 |                    1 |           4 |             3 |
  ->  XN Sort  (cost=3604047533041.00..3604047533148.02 rows=42809 width=20)

                                         | Sort Key: sum(b.totalprice)
```

```
   100 |   612635 |                    1 |           5 |              4 |
          ->  XN HashAggregate  (cost=2604047529640.76..2604047529747.78 rows=42809
width=20)

                                          |

   100 |   612635 |                    1 |           6 |              5 |
                ->  XN Hash Join DS_DIST_NONE  (cost=15104956.16..2602364653507.34
rows=336575226684 width=20)

                                                    | Hash Cond: (("outer".listid =
"inner".listid) AND ("outer".sellerid = "inner".sellerid))

   100 |   612635 |                    1 |           7 |              6 |
                           ->  XN Seq Scan on listing b  (cost=0.00..7884677.12
rows=788467712 width=24)

                                                     |

   100 |   612635 |                    1 |           8 |              6 |
                  ->  XN Hash  (cost=7063797.76..7063797.76 rows=706379776 width=8)

                                        |
```

```
   100 |   612635 |                    1 |            9 |              8 |
                               ->  XN Seq Scan on sales a  (cost=0.00..7063797.76
rows=706379776 width=8)

                                                        |




(9 rows)
```

SYS_QUERY_HISTORY

Gunakan SYS_QUERY_HISTORY untuk melihat detail permintaan pengguna. Setiap baris mewakili permintaan pengguna dengan statistik terakumulasi untuk beberapa bidang. Tampilan ini berisi banyak jenis kueri, seperti bahasa definisi data (DDL), bahasa manipulasi data (DML), copy, unload, dan Amazon Redshift Spectrum. Ini berisi kueri yang sedang berjalan dan selesai.

SYS_QUERY_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> Untuk memverifikasi apakah transaksi yang berisi kueri yang dieksekusi berhasil dilakukan, Anda perlu melakukan operasi gabungan antara tabel sistem dan `sys_transaction_history` tabel. Contoh:
>
> ```
> SELECT
>     qh.transaction_id,
>     qh.query_id,
>     qh.status AS query_status,
>     qh.query_type,
>     TRIM(qh.query_text) AS query_text,
>     th.status AS transaction_status
> FROM
>     sys_query_history qh
> ```

```
LEFT JOIN
    sys_transaction_history th ON qh.transaction_id = th.transaction_id;
```

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
Query_id	bigint	Pengidentifikasi kueri.
label kueri	karakter (320)	Nama pendek untuk query.
id transaksi_	bigint	Pengidentifikasi transaksi.
session_id	integer	Pengidentifikasi proses dari proses yang menjalankan kueri.
database_name	karakter (128)	Nama database yang terhubung ke pengguna saat kueri dikeluarkan.
query_type	karakter (32)	Jenis kueri, seperti, SELECT, INSERT, UPDATE, UNLOAD, COPY, COMMAND, DDL, UTILITY, CTAS, dan LAINNYA.
status	karakter (10)	Status kueri. Nilai yang valid: perencanaan, antrian, menjalankan, mengembal ikan, gagal, dibatalkan, dan berhasil.
result_cache_hit	Boolean	Menunjukkan apakah kueri cocok dengan cache hasil.

Nama kolom	Jenis data	Deskripsi
waktu mulai	timestamp	Waktu ketika kueri dimulai.
akhir_waktu	timestamp	Waktu ketika kueri selesai.
waktu yang telah berlalu	bigint	Jumlah total waktu (mikrodetik) yang dihabiskan untuk kueri.
antrian_waktu	bigint	Total waktu (mikrodetik) yang dihabiskan pada antrian kueri kelas layanan.
waktu_eksekusi	bigint	Total waktu (mikrodetik) berjalan di kelas layanan.
pesan kesalahan	karakter (512)	Alasan permintaan gagal.
baris yang dikembalikan	bigint	Jumlah baris dikembalikan ke klien.
Byte_dikembalikan	bigint	Jumlah byte yang dikembalikan ke klien.
kueri_teks	karakter (4000)	String kueri. String ini mungkin terpotong.
redshift_versi	karakter (256)	Versi Amazon Redshift saat kueri berjalan.
pemakaian_batas	karakter (150)	Daftar ID batas penggunaan yang dicapai oleh kueri.

Nama kolom	Jenis data	Deskripsi
tipe komputasi	varchar(32)	Menunjukkan apakah kueri berjalan pada cluster utama atau cluster penskalaan konkurensi. Nilai yang mungkin adalah `primary` (kueri berjalan pada cluster utama), `secondary` (kueri berjalan pada cluster sekunder), atau `primary-scale` (kueri berjalan pada cluster konkurensi). Ini hanya berlaku untuk cluster yang disediakan.
compile_time	bigint	Total waktu (mikrodetik) yang dihabiskan untuk kompilasi kueri.
waktu_perencanaan	bigint	Total waktu (mikrodetik) yang dihabiskan untuk perencanaan kueri.
kunci_waktu_tunggu	bigint	Total waktu (mikrodetik) yang dihabiskan untuk menunggu kunci relasi.
id kelas layanan	integer	ID kelas layanan. Untuk daftar ID kelas layanan, bukaID kelas layanan WLM. Kolom ini hanya digunakan untuk kueri yang dijalankan pada cluster yang disediakan. Untuk kueri yang dijalankan di Redshift Serverless, kolom ini berisi -1.

Nama kolom	Jenis data	Deskripsi
nama_kelas_layanan	karakter (64)	Nama kelas layanan. Kolom ini hanya digunakan untuk kueri yang dijalankan pada cluster yang disediaka n. Untuk kueri yang dijalanka n di Amazon Redshift Redshift Serverless, kolom ini kosong.
Query_Prioritas	karakter (20)	Prioritas antrian di mana kueri berjalan. Kemungkin an nilainya adalah sebagai berikut: • NULL • terendah • rendah • biasa • tinggi • tertinggi NULL berarti prioritas kueri tidak didukung untuk kueri. Kolom ini hanya digunakan untuk kueri yang dijalankan pada cluster yang disediakan. Untuk kueri yang dijalankan di Redshift Serverless, kolom ini kosong.

Nama kolom	Jenis data	Deskripsi
short_query_dipercepat	karakter (10)	Apakah kueri dipercepat menggunakan akselerasi kueri pendek (SQA). Kemungkinan nilainya adalah sebagai berikut: • true • SALAH • NULL Kolom ini hanya digunakan untuk kueri yang dijalankan pada cluster yang disediakan. Untuk kueri yang dijalankan di Redshift Serverless, kolom ini kosong.
user_query_hash	karakter (40)	Hash kueri yang dihasilkan dari kueri, termasuk literal kuerinya. Kueri berulang dengan teks kueri yang sama akan memiliki nilai user_query_hash yang sama.
generik_query_hash	karakter (40)	Hash kueri yang dihasilkan dari kueri, tidak termasuk literal kuerinya. Kueri berulang dengan teks kueri yang sama, tetapi literal kueri yang berbeda, akan memiliki nilai generic_query_hash yang sama.
versi_hash kueri	integer	Nomor versi untuk hash kueri yang dihasilkan dari kueri.

Nama kolom	Jenis data	Deskripsi
result_cache_query_id	integer	Jika kueri menggunakan hasil caching, nilai bidang ini adalah ID kueri dari kueri yang merupakan sumber hasil cache. Jika hasil caching tidak digunakan, nilai bidang ini adalah0.
nama pengguna	karakter (128)	Nama pengguna yang mengirimkan kueri.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Kueri berikut mengembalikan kueri yang sedang berjalan dan mengantri.

```
SELECT user_id,
       query_id,
       transaction_id,
       session_id,
       status,
       trim(database_name) AS database_name,
       start_time,
       end_time,
       result_cache_hit,
       elapsed_time,
       queue_time,
       execution_time
FROM sys_query_history
WHERE status IN ('running','queued')
ORDER BY start_time;
```

Sampel keluaran.

```
 user_id | query_id | transaction_id | session_id | status  | database_name |
 start_time         |          end_time          | result_cache_hit | elapsed_time |
 queue_time | execution_time
---------+----------+----------------+------------+---------+---------------
+--------------------------+---------------------------+-----------------
+--------------+-----------+----------------
     101 |   760705 |         852337 | 1073832321 | running | tpcds_1t      |
 2022-02-15 19:03:19.67849 | 2022-02-15 19:03:19.739811 | f               |
 61321 |          0 |              0
```

Kueri berikut mengembalikan waktu mulai kueri, waktu akhir, waktu antrian, waktu berlalu, waktu perencanaan, dan metadata lainnya untuk kueri tertentu.

```
SELECT user_id,
       query_id,
       transaction_id,
       session_id,
       status,
       trim(database_name) AS database_name,
       start_time,
       end_time,
       result_cache_hit,
       elapsed_time,
       queue_time,
       execution_time,
       planning_time,
       trim(query_text) as query_text
FROM sys_query_history
WHERE query_id = 3093;
```

Sampel keluaran.

```
user_id | query_id | transaction_id | session_id |   status   | database_name |
   start_time         |          end_time          | result_cache_hit | elapsed_time |
 queue_time | execution_time | planning_time | query_text
--------+----------+----------------+------------+------------
+--------------+---------------------------+---------------------------
+-----------------+-------------+-----------+----------------+--------------
+-------------------------------------
    106 |     3093 |          11759 | 1073750146 | success    | dev           |
 2023-03-16 16:53:17.840214 | 2023-03-16 16:53:18.106588 | f               |
 266374 |          0 |         105725 |        136589 | select count(*) from item;
```

Kueri berikut mencantumkan 10 kueri SELECT terbaru.

```
SELECT query_id,
       transaction_id,
       session_id,
       start_time,
       elapsed_time,
       queue_time,
       execution_time,
       returned_rows,
       returned_bytes
FROM sys_query_history
WHERE query_type = 'SELECT'
ORDER BY start_time DESC limit 10;
```

Sampel keluaran.

```
 query_id | transaction_id | session_id |         start_time         | elapsed_time |
 queue_time | execution_time | returned_rows | returned_bytes
----------+----------------+------------+----------------------------+--------------
+------------+----------------+---------------+----------------
   526532 |          61093 | 1073840313 | 2022-02-09 04:43:24.149603 |       520571 |
        0 |         481293 |              1 |           3794
   526520 |          60850 | 1073840313 | 2022-02-09 04:38:27.24875  |       635957 |
        0 |         596601 |              1 |           3679
   526508 |          60803 | 1073840313 | 2022-02-09 04:37:51.118835 |       563882 |
        0 |         503135 |              5 |          17216
   526505 |          60763 | 1073840313 | 2022-02-09 04:36:48.636224 |       649337 |
        0 |         589823 |              1 |            652
   526478 |          60730 | 1073840313 | 2022-02-09 04:36:11.741471 |     14611321 |
        0 |       14544058 |              0 |              0
   526467 |          60636 | 1073840313 | 2022-02-09 04:34:11.91463  |     16711367 |
        0 |       16633767 |              1 |            575
   511617 |         617946 | 1074009948 | 2022-01-20 06:21:54.44481  |      9937090 |
        0 |        9899271 |            100 |          12500
   511603 |         617941 | 1074259415 | 2022-01-20 06:21:45.71744  |      8065081 |
        0 |        7582500 |            100 |           8889
   511595 |         617935 | 1074128320 | 2022-01-20 06:21:44.030876 |      1051270 |
        0 |        1014879 |              1 |             72
   511584 |         617931 | 1074030019 | 2022-01-20 06:21:42.764088 |       609033 |
        0 |         485887 |            100 |           8438
```

Kueri berikut menunjukkan jumlah kueri pilihan harian dan waktu kueri rata-rata yang telah berlalu.

```
SELECT date_trunc('day',start_time) AS exec_day,
       status,
       COUNT(*) AS query_cnt,
       AVG(datediff (microsecond,start_time,end_time)) AS elapsed_avg
FROM sys_query_history
WHERE query_type = 'SELECT'
AND start_time >= '2022-01-14'
AND start_time <= '2022-01-18'
GROUP BY exec_day,
         status
ORDER BY exec_day,
         status;
```

Sampel keluaran.

```
      exec_day       | status  | query_cnt | elapsed_avg
---------------------+---------+-----------+------------
 2022-01-14 00:00:00 | success |      5253 |  56608048
 2022-01-15 00:00:00 | success |      7004 |  56995017
 2022-01-16 00:00:00 | success |      5253 |  57016363
 2022-01-17 00:00:00 | success |      5309 |  55236784
 2022-01-18 00:00:00 | success |      8092 |  54355124
```

Kueri berikut menunjukkan kinerja waktu kueri harian yang telah berlalu.

```
SELECT distinct date_trunc('day',start_time) AS exec_day,
       query_count.cnt AS query_count,
       Percentile_cont(0.5) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P50_runtime,
       Percentile_cont(0.8) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P80_runtime,
       Percentile_cont(0.9) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P90_runtime,
       Percentile_cont(0.99) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P99_runtime,
       Percentile_cont(1.0) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS max_runtime
FROM sys_query_history
LEFT JOIN (SELECT  date_trunc('day',start_time) AS day, count(*) cnt
           FROM sys_query_history
           WHERE query_type = 'SELECT'
```

```
           GROUP by 1) query_count
ON date_trunc('day',start_time) = query_count.day
WHERE query_type = 'SELECT'
ORDER BY exec_day;
```

Sampel keluaran.

```
      exec_day       | query_count | p50_runtime | p80_runtime | p90_runtime |
 p99_runtime  | max_runtime
--------------------+-------------+-------------+-------------+-------------
+-------------+-------------
 2022-01-14 00:00:00 |        5253 |  16816922.0 |  69525096.0 | 158524917.8 |
 486322477.52 | 1582078873.0
 2022-01-15 00:00:00 |        7004 |  15896130.5 |  71058707.0 | 164314568.9 |
 500331542.07 | 1696344792.0
 2022-01-16 00:00:00 |        5253 |  15750451.0 |  72037082.2 | 159513733.4 |
 480372059.24 | 1594793766.0
 2022-01-17 00:00:00 |        5309 |  15394513.0 |  68881393.2 | 160254700.0 |
 493372245.84 | 1521758640.0
 2022-01-18 00:00:00 |        8092 |  15575286.5 |  68485955.4 | 154559572.5 |
 463552685.39 | 1542783444.0
 2022-01-19 00:00:00 |        5860 |  16648747.0 |  72470482.6 | 166485138.2 |
 492038228.67 | 1693483241.0
 2022-01-20 00:00:00 |        1751 |  15422072.0 |  69686381.0 | 162315385.0 |
 497066615.00 | 1439319739.0
 2022-02-09 00:00:00 |          13 |   6382812.0 |  17616161.6 |  21197988.4 |
 23021343.84 |   23168439.0
```

Kueri berikut menunjukkan distribusi jenis kueri.

```
SELECT query_type,
       COUNT(*) AS query_count
FROM sys_query_history
GROUP BY query_type
ORDER BY query_count DESC;
```

Sampel keluaran.

```
 query_type | query_count
------------+-------------
 UTILITY    |      134486
```

```
 SELECT     |       38537
 DDL        |        4832
 OTHER      |         768
 LOAD       |         768
 CTAS       |         748
 COMMAND    |          92
```

Contoh berikut menunjukkan perbedaan hasil hash query antara beberapa query. Amati pertanyaan berikut:

```
CREATE TABLE test_table (col1 INT);

INSERT INTO test_table VALUES (1),(2);

SELECT * FROM test_table;

SELECT * FROM test_table;

SELECT col1 FROM test_table;

SELECT * FROM test_table WHERE col1=1;

SELECT * FROM test_table WHERE col1=2;

SELECT query_id, TRIM(user_query_hash) AS user_query_hash, TRIM(generic_query_hash)
 AS generic_query_hash, TRIM(query_text) AS text FROM sys_query_history ORDER BY
 start_time
DESC LIMIT 10;
```

Berikut ini adalah contoh output:

```
query_id | user_query_hash | generic_query_hash | text
---------+-----------------+--------------------+----------
24723049 | oPuFtjEPLTs=    | oPuFtjEPLTs=       | select query_id,
 trim(user_query_hash) as user_query_hash, trim(generic_query_hash) as
 generic_query_hash, query_hash_version, trim(query_text) as text from
 sys_query_history order by start_time\r\ndesc limit 20
24723045 | Gw2Kwdd8m2I=    | IwfRu8/XAKI=       | select * from test_table where col1=2
 limit 100
24723041 | LNw2vx0GDXo=    | IwfRu8/XAKI=       | select * from test_table where col1=1
 limit 100
24723036 | H+qep/c82Y8=    | H+qep/c82Y8=       | select col1 from test_table limit 100
24723033 | H+qep/c82Y8=    | H+qep/c82Y8=       | select * from test_table limit 100
```

```
24723029 | H+qep/c82Y8=    | H+qep/c82Y8=       | select * from test_table limit 100
24723023 | 50sirx9E1hU=    | uO36Z1a/QYs=       | insert into test_table values (1),(2)
24723021 | YSVnlivZHeo=    | YSVnlivZHeo=       | create table test_table (col1 int)
```

`SELECT * FROM test_table;`dan `SELECT col1 FROM test_table;` memiliki nilai user_query_hash yang sama, karena test_table hanya memiliki satu kolom. `SELECT * FROM test_table WHERE col1=1;`dan `SELECT * FROM test_table WHERE col1=2;` memiliki nilai user_query_hash yang berbeda, tetapi nilai generic_query_hash identik, karena kedua kueri identik di luar literal kueri 1 dan 2.

SYS_QUERY_TEKS

Gunakan SYS_QUERY_TEXT untuk melihat teks kueri dari semua kueri. Setiap baris mewakili teks kueri hingga 4000 karakter dimulai dengan nomor urut 0. Ketika pernyataan query berisi lebih dari 4000 karakter, baris tambahan dicatat untuk pernyataan dengan menambah nomor urutan untuk setiap baris. Tampilan ini mencatat semua teks kueri pengguna seperti DDL, utilitas, kueri Amazon Redshift, dan kueri hanya leader-node.

SYS_QUERY_TEXT terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
Query_id	bigint	Pengidentifikasi kueri.
id transaksi_	bigint	Pengidentifikasi transaksi yang terkait dengan pernyataa n.
session_id	integer	Pengidentifikasi proses sesi yang menjalankan kueri.
waktu mulai	timestamp	Waktu ketika kueri dimulai.

Nama kolom	Jenis data	Deskripsi
urutan	integer	Ketika pernyataan tunggal berisi lebih dari 4000 karakter, baris tambahan dicatat untuk pernyataan tersebut. Urutan 0 adalah baris pertama, 1 adalah baris kedua, dan seterusnya.
text	karakter (4000)	Teks kueri SQL yang dalam penambahan 4000 karakter. Bidang ini mungkin berisi karakter khusus, seperti garis miring terbalik (\) dan baris baru (\ n).
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Kueri berikut mengembalikan kueri yang sedang berjalan dan mengantri.

```
SELECT user_id,
 query_id,
 transaction_id,
 session_id, start_time,
 sequence, trim(text) as text from sys_query_text
 ORDER BY sequence;
```

Sampel keluaran.

```
 user_id | query_id | transaction_id | session_id |        start_time          |
 sequence |                                                        text
--------+----------+----------------+------------+----------------------------
+----------
+-------------------------------------------------------------------------------------------------
```

```
   100  |     4    |       1396      | 1073750220 | 2023-04-28 16:44:55.887184 |
   0    | SELECT trim(text) as text, sequence FROM sys_query_text WHERE query_id =
 pg_last_query_id() AND user_id > 1 AND start
_time > '2023-04-28 16:44:55.922705+00:00'::timestamp order by sequence;
```

Kueri berikut mengembalikan izin yang telah diberikan atau dicabut dari grup dalam database Anda.

```
SELECT
    SPLIT_PART(text, ' ', 1) as grantrevoke,
    SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'GROUP'))), ' ', 2) as group,
    SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), ' '))), 'ON', 1) as type,
    SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'ON'))), ' ', 2) || ' ' ||
 SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'ON'))), ' ', 3) as entity
FROM SYS_QUERY_TEXT
WHERE (text LIKE 'GRANT%' OR text LIKE 'REVOKE%') AND text LIKE '%GROUP%';

+-------------+----------+--------+----------+
| grantrevoke |  group   |  type  |  entity  |
+-------------+----------+--------+----------+
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | USAGE  | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
+-------------+----------+--------+----------+
```

SYS_REDSHIFT_TEMPLATE

Gunakan SYS_REDSHIFT_TEMPLATE untuk melihat detail Redshift TEMPLATES.

Tampilan ini berisi TEMPLATE yang telah dibuat.

SYS_REDSHIFT_TEMPLATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
database_name	karakter (128)	Database Redshift tempat template dibuat.
schema_name	karakter (128)	Skema Pergeseran Merah tempat template dibuat.
nama templat	karakter (128)	Nama templat.
tipe_template	integer	Bilangan bulat yang menunjukkan tipe perintah Redshift yang terkait dengan template. 1 = perintah COPY
buat_waktu	timestamp	Timestamp saat template dibuat.
waktu_terakhir_dimodifikasi	timestamp	Stempel waktu saat template terakhir dimodifikasi.
id pemiliknya	integer	ID pengguna pengguna yang memiliki template.
terakhir_dimodifikasi oleh	integer	ID pengguna pengguna yang terakhir mengubah template.
template_parameter	text	String JSON yang berisi parameter template dan nilainya.

Kueri Sampel

Kueri berikut mengembalikan semua templat yang terlihat oleh pengguna saat ini:

```
SELECT * FROM SYS_REDSHIFT_TEMPLATE;
```

Sampel keluaran.

```
 database_name | schema_name |   template_name    | template_type |
 create_time         |     last_modified_time     | owner_id | last_modified_by |
 template_parameters
--------------+------------+-------------------+--------------
+--------------------------+---------------------------+----------
+------------------+---------------------
 dev           | s1          | shapefile_template |             1 | 2025-12-17
 22:42:02.079758 | 2025-12-17 22:42:02.079758 |      101 |              101 | {
    "SIMPLIFY_AUTO": 0.000001,
    "SHAPEFILE": 1,
    "COMPRESSION_UPDATE": 0
}
 dev           | s2          | orc_template       |             1 | 2025-12-17
 22:42:23.582815 | 2025-12-17 22:42:23.582815 |      101 |              101 | {
    "ORC": "serializetojson_default"
}
 dev           | s1          | csv_template       |             1 | 2025-12-17
 22:43:01.822361 | 2025-12-17 22:43:01.822361 |      101 |              101 | {
    "ENCRYPTED": 1,
    "CSV": 1,
    "ENCODING": 1,
    "DELIMITER": ","
}
(3 rows)
```

SYS_RESTORE_LOG

Gunakan SYS_RESTORE_LOG untuk memantau kemajuan migrasi setiap tabel di cluster selama perubahan ukuran klasik ke node RG atau RA3. Ini menangkap throughput historis migrasi data selama operasi pengubahan ukuran. Untuk informasi selengkapnya tentang mengubah ukuran klasik ke node RG atau RA3, lihat Perubahan ukuran klasik.

SYS_RESTORE_LOG hanya terlihat oleh pengguna super.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu_acara	timestamp	Stempel waktu yang menunjukkan kapan entri log direkam.
database_name	karakter (128)	Nama basis data.
schema_name	karakter (128)	Nama skema.
table_name	karakter (128)	Nama tabel.
tabel_id	integer	ID tabel.
tindakan	karakter (128)	Tindakan yang diambil pada saat entri. Nilai dapat mencakup: Migrasi dimulai, pos pemeriksaan, dilanjutkan, selesai, dibatalkan, atau diatur ulang.
ukuran tabel_	long	Ukuran meja.
total_data_diproses	long	Ukuran data dalam MB diproses hingga titik ini untuk tabel.
delta_data_diproses	long	Ukuran data yang diproses sejak pembaruan event_tim e terakhir, dalam MB. Ini membantu Anda menentuka n berapa banyak data yang telah diproses sejak interval waktu yang direkam sebelumnya. Anda dapat membandingkannya dengan table_size untuk mengetahui

Nama kolom	Jenis data	Deskripsi
		seberapa cepat pemrosesan data berlangsung.
pesan	karakter (512)	Penjelasan rinci untuk nilai di kolom tindakan.
redistribution_type	karakter (32)	Jenis redistribusi untuk tabel. Baik konversi KEY atau tugas penyeimbangan ulang EVEN. Untuk informasi selengkapnya tentang gaya distribusi, lihat Gaya distribusi.

Kueri Sampel

Kueri berikut menghitung throughput pemrosesan data, menggunakan SYS_RESTORE_LOG.

```
SELECT
    ROUND(sum(delta_data_processed) / 1024.0, 2) as data_processed_gb,
    ROUND(datediff(sec, min(event_time), max(event_time)) / 3600.0, 2) as duration_hr,
    ROUND(data_processed_gb/duration_hr, 2) as throughput_gb_per_hr
from sys_restore_log;
```

Sampel keluaran.

```
 data_processed_gb | duration_hr | throughput_gb_per_hr
-------------------+-------------+----------------------
              0.91 |        8.37 |                 0.11
(1 row)
```

Kueri berikut yang menunjukkan semua jenis redistribusi.

```
SELECT * from sys_restore_log ORDER BY event_time;
```

```
 database_name |     schema_name      |      table_name      | table_id |
 action             | total_data_processed | delta_data_processed |         event_time
        | table_size | message |   redistribution_type
```

```
--------------+----------------------+----------------------+----------
+----------------------------+----------------------+----------------------
+----------------------------+------------+---------+--------------------------
 dev           | schemaaaa877096d844d | customer_key         |   106424 |
 Redistribution started      |                    0 |                        | 2024-01-05
 02:18:00.744977 |        325 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | dp30907_t2_autokey   |   106430 |
 Redistribution started      |                    0 |                        | 2024-01-05
 02:18:02.756675 |         90 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | dp30907_t2_autokey   |   106430 |
 Redistribution completed    |                   90 |                     90 | 2024-01-05
 02:23:30.643718 |         90 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | customer_key         |   106424 |
 Redistribution completed    |                  325 |                    325 | 2024-01-05
 02:23:45.998249 |        325 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | dp30907_t1_even      |   106428 |
 Redistribution started      |                    0 |                        | 2024-01-05
 02:23:46.083849 |         30 |         | Rebalance Disteven Table
 dev           | schemaaaa877096d844d | dp30907_t5_auto_even |   106436 |
 Redistribution started      |                    0 |                        | 2024-01-05
 02:23:46.855728 |         45 |         | Rebalance Disteven Table
 dev           | schemaaaa877096d844d | dp30907_t5_auto_even |   106436 |
 Redistribution completed    |                   45 |                     45 | 2024-01-05
 02:24:16.343029 |         45 |         | Rebalance Disteven Table
 dev           | schemaaaa877096d844d | dp30907_t1_even      |   106428 |
 Redistribution completed    |                   30 |                     30 | 2024-01-05
 02:24:20.584703 |         30 |         | Rebalance Disteven Table
 dev           | schemaefd028a2a48a4c | customer_even        |   130512 |
 Redistribution started      |                    0 |                        | 2024-01-05
 04:54:55.641741 |        190 |         | Restore Disteven Table
 dev           | schemaefd028a2a48a4c | customer_even        |   130512 |
 Redistribution checkpointed |     29.4342113157737 |     29.4342113157737 | 2024-01-05
 04:55:04.770696 |        190 |         | Restore Disteven Table
(8 rows)
```

SYS_RESTORE_STATE

Gunakan SYS_RESTORE_STATE untuk memantau kemajuan migrasi setiap tabel selama perubahan ukuran klasik. Ini secara khusus berlaku ketika tipe node target adalah RG atau RA3. Untuk informasi selengkapnya tentang mengubah ukuran klasik ke node RG atau RA3, lihat Perubahan ukuran klasik.

SYS_RESTORE_STATE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan kueri.
database_name	karakter (64)	Nama database tabel.
Schema_id	integer	ID skema tabel.
tabel_id	integer	ID tabel.
table_name	karakter (128)	Nama tabel.
status redistribusi_	karakter (128)	Status kemajuan redistrib usi tabel. Nilai yang mungkin adalah `Completed` , `In progress`, dan `Pending`.
Persentage_didistribusikan kembali	float	Persentase kemajuan redistrib usi tabel. Nilai yang mungkin adalah dari 0 hingga 100%. Misalnya, nilai 25 menunjukk an bahwa 25% dari data didistribusikan kembali.
redistribution_type	karakter (32)	Jenis redistribusi untuk tabel. Baik konversi KEY atau tugas penyeimbangan ulang EVEN. Untuk informasi selengkapnya tentang gaya distribusi, lihat Gaya distribusi.

Kueri Sampel

Kueri berikut mengembalikan catatan untuk menjalankan dan mengantri kueri.

```
SELECT * FROM sys_restore_state;
```

Sampel keluaran.

```
 userid | database_name | schema_id | table_id |   table_name   | redistribution_status
 | precentage_redistributed |   redistribution_type
--------+---------------+-----------+----------+----------------
+----------------------+--------------------------+-------------------------
    1   |     test1     |   124865  |  124878  | customer_key_4 |         Pending
 |      0                   |  Rebalance Disteven Table
    1   |      dev      |   124865  |  124874  | customer_key_3 |         Pending
 |      0                   |  Rebalance Disteven Table
    1   |      dev      |   124865  |  124870  | customer_key_2 |        Completed
 |     100                  |  Rebalance Disteven Table
    1   |      dev      |   124865  |  124866  | customer_key_1 |       In progress
 |     13.52                |  Restore Distkey Table
```

Berikut ini memberi Anda status pemrosesan data.

```
SELECT
    redistribution_status, ROUND(SUM(block_count) / 1024.0, 2) AS total_size_gb
FROM sys_restore_state sys inner join stv_tbl_perm stv
    on sys.table_id = stv.id
GROUP BY sys.redistribution_status;
```

Sampel keluaran.

```
 redistribution_status | total_size_gb
-----------------------+---------------
 Completed             |          0.07
 Pending               |          0.71
 In progress           |          0.20
(3 rows)
```

SYS_SCHEMA_KUOTA_PELANGGARAN

Merekam kejadian, ID transaksi, dan informasi berguna lainnya saat kuota skema terlampaui. Tabel sistem ini adalah terjemahan dariPELANGGARAN STL_SCHEMA_KUOTA_.

R_sys_schema_quota_terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
id pemiliknya	integer	ID pemilik skema.
user_id	integer	ID pengguna yang membuat entri.
id transaksi _	bigint	ID transaksi yang terkait dengan pernyataan.
session_id	integer	ID proses yang terkait dengan pernyataan.
Schema_id	integer	Namespace atau ID skema.
schema_name	karakter (128)	Namespace atau nama skema.
kuota	integer	Jumlah ruang disk (dalam MB) yang dapat digunakan skema.
penggunaan disk	integer	Ruang disk (dalam MB) yang saat ini digunakan oleh skema.
waktu_rekor	stempel waktu tanpa zona waktu	Waktu ketika pelanggaran terjadi.

Kueri Sampel

Kueri berikut menunjukkan hasil pelanggaran kuota:

```
SELECT user_id, TRIM(schema_name) "schema_name", quota, disk_usage, record_time FROM
sys_schema_quota_violations WHERE SCHEMA_NAME = 'sales_schema' ORDER BY timestamp DESC;
```

Kueri ini mengembalikan output sampel berikut untuk skema yang ditentukan:

```
user_id| schema_name  | quota | disk_usage | record_time
-------+--------------+-------+------------+----------------------------
104    | sales_schema | 2048  | 2798       | 2020-04-20 20:09:25.494723
(1 row)
```

PENGGUNAAN_SYS_TANPA SERVER

Gunakan SYS_SERVERLESS_USAGE untuk melihat detail penggunaan sumber daya Amazon Redshift Serverless. Tampilan sistem ini tidak berlaku untuk cluster Amazon Redshift yang disediakan.

Tampilan ini berisi ringkasan penggunaan tanpa server termasuk berapa banyak kapasitas komputasi yang digunakan untuk memproses kueri dan jumlah penyimpanan terkelola Amazon Redshift yang digunakan pada perincian 1 menit. Kapasitas komputasi diukur dalam unit pemrosesan Redshift (RPU) dan diukur untuk beban kerja yang Anda jalankan per detik. RPU-seconds RPU digunakan untuk memproses kueri pada data yang dimuat di gudang data, ditanyakan dari danau data Amazon S3, atau diakses dari database operasional menggunakan kueri federasi. Amazon Redshift Serverless menyimpan informasi di SYS_SERVERLESS_USAGE selama 7 hari.

Untuk contoh tentang penagihan biaya komputasi, lihat Pen agihan untuk Amazon Redshift Serverless.

SYS_SERVERLESS_USAGE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu mulai	timestamp	Waktu ketika interval dimulai.

Nama kolom	Jenis data	Deskripsi
akhir_waktu	timestamp	Waktu ketika interval selesai.
komput_detik	double precision	Akumulasi unit komputasi (RPU) detik yang dikonsumsi selama interval waktu ini. Nilai ini menyumbang kapasitas RPU dasar yang dialokasikan untuk akun.
komput_kapasitas	double precision	Jumlah rata-rata unit komputasi (unit pemrosesan pergeseran merah, atau RPU) yang digunakan selama interval waktu ini. Nilai compute_capacity dapat diubah secara dinamis.
penyimpanan data	bigint	Ruang penyimpanan data rata-rata dalam MB yang digunakan selama interval waktu ini. Penyimpanan data yang digunakan dapat berubah secara dinamis saat data dimuat atau dihapus dari database.
data yang ditransfer lintas wilayah	bigint	Akumulasi data ditransfer untuk berbagi data lintas wilayah dalam byte selama interval waktu ini.

Nama kolom	Jenis data	Deskripsi
detik terisi	bigint	Akumulasi unit komputasi (RPU) detik diisi selama interval waktu ini. Ini dihitung setelah transaksi berakhir, dan karenanya bisa menjadi 0 saat transaksi berjalan. Gunakan charged_seconds untuk menghitung biaya untuk grup kerja Amazon Redshift Serverless. Nilai ini menyumbang kapasitas RPU yang dialokasikan untuk kelompok kerja Amazon Redshift Serverless.
dibebankan ekstra_kompute_untuk_detik_optimisasi_otomatis	bigint	Detik unit komputasi akumulasi (RPU) dibebankan untuk pengoptimalan otomatis selama interval waktu ini. Nilai ini menyumbang kapasitas RPU yang dialokasikan untuk kelompok kerja Amazon Redshift Serverless. Untuk informasi selengkapnya tentang pengoptimalan otomatis yang dijalankan dengan sumber daya komputasi tambahan, lihat. Mengalokasikan sumber daya komputasi tambahan untuk pengoptimalan database otomatis

Catatan penggunaan

- Ada situasi di mana compute_seconds adalah 0 tetapi charged_seconds lebih besar dari 0, atau sebaliknya. Ini adalah perilaku normal yang dihasilkan dari cara data direkam dalam tampilan sistem. Untuk representasi detail penggunaan tanpa server yang lebih akurat, kami sarankan untuk menggabungkan data.

Contoh

Untuk mendapatkan total biaya untuk jam RPU yang digunakan untuk interval waktu dengan menanyakan charged_seconds, jalankan kueri berikut:

```
select trunc(start_time) "Day",
(sum(charged_seconds)/3600::double precision) * <Price for 1 RPU> as cost_incurred
from sys_serverless_usage
group by 1
order by 1
```

Perhatikan bahwa mungkin ada waktu idle selama interval. Waktu idle tidak menambah RPU yang dikonsumsi.

SYS_SESSION_RIWAYAT

Gunakan SYS_SESSION_HISTORY untuk melihat informasi tentang sesi aktif saat ini dan riwayat sesi.

SYS_SESSION_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	arang (50)	Pengidentifikasi pengguna yang menghasilkan entri.
session_id	integer	ID sesi yang terkait dengan pernyataan.

Nama kolom	Jenis data	Deskripsi
database_name	arang (50)	Nama basis data.
status	char	Status sesi. Nilai yang mungkin adalah `active`, `timed out`, dan `closed`.
session_timeout	integer	Waktu maksimum dalam detik sesi tetap tidak aktif atau idle sebelum waktu habis. 0 menunjukkan bahwa tidak ada batas waktu yang ditetapkan.
waktu mulai	timestamp	Stempel waktu bahwa koneksi dibuat.
akhir_waktu	timestamp	Stempel waktu yang tersambung dihentikan.

Contoh

Berikut ini adalah contoh output dari SYS_SESSION_HISTORY.

```
select * from sys_session_history;
 user_id | session_id |   database_name  | status | session_timeout |
 start_time         |          end_time
---------+------------+------------------+--------+-----------------
+----------------------------+----------------------------
       1 | 1073971370 | dev              | closed |        0        | 2023-07-17
 15:50:12.030104 | 2023-07-17 15:50:12.123218
       1 | 1073979694 | dev              | closed |        0        | 2023-07-17
 15:50:24.117947 | 2023-07-17 15:50:24.131859
       1 | 1073873049 | dev              | closed |        0        | 2023-07-17
 15:49:29.067398 | 2023-07-17 15:49:29.070294
       1 | 1073873086 | database18127a4a | closed |        0        | 2023-07-17
 15:49:29.119018 | 2023-07-17 15:49:29.125925
       1 | 1073832112 | dev              | closed |        0        | 2023-07-17
 15:49:29.164688 | 2023-07-17 15:49:29.179631
       1 | 1073987697 | dev              | closed |        0        | 2023-07-17
 15:49:29.26749  | 2023-07-17 15:49:29.273034
```

```
      1 | 1073922429 | dev              | closed |        0        | 2023-07-17
15:49:33.35315  | 2023-07-17 15:49:33.367499
      1 | 1073766783 | dev              | closed |        0        | 2023-07-17
15:49:45.38237  | 2023-07-17 15:49:45.396902
      1 | 1073807506 | dev              | active |        0        | 2023-07-17
15:51:48        |
```

SYS_SPATIAL_SIMPLIFIKASI

Anda dapat menanyakan tampilan sistem SYS_SPATIAL_SIMPLIFY untuk mendapatkan informasi tentang objek geometri spasial yang disederhanakan menggunakan perintah COPY. Saat Anda menggunakan COPY pada shapefile, Anda dapat menentukan opsi penyerapan SIMPLIFY`tolerance`, SIMPLIFY AUTO, dan SIMPLIFY AUTO`max_tolerance`. Hasil penyederhanaan diringkas dalam tampilan sistem SYS_SPATIAL_SIMPLIFY.

Ketika SIMPLIFY AUTO `max_tolerance` diatur, tampilan ini berisi baris untuk setiap geometri yang melebihi ukuran maksimum. Ketika SIMPLIFY `tolerance` diatur, maka satu baris untuk seluruh operasi COPY disimpan. Baris ini mereferensikan ID kueri COPY dan toleransi penyederhanaan yang ditentukan.

Untuk informasi selengkapnya tentang memuat shapefile, lihat. Memuat shapefile ke Amazon Redshift

SYS_SPATIAL_SIMPLIFY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Query_id	bigint	ID kueri (perintah COPY) yang menghasilkan baris ini.
bilangan garis	bigint	Ketika `SIMPLIFY AUTO` opsi COPY ditentukan, nilai ini adalah nomor catatan rekaman yang disederhanakan dalam shapefile.
tolerans_ maksimum	double precision	Nilai toleransi jarak yang ditentukan dalam perintah COPY. Ini adalah nilai toleransi maksimum menggunak

Nama kolom	Jenis data	Deskripsi
		an SIMPLIFY AUTO opsi, atau nilai toleransi tetap menggunakan SIMPLIFY opsi.
ukuran awal	bigint	Ukuran dalam byte dari nilai GEOMETRY data sebelum penyederhanaan.
disederhanakan	harimau (1)	Ketika SIMPLIFY AUTO opsi COPY ditentukan, t jika geometri berhasil disederhanakan, atau f sebaliknya. Geometri mungkin tidak berhasil disederhanakan jika setelah penyederhanaan dengan toleransi maksimum yang diberikan ukurannya masih lebih besar dari ukuran geometri maksimum.
ukuran akhir	bigint	Ketika SIMPLIFY AUTO opsi COPY ditentukan, ini adalah ukuran dalam byte geometri setelah penyederhanaan.
tolerans_akhir	double precision	Toleransi akhir dipilih untuk penyederhanaan.

Contoh kueri

Kueri berikut mengembalikan daftar catatan yang menyederhanakan COPY.

```
SELECT * FROM sys_spatial_simplify;


 query_id | line_number | maximum_tolerance | initial_size | simplified | final_size |
  final_tolerance
----------+-------------+-------------------+--------------+------------+------------
+----------------------
    20    |     1184704 |                -1 |      1513736 | t          |    1008808 |
  1.276386653895e-05
    20    |     1664115 |                -1 |      1233456 | t          |    1023584 |
 6.11707814796635e-06
```

SYS_STREAM_SCAN_ERROR

Merekam kesalahan untuk catatan yang dimuat melalui penyerapan streaming.

SYS_STREAM_SCAN_ERRORS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_skema eksternal	karakter (128)	Nama aliran Kinesis atau skema topik Amazon MSK. Ini peka huruf besar/kecil.
nama aliran	karakter (255)	Nama aliran atau topik. Ini peka huruf besar/kecil.
mv_name	karakter (128)	Nama tampilan terwujud terkait. Kosong jika tidak ada. Ini peka huruf besar/kecil.
id transaksi_	bigint	ID transaksi.
Query_id	bigint	ID kueri.
tipe_timestamp_aliran	karakter (1)	Jenis stempel waktu aliran. Ini peka huruf besar/kecil.
stempel waktu aliran	stempel waktu tanpa zona waktu	Waktu ketika rekor itu tiba.
waktu_rekor	stempel waktu tanpa	Waktu ketika pesan kesalahan dicatat.

Nama kolom	Jenis data	Deskripsi
	zona waktu	
partisi_id	karakter (128)	I partition/shard d. Ini peka huruf besar/kecil.
posisi	karakter (128)	Posisi catatan. Ini sesuai dengan nomor urut di Kinesis atau offset di Amazon MSK. Ini peka huruf besar/kecil.
kode_kesalahan	integer	Kode kesalahan.
alasan_kesalahan	karakter (128)	Alasan kesalahan. Ini peka huruf besar/kecil.

SYS_STREAM_SCAN_STATE

Merekam status pemindaian untuk catatan yang dimuat melalui penyerapan streaming.

SYS_STREAM_SCAN_STATES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama_skema eksternal	karakter (128)	Nama skema eksternal. Ini peka huruf besar/kecil.
nama aliran	karakter (255)	Nama stream. Ini peka huruf besar/kecil.
mv_name	karakter (128)	Nama tampilan terwujud terkait. Kosong jika tidak ada. Ini peka huruf besar/kecil.

Nama kolom	Jenis data	Deskripsi
id transaksi_	bigint	ID transaksi.
Query_id	bigint	ID kueri.
waktu_rekor	stempel waktu tanpa zona waktu	Waktu ketika data dicatat.
partisi_id	karakter (128)	Partisi atau ID pecahan. Ini peka huruf besar/kecil.
posisi_terbaru	karakter (128)	Posisi catatan terakhir dibaca dalam batch. Ini sesuai dengan nomor urut di Kinesis atau offset di Amazon MSK. Ini peka huruf besar/kecil.
memindai baris	bigint	Jumlah catatan yang dipindai dalam batch.
baris yang dilewati	bigint	Jumlah catatan yang dilewati dalam batch.
scanned_byte	bigint	Jumlah byte yang dipindai dalam batch.
stream_record_time_min	stempel waktu tanpa zona waktu	Waktu kedatangan Kinesis atau Amazon MSK untuk catatan paling awal dalam batch.
stream_record_time_max	stempel waktu tanpa zona waktu	Waktu kedatangan Kinesis atau Amazon MSK untuk catatan terbaru dalam batch.

Kueri berikut menunjukkan data aliran dan topik untuk kueri tertentu.

```
select
 query_id,mv_name::varchar,external_schema_name::varchar,stream_name::varchar,sum(scanned_rows)
 total_records,
sum(scanned_bytes) total_bytes from sys_stream_scan_states where query in
 (5401180,8601939) group by 1,2,3,4;

  query_id  |    mv_name     | external_schema_name |   stream_name   | total_records |
  total_bytes
------------+----------------+----------------------+-----------------+---------------
+----------------
 5401180    | kinesistest    | kinesis              | kinesisstream   |    1493255696 |
 3209006490704
 8601939    | msktest        | msk                  | mskstream       |      14677023 |
   31056580668
(2 rows)
```

SYS_RIWAYAT TRANSAKSI_

Gunakan SYS_TRANSACTION_HISTORY untuk melihat detail transaksi saat melacak kueri.

SYS_TRANSACTION_HISTORY hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang membuat entri.
id transaksi_	bigint	ID transaksi.
tingkat isolasi	text	Tingkat isolasi transaksi . Kemungkinan nilainya adalah `Serializable` and `Snapshot Isolation` .

Nama kolom	Jenis data	Deskripsi
status	text	Status transaksi. Status yang mungkin adalah `committed` dan`rolledback` .
waktu mulai transaksi	timestamp	Waktu mulai transaksi.
waktu_mulai_komit	timestamp	Waktu mulai komit.
waktu_akhir_komitmen	timestamp	Waktu akhir komit.
blocks_diikat	bigint	Jumlah blok yang harus ditulis sebagai bagian dari komit ini.
undo_transaksi_id	bigint	ID transaksi undo jika ada transaksi yang dibatalkan. Jika tidak, nilai ini adalah -1.

Kueri Sampel

```
select * from sys_transaction_history order by transaction_start_time desc;

 user_id | transaction_id | isolation_level |   status   |   transaction_start_time
   |     commit_start_time      |      commit_end_time       | blocks_committed |
 undo_transaction_id
---------+----------------+-----------------+------------+--------------------------
+--------------------------+--------------------------+------------------
+---------------------
     100 |           1310 | Serializable    | committed  | 2023-08-27 21:03:11.822205 |
 2023-08-28 21:03:11.825287 | 2023-08-28 21:03:11.854883 |               17 |
        -1
     101 |           1345 | Serializable    | committed  | 2023-08-27 21:03:12.000278 |
 2023-08-28 21:03:12.003736 | 2023-08-28 21:03:12.030061 |               17 |
        -1
     102 |           1367 | Serializable    | committed  | 2023-08-27 21:03:12.1532   |
 2023-08-28 21:03:12.156124 | 2023-08-28 21:03:12.186468 |               17 |
        -1
     100 |           1370 | Serializable    | committed  | 2023-08-27 21:03:12.199316 |
 2023-08-28 21:03:12.204854 | 2023-08-28 21:03:12.238186 |               24 |
        -1
```

```
    100 |           1408 | Serializable    | committed  | 2023-08-27 21:03:53.891107 |
2023-08-28 21:03:53.894825 | 2023-08-28 21:03:53.927465 |               17 |
       -1
    100 |           1409 | Serializable    | rolledback | 2023-08-27 21:03:53.936431 |
2000-01-01 00:00:00        | 2023-08-28 21:04:08.712532 |                0 |
     1409
    101 |           1415 | Serializable    | committed  | 2023-08-27 21:04:24.283188 |
2023-08-28 21:04:24.289196 | 2023-08-28 21:04:24.374318 |               25 |
       -1
    101 |           1416 | Serializable    | committed  | 2023-08-27 21:04:24.38818  |
2023-08-28 21:04:24.391688 | 2023-08-28 21:04:24.415135 |               17 |
       -1
    100 |           1417 | Serializable    | rolledback | 2023-08-27 21:04:24.424252 |
2000-01-01 00:00:00        | 2023-08-28 21:04:28.354826 |                0 |
     1417
    101 |           1418 | Serializable    | rolledback | 2023-08-27 21:04:24.425195 |
2000-01-01 00:00:00        | 2023-08-28 21:04:28.680355 |                0 |
     1418
    100 |           1420 | Serializable    | committed  | 2023-08-27 21:04:28.697607 |
2023-08-28 21:04:28.702374 | 2023-08-28 21:04:28.735541 |               23 |
       -1
    101 |           1421 | Serializable    | committed  | 2023-08-27 21:04:28.744854 |
2023-08-28 21:04:28.749344 | 2023-08-28 21:04:28.779029 |               23 |
       -1
    101 |           1423 | Serializable    | committed  | 2023-08-27 21:04:28.78942  |
2023-08-28 21:04:28.791556 | 2023-08-28 21:04:28.817485 |               16 |
       -1
    100 |           1430 | Serializable    | committed  | 2023-08-27 21:04:28.917788 |
2023-08-28 21:04:28.919993 | 2023-08-28 21:04:28.944812 |               16 |
       -1
    102 |           1494 | Serializable    | committed  | 2023-08-27 21:04:37.029058 |
2023-08-28 21:04:37.033137 | 2023-08-28 21:04:37.062001 |               16 |
       -1
```

SYS_UDF_LOG

Merekam pesan kesalahan dan peringatan yang ditentukan sistem yang dihasilkan selama eksekusi fungsi yang ditentukan pengguna (UDF).

SYS_UDF_LOG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Query_id	bigint	Pengidentifikasi kueri.
function_name	text	Nama fungsi yang ditentukan pengguna.
waktu_rekor	timestamp	Waktu ketika rekaman itu dibuat.
urutan	integer	Urutan pesan log tunggal.
pesan	text	Teks pesan log.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Contoh berikut menunjukkan bagaimana UDF menangani kesalahan yang ditentukan sistem. Blok pertama menunjukkan definisi untuk fungsi UDF yang mengembalikan kebalikan dari argumen. Ketika Anda menjalankan fungsi dan memberikan 0 sebagai argumen Anda, fungsi mengembalikan kesalahan. Pernyataan terakhir mengembalikan pesan kesalahan yang dicatat di SYS_UDF_LOG.

```
-- Create a function to find the inverse of a number.
CREATE OR REPLACE FUNCTION f_udf_inv(a int)

RETURNS float

IMMUTABLE AS $$return 1/a

$$ LANGUAGE plpythonu;

-- Run the function with 0 to create an error.
Select f_udf_inv(0);

-- Query SYS_UDF_LOG to view the message.
Select query_id, record_time, message::varchar from sys_udf_log;
```

```
query_id    |    record_time              |                message
----------+----------------------------
+-----------------------------------------------------
2211        | 2023-08-23 15:53:11.360538 |  ZeroDivisionError: integer division or
 modulo by zero line 2, in f_udf_inv\n return 1/a\n
```

Contoh berikut menambahkan logging dan pesan peringatan ke UDF sehingga operasi dibagi dengan nol menghasilkan pesan peringatan alih-alih berhenti dengan pesan kesalahan.

```
-- Create a function to find the inverse of a number and log a warning if you input 0.
CREATE OR REPLACE FUNCTION f_udf_inv_log(a int)
  RETURNS float IMMUTABLE
 AS $$
  import logging
  logger = logging.getLogger() #get root logger
  if a==0:
    logger.warning('You attempted to divide by zero.\nReturning zero instead of error.
\n')
    return 0
  else:
     return 1/a
$$ LANGUAGE plpythonu;

-- Run the function with 0 to trigger the warning.
Select f_udf_inv_log(0);

-- Query SYS_UDF_LOG to view the message.
Select query_id, record_time, message::varchar from sys_udf_log;

 query_id |        record_time         |                                        message
----------+----------------------------
+-------------------------------------------------------------------------------
     0   | 2023-08-23 16:10:48.833503 | WARNING: You attempted to divide by zero.
\nReturning zero instead of error.\n
```

SYS_UNLOAD_DETAIL

Gunakan SYS_UNLOAD_DETAIL untuk melihat detail operasi UNLOAD. Ini merekam satu baris untuk setiap file yang dibuat oleh pernyataan UNLOAD. Misalnya, jika UNLOAD membuat 12 file, SYS_UNLOAD_DETAIL akan berisi 12 baris yang sesuai.

SYS_UNLOAD_DETAIL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang menghasilkan entri.
Query_id	integer	Pengidentifikasi kueri dari perintah UNLOAD.
session_id	integer	ID proses yang terkait dengan pernyataan query.
id transaksi_	bigint	ID transaksi yang terkait dengan pernyataan query.
file_name	karakter (1280)	Jalur objek Amazon S3 lengkap untuk file.
waktu mulai	timestamp	Waktu ketika transaksi dimulai.
akhir_waktu	timestamp	Waktu ketika transaksi selesai.
hitung_garis	bigint	Jumlah baris (baris) yang dibongkar ke file.
ukuran transfer	bigint	Jumlah byte yang ditransfer.
file_format	karakter (10)	Format file dari file yang dibongkar.

Kueri Sampel

Kueri berikut menunjukkan detail kueri yang tidak dimuat, termasuk format, baris, dan jumlah file dari perintah bongkar muat.

```
select query_id, substring(file_name, 0, 50), transfer_size, file_format from
 sys_unload_detail;
```

Sampel keluaran.

```
 query_id |                    substring                               | transfer_size
 | file_format
----------+--------------------------------------------------------------
+---------------+-------------
     9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0000_part_00.gz  |
 395886 | Text
     9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0001_part_00.gz  |
 406444 | Text
     9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0002_part_00.gz  |
 409431 | Text
     9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0003_part_00.gz  |
 403051 | Text
     9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0004_part_00.gz  |
 413592 | Text
     9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0005_part_00.gz  |
 395689 | Text
(6 rows)
```

SYS_UNLOAD_HISTORY

Gunakan SYS_UNLOAD_HISTORY untuk melihat detail perintah UNLOAD. Setiap baris mewakili perintah UNLOAD dengan statistik akumulasi untuk beberapa bidang. Ini berisi perintah UNLOAD yang sedang berjalan dan selesai.

SYS_UNLOAD_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan pembongkaran.
Query_id	bigint	Pengidentifikasi kueri dari perintah UNLOAD.
id transaksi_	bigint	Pengidentifikasi transaksi.
session_id	integer	Pengidentifikasi proses dari proses yang menjalankan pembongkaran.
database_name	text	Nama database yang terhubung ke pengguna saat operasi dikeluarkan.
status	text	Status perintah UNLOAD. Nilai yang valid meliputi: `running`,`completed` ,`aborted`, danunknown.
waktu mulai	timestamp	Waktu ketika pembongkaran dimulai.
akhir_waktu	timestamp	Waktu ketika pembongkaran selesai.
durasi	bigint	Jumlah waktu (mikrodetik) yang dihabiskan dalam perintah UNLOAD.
file_format	text	Format file dari file output.
compression_type	text	Jenis kompresi.

Nama kolom	Jenis data	Deskripsi
lokasi_dibongkar	text	Lokasi Amazon S3 dari file yang dibongkar.
baris yang dibongkar	bigint	Jumlah baris.
hitung_berkas yang dibongkar	bigint	Jumlah file dari file output.
ukuran_file yang dibongkar	bigint	Ukuran file dari file output.
pesan kesalahan	text	Pesan kesalahan dari perintah UNLOAD.

Kueri Sampel

Kueri berikut menunjukkan detail kueri yang tidak dimuat, termasuk format, baris, dan jumlah file dari perintah bongkar muat.

```
SELECT query_id,
       file_format,
       start_time,
       duration,
       unloaded_rows,
       unloaded_files_count
FROM sys_unload_history
ORDER BY query_id,
file_format limit 100;
```

Sampel keluaran.

```
 query_id | file_format |         start_time         | duration | unloaded_rows |
 unloaded_files_count
----------+-------------+----------------------------+----------+---------------
+----------------------
   527067 | Text        | 2022-02-09 05:18:35.844452 |  5932478 |            10 |
             1
```

SYS_USERLOG

Merekam detail untuk perubahan berikut pada pengguna database:

- Buat pengguna
- Jatuhkan pengguna
- Ubah pengguna (ganti nama)
- Ubah pengguna (ubah properti)

Anda dapat menanyakan tampilan ini untuk melihat informasi tentang grup kerja tanpa server dan cluster yang disediakan.

SYS_USERLOG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	Pengidentifikasi pengguna yang mengirimkan pembongkaran.
user_name	karakter (50)	Nama pengguna yang terpengaruh oleh perubahan.
nama_pengguna asli	karakter (50)	Nama pengguna asli dalam tindakan ganti nama. Bidang ini kosong untuk semua tindakan lainnya.
tindakan	karakter (10)	Tindakan yang terjadi. Nilai yang valid adalah ubah, buat, lepas, dan ganti nama.
has_create_db_privs	integer	Jika benar (nilai 1), pengguna telah membuat izin database.

Nama kolom	Jenis data	Deskripsi
adalah_superuser	integer	Jika benar (nilai 1), pengguna dapat memperbarui katalog sistem.
memiliki perbaruan_katalog_privs	integer	Jika benar (nilai 1), pengguna dapat memperbarui katalog sistem.
kata sandi_kedaluwarsa	timestamp	Tanggal kedaluwarsa kata sandi.
session_id	integer	ID proses.
id transaksi_	bigint	ID transaksi.
waktu_rekor	timestamp	Waktu dalam UTC saat kueri dimulai.

Kueri Sampel

Contoh berikut melakukan empat tindakan pengguna, lalu menanyakan tampilan SYS_USERLOG.

```
CREATE USER userlog1 password 'Userlog1';
ALTER USER userlog1 createdb createuser;
ALTER USER userlog1 rename  to userlog2;
DROP user userlog2;

SELECT user_id, user_name, original_user_name, action, has_create_db_privs,
 is_superuser from SYS_USERLOG order by record_time desc;
```

```
user_id |  user_name | original_user_name |  action | has_create_db_privs |
 is_superuser
--------+------------+--------------------+---------+---------------------+------------
    108 |  userlog2  |                    | drop    |                   1 |   1
    108 |  userlog2  |     userlog1       | rename  |                   1 |   1
    108 |  userlog1  |                    | alter   |                   1 |   1
    108 |  userlog1  |                    | create  |                   0 |   0
```

```
(4 rows)
```

SYS_VACUUM_HISTORY

Gunakan SYS_VACUUM_HISTORY untuk melihat detail kueri vakum. Untuk informasi tentang perintah VACUUM, lihatVAKUM.

SYS_VACUUM_HISTORY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
user_id	integer	ID pengguna yang memulai kueri.
id transaksi_	long	ID transaksi untuk pernyataan VACUUM.
Query_id	long	Pengidentifikasi kueri untuk pernyataan VACUUM. Anda dapat menggabungkan tabel ini ke tampilan SYS_QUERY _DETAIL untuk melihat pernyataan SQL individua l yang dijalankan untuk transaksi VACUUM tertentu. Jika Anda menyedot seluruh database, setiap tabel disedot dalam transaksi terpisah. Untuk operasi VACUUM otomatis, nilai ini adalah null.
database_name	text	Nama basis data.
schema_name	text	Nama skema.

Nama kolom	Jenis data	Deskripsi
table_name	text	Nama tabel.
tabel_id	integer	ID tabel.
tipe vakum	karakter	Jenis operasi VACUUM. Kemungkinan nilainya adalah sebagai berikut: • **Delete** • **Sort** • **Reindex** • **Recluster** • **Full** Untuk informasi lebih lanjut tentang jenis vakum, lihatVAKUM.
is_otomatis	boolean	`true`jika operasi adalah vakum otomatis. Atau, `false`.

Nama kolom	Jenis data	Deskripsi
status	karakter	Deskripsi aktivitas saat ini yang dilakukan sebagai bagian dari operasi vakum: • Inisialisasi • Urutkan • Gabungkan • Delete • Pilih • Gagal • Selesai • Dilewati • Membangun pesanan INTERLEAVED SORTKEY
waktu mulai	timestamp	Waktu operasi vakum dimulai.
akhir_waktu	timestamp	Waktu operasi vakum berakhir. Jika operasi sedang berlangsung, bidang ini kosong.
waktu_rekor	timestamp	Waktu operasi vakum dicatat di SYS_VACUUM_HISTORY.
durasi	integer	Jumlah mikrodetik antara awal dan akhir operasi vakum. Jika operasi vakum sedang berlangsung, bidang ini kosong.

Nama kolom	Jenis data	Deskripsi
baris sebelumnya_vakum	bigint	Jumlah sebenarnya dari baris dalam tabel ditambah baris yang dihapus yang masih disimpan di disk (menunggu untuk disedot).
ukuran_sebelum_vakum	integer	Ukuran tabel sebelum operasi vakum dimulai, dalam MB.
baris yang dapat dipulihkan	bigint	Jumlah baris yang diperkira kan operasi vakum akan diperoleh kembali sebelum memulai.
baris yang dipulihkan	bigint	Jumlah baris operasi vakum direklamasi.
blok yang dipulihkan	bigint	Jumlah blok operasi vakum direklamasi.
diurutkan baris sebelumny a_vakum	integer	Jumlah baris yang diurutkan dalam tabel sebelum operasi vakum dimulai.
diurutkan baris_setelah_vakum	integer	Jumlah tambahan baris yang diurutkan dalam tabel setelah operasi vakum selesai. Ini tidak termasuk baris yang dihitung`sortedrow s_before_vacuum` .

Pemetaan tampilan sistem untuk bermigrasi ke tampilan pemantauan SYS

Saat Anda memigrasikan klaster yang disediakan Amazon Redshift ke Amazon Redshift Tanpa Server, kueri pemantauan atau diagnostik Anda mungkin mereferensikan tampilan sistem yang hanya tersedia di kluster yang disediakan. Anda dapat memperbarui kueri Anda untuk menggunakan tampilan pemantauan SYS. Halaman ini menyediakan hanya disediakan untuk pemetaan tampilan SYS untuk Anda referensikan saat memperbarui kueri Anda.

Topik

- SYS_QUERY_HISTORY
- SYS_QUERY_DETAIL
- SYS_RESTORE_LOG
- SYS_RESTORE_STATE
- SYS_TRANSACTION_HISTORY
- SYS_QUERY_TEXT
- SYS_CONNECTION_LOG
- SYS_SESSION_HISTORY
- SYS_LOAD_DETAIL
- SYS_LOAD_HISTORY
- SYS_LOAD_ERROR_DETAIL
- SYS_UNLOAD_HISTORY
- SYS_UNLOAD_DETAIL
- SYS_COPY_REPLACEMENTS
- SYS_DATASHARE_USAGE_CONSUMER
- SYS_DATASHARE_USAGE_PRODUCER
- SYS_DATASHARE_CROSS_REGION_USAGE
- SYS_DATASHARE_CHANGE_LOG
- SYS_EXTERNAL_QUERY_DETAIL
- SYS_EXTERNAL_QUERY_ERROR
- SYS_VACUUM_HISTORY
- SYS_ANALYZE_HISTORY

- SYS_ANALYZE_COMPRESSION_HISTORY
- SYS_MV_REFRESH_HISTORY
- SYS_MV_STATE
- SYS_PROCEDURE _CALL
- SYS_PROCEDURE _MESSAGES
- SYS_UDF_LOG
- SYS_USERLOG
- SYS_SCHEMA_QUOTA_VIOLATIONS
- SYS_SPATIAL_MENYEDERHANAKAN

SYS_QUERY_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_QUERY_HISTORY.

- STL_DDLTEXT
- STL_ERROR
- STL_QUERY
- STL_UTILITYTEXT
- STL_WLM_QUERY
- STV_INFLIGHT
- STV_TERBARU
- STV_WLM_QUERY_STATE
- SVL_COMPILE
- SVL_MULTI_STATEMENT_VIOLATIONS
- SVL_QLOG
- SVL_QUERY_QUEUE_INFO
- SVL_STATEMENTTEXT
- SVL_TERMINATE

SYS_QUERY_DETAIL

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_QUERY_DETAIL.

- STL_AGGR
- STL_ALERT_EVENT_LOG
- STL_BCAST
- STL_HAPUS
- STL_DIST
- STL_JELASKAN
- STL_HASH
- STL_HASHJOIN
- STL_INSERT
- STL_LIMIT
- STL_MERGE
- STL_MERGEJOIN
- STL_NESTLOOP
- STL_PARSE
- STL_PLAN_INFO
- STL_PROJECT
- STL_QUERY_METRICS
- STL_RETURN
- STL_SAVE
- STL_SCAN
- STL_SORT
- STL_STREAM_SEGS
- STL_UNIK
- STL_WINDOW
- STV_EXEC_STATE
- STV_QUERY_METRICS
- SVCS_QUERY_RINGKASAN
- SVL_QUERY_METRICS
- SVL_QUERY_METRICS_SUMMARY

- SVL_QUERY_REPORT
- SVL_QUERY_SUMMARY
- SVV_QUERY_STATE

SYS_RESTORE_LOG

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_RESTORE_LOG.

- SVL_RESTORE_ALTER_TABLE_PROGRESS

SYS_RESTORE_STATE

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_RESTORE_STATE.

- STV_XRESTORE_ALTER_QUEUE_STATE

SYS_TRANSACTION_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_RIWAYAT TRANSAKSI_.

- STL_COMM_STATS
- STL_TR_KONFLIK
- STL_UNDONE

SYS_QUERY_TEXT

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_QUERY_TEKS.

- STL_QUERYTEXT

SYS_CONNECTION_LOG

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_CONNECTION_LOG.

- STL_CONNECTION_LOG

SYS_SESSION_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_SESSION_RIWAYAT.

- STL_SESSIONS
- STL_RESTARTED_SESSIONS
- STV_SESSION

SYS_LOAD_DETAIL

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_LOAD_DETAIL.

- STL_LOAD_COMMIT

SYS_LOAD_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_LOAD_HISTORY.

- STL_LOAD_COMMIT

SYS_LOAD_ERROR_DETAIL

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_LOAD_ERROR_DETAIL.

- STL_LOADERROR_DETAIL
- STL_LOAD_ERROR

SYS_UNLOAD_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_UNLOAD_HISTORY.

- STL_UNLOAD_LOG

SYS_UNLOAD_DETAIL

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_UNLOAD_DETAIL.

- STL_UNLOAD_LOG

SYS_COPY_REPLACEMENTS

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_COPY_PENGGANTIAN.

- STL_PENGGANTIAN

SYS_DATASHARE_USAGE_CONSUMER

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_DATASHARE_PENGGUNAAN_KONSUMEN.

- SVL_DATASHARE_USAGE_CONSUMER

SYS_DATASHARE_USAGE_PRODUCER

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_DATASHARE_PENGGUNAAN_PRODUSEN.

- SVL_DATASHARE_USAGE_PRODUCER

SYS_DATASHARE_CROSS_REGION_USAGE

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_DATASHARE_CROSS_REGION_PENGGUNAAN.

- SVL_DATASHARE_CROSS_REGION_USAGE

SYS_DATASHARE_CHANGE_LOG

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_DATASHARE_CHANGE_LOG.

- SVL_DATASHARE_CHANGE_LOG

SYS_EXTERNAL_QUERY_DETAIL

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_EXTERNAL_QUERY_DETAIL.

- SVL_FEDERATED_QUERY
- SVL_S3LIST
- SVL_S3QUERY
- SVL_S3QUERY_SUMMARY

SYS_EXTERNAL_QUERY_ERROR

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_EXTERNAL_QUERY_ERROR.

- SVL_SPECTRUM_SCAN_ERROR

SYS_VACUUM_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_VACUUM_HISTORY.

- STL_VACUUM
- SVL_VACUUM_PERCENTAGE
- SVV_VACUUM_PROGRES
- SVV_VACUUM_RINGKASAN

SYS_ANALYZE_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_ANALIS_RIWAYAT.

- STL_ANALISIS

SYS_ANALYZE_COMPRESSION_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_ANALIS_RIWAYAT KOMPRESI.

- STL_ANALISE_KOMPRESI

SYS_MV_REFRESH_HISTORY

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_MV_REFRESH RIWAYAT.

- SVL_MV_REFRESH_STATUS

SYS_MV_STATE

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_MV_STATUS.

- STL_MV_STATE

SYS_PROCEDURE _CALL

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_PROSEDUR_PANGGILAN.

- SVL_STORED_PROC_CALL

SYS_PROCEDURE _MESSAGES

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_PROSEDUR_PESAN-PESAN.

- SVL_STORED_PROC_MESSAGES

SYS_UDF_LOG

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_UDF_LOG.

- SVL_UDF_LOG

SYS_USERLOG

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_USERLOG.

- STL_USERLOG

SYS_SCHEMA_QUOTA_VIOLATIONS

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_SCHEMA_KUOTA_PELANGGARAN.

- PELANGGARAN STL_SCHEMA_KUOTA_

SYS_SPATIAL_MENYEDERHANAKAN

Beberapa atau semua kolom dalam tabel berikut juga didefinisikan dalamSYS_SPATIAL_SIMPLIFIKASI.

- SVL_SPATIAL_MENYEDERHANAKAN

Pemantauan sistem (hanya disediakan)

Tabel dan tampilan sistem berikut dapat ditanyakan pada kluster yang disediakan. Tabel dan tampilan STL dan STV berisi subset data yang ditemukan di beberapa tabel sistem. Ini memberikan akses yang lebih cepat dan lebih mudah ke data yang sering ditanyakan yang ditemukan di tabel tersebut.

Tampilan SVCS memberikan detail tentang kueri pada kluster penskalaan utama dan konkurensi. Tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan di cluster utama, dengan pengecualian SVL_STATENTTEXT. SVL_STATENTTEXT dapat berisi informasi untuk kueri yang dijalankan pada cluster penskalaan konkurensi serta cluster utama.

Topik

- Tampilan STL untuk logging
- Tabel STV untuk data snapshot
- Tampilan SVCS untuk cluster penskalaan utama dan konkurensi
- Tampilan SVL untuk cluster utama

Tampilan STL untuk logging

Tampilan sistem STL dihasilkan dari file log Amazon Redshift untuk memberikan riwayat sistem.

File-file ini berada di setiap node di cluster gudang data. Tampilan STL mengambil informasi dari log dan memformatnya menjadi tampilan yang dapat digunakan untuk administrator sistem.

Retensi log — Tampilan sistem STL mempertahankan tujuh hari riwayat log. Retensi log dijamin untuk semua ukuran cluster dan jenis node, dan tidak terpengaruh oleh perubahan beban kerja cluster. Retensi log juga tidak terpengaruh oleh status cluster, seperti saat cluster dijeda. Anda memiliki kurang dari tujuh hari riwayat log hanya dalam kasus di mana cluster baru. Anda tidak perlu mengambil tindakan apa pun untuk menyimpan log, tetapi Anda harus menyalin data log secara berkala ke tabel lain atau membongkarnya ke Amazon S3 untuk menyimpan data log yang berusia lebih dari 7 hari.

Topik

- STL_AGGR
- STL_ALERT_EVENT_LOG
- STL_ANALISIS
- STL_ANALISE_KOMPRESI
- STL_BCAST
- STL_COMM_STATS
- STL_CONNECTION_LOG
- STL_DDLTEXT
- STL_HAPUS
- STL_DISK_FULL_DIAG
- STL_DIST
- STL_ERROR
- STL_JELASKAN
- STL_FILE_SCAN
- STL_HASH
- STL_HASHJOIN
- STL_INSERT
- STL_LIMIT

- STL_LOAD_COMMIT
- STL_LOAD_ERROR
- STL_LOADERROR_DETAIL
- STL_MERGE
- STL_MERGEJOIN
- STL_MV_STATE
- STL_NESTLOOP
- STL_PARSE
- STL_PLAN_INFO
- STL_PROJECT
- STL_QUERY
- STL_QUERY_METRICS
- STL_QUERYTEXT
- STL_PENGGANTIAN
- STL_RESTARTED_SESSIONS
- STL_RETURN
- STL_S3KLIEN
- STL_S3CLIENT_ERROR
- STL_SAVE
- STL_SCAN
- PELANGGARAN STL_SCHEMA_KUOTA_
- STL_SESSIONS
- STL_SORT
- STL_SSHCLIENT_ERROR
- STL_STREAM_SEGS
- STL_TR_KONFLIK
- STL_UNDONE
- STL_UNIK
- STL_UNLOAD_LOG
- STL_PENGGUNAAN_KONTROL

- STL_USERLOG
- STL_UTILITYTEXT
- STL_VACUUM
- STL_WINDOW
- STL_WLM_ERROR
- STL_WLM_RULE_ACTION
- STL_WLM_QUERY

STL_AGGR

Menganalisis langkah-langkah eksekusi agregat untuk kueri. Langkah-langkah ini terjadi selama eksekusi fungsi agregat dan klausa GROUP BY.

STL_AGGR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_AGGR hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
slot	integer	Jumlah bucket hash.
sibuk	integer	Jumlah slot yang berisi catatan.
panjang maksimal	integer	Ukuran slot terbesar.
tbl	integer	ID tabel.
berbasis-disk	karakter (1)	Jika true (t), query dijalankan sebagai operasi berbasis disk. Jika false (f), query dijalankan di memori.
workmem	bigint	Jumlah byte memori kerja yang ditetapkan untuk langkah.

Nama kolom	Jenis data	Deskripsi
jenis	karakter (6)	Jenis langkah. Nilai yang valid adalah: • DI-HASH. Menunjukkan bahwa langkah yang digunakan dikelompokkan, agregasi tidak disortir. • POLOS. Menunjukkan bahwa langkah yang digunakan agregasi skalar yang tidak dikelompokkan. • DISORTIR. Menunjukkan bahwa langkah yang digunakan dikelompokkan, diurutkan agregasi.
mengubah ukuran	integer	Informasi ini hanya untuk penggunaan internal.
dapat dibilas	integer	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Mengembalikan informasi tentang langkah-langkah eksekusi agregat untuk SLICE 1 dan TBL 239.

```
select query, segment, bytes, slots, occupied, maxlength, is_diskbased, workmem, type
from stl_aggr where slice=1 and tbl=239
order by rows
limit 10;
```

```
 query | segment | bytes |  slots  | occupied | maxlength | is_diskbased |  workmem  |
 type
-------+---------+-------+---------+----------+-----------+--------------+-----------
+--------
   562 |       1 |     0 | 4194304 |        0 |         0 | f            | 383385600 |
 HASHED
   616 |       1 |     0 | 4194304 |        0 |         0 | f            | 383385600 |
 HASHED
   546 |       1 |     0 | 4194304 |        0 |         0 | f            | 383385600 |
 HASHED
   547 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
```

```
  685 |       1 |    32 | 4194304 |        1 |         0 | f            | 383385600 |
 HASHED
  652 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
  680 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
  658 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
  686 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
  695 |       1 |    32 | 4194304 |        1 |         0 | f            | 383385600 |
 HASHED
(10 rows)
```

STL_ALERT_EVENT_LOG

Merekam peringatan saat pengoptimal kueri mengidentifikasi kondisi yang mungkin menunjukkan masalah kinerja. Gunakan tampilan STL_ALERT_EVENT_LOG untuk mengidentifikasi peluang untuk meningkatkan kinerja kueri.

Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Untuk informasi selengkapnya, lihat Pemrosesan kueri.

STL_ALERT_EVENT_LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_ALERT_EVENT_LOG hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
pid	integer	ID proses yang terkait dengan pernyataan dan irisan. Kueri yang sama mungkin memiliki beberapa PID jika berjalan pada beberapa irisan.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
kejadian	karakter (1024)	Deskripsi peristiwa peringatan.
solusi	karakter (1024)	Solusi yang disarankan.
waktu_acara	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .

Catatan penggunaan

Anda dapat menggunakan STL_ALERT_EVENT_LOG untuk mengidentifikasi masalah potensial dalam kueri Anda, lalu ikuti praktik untuk mengoptimalkan desain database Anda dan menulis ulang Penyetelan kinerja kueri kueri Anda. STL_ALERT_EVENT_LOG mencatat peringatan berikut:

- Statistik hilang

Statistik hilang. Jalankan ANALISIS berikut pemuatan data atau pembaruan signifikan dan gunakan STATUPDATE dengan operasi COPY. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang kueri.

- Loop bersarang

  Loop bersarang biasanya merupakan produk Cartesian. Evaluasi kueri Anda untuk memastikan bahwa semua tabel yang berpartisipasi digabungkan secara efisien.

- Filter yang sangat selektif

  Rasio baris yang dikembalikan ke baris yang dipindai kurang dari 0,05. Baris yang dipindai adalah nilai `rows_pre_user_filter` dan baris yang dikembalikan adalah nilai baris dalam STL_SCAN tampilan sistem. Menunjukkan bahwa kueri memindai jumlah baris yang luar biasa besar untuk menentukan kumpulan hasil. Hal ini dapat disebabkan oleh kunci pengurutan yang hilang atau salah. Untuk informasi selengkapnya, lihat Urutkan kunci.

- Baris hantu yang berlebihan

  Pemindaian melewatkan sejumlah besar baris yang ditandai sebagai dihapus tetapi tidak disedot, atau baris yang telah dimasukkan tetapi tidak dilakukan. Untuk informasi selengkapnya, lihat Meja penyedot debu.

- Distribusi besar

  Lebih dari 1.000.000 baris didistribusikan kembali untuk gabungan hash atau agregasi. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

- Siaran besar

  Lebih dari 1.000.000 baris disiarkan untuk hash join. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

- Eksekusi serial

  Gaya redistribusi DS_DIST_ALL_INNER ditunjukkan dalam rencana kueri, yang memaksa eksekusi serial karena seluruh tabel bagian dalam didistribusikan kembali ke satu node. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

Kueri Sampel

Kueri berikut menunjukkan peristiwa peringatan untuk empat kueri.

```
SELECT query, substring(event,0,25) as event,
substring(solution,0,25) as solution,
trim(event_time) as event_time from stl_alert_event_log order by query;

 query |             event             |          solution            |     event_time

-------+-------------------------------+-------------------------------
+---------------------
  6567 | Missing query planner statist | Run the ANALYZE command      | 2014-01-03
 18:20:58
  7450 | Scanned a large number of del | Run the VACUUM command to rec| 2014-01-03
 21:19:31
  8406 | Nested Loop Join in the query | Review the join predicates to| 2014-01-04
 00:34:22
 29512 | Very selective query filter:r | Review the choice of sort key| 2014-01-06
 22:00:00

(4 rows)
```

STL_ANALISIS

Merekam rincian untuk MENGANALISA operasi.

STL_ANALYZE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_ANALIS_RIWAYAT. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	long	ID transaksi.
basis data	karakter (30)	Nama database.

Nama kolom	Jenis data	Deskripsi
tabel_id	integer	ID tabel.
status	karakter (15)	Hasil dari perintah analisis. Nilai yang mungkin adalah `Full`, `Skipped`, dan `PredicateColumn` .
baris	double	Jumlah total baris dalam tabel.
baris yang dimodifikasi	double	Jumlah total baris yang dimodifikasi sejak operasi ANALYZE terakhir.
ambang_persen	integer	Nilai `analyze_threshold_percent` parameter.
is_auto	harimau (1)	Nilai true (`t`) jika operasi menyertakan operasi analisis Amazon Redshift secara default. Nilai false (`f`) jika perintah ANALYZE dijalankan secara eksplisit.
waktu mulai	timestamp	Waktu dalam UTC saat operasi analisis mulai berjalan.
waktu akhir	timestamp	Waktu dalam UTC saat operasi analisis selesai berjalan.
waktu sebelumnya	timestamp	Waktu dalam UTC tabel dianalisis sebelumnya.
nom_predikat_cols	integer	Jumlah kolom predikat saat ini dalam tabel.
nom_baru_predikat_cols	integer	Jumlah kolom predikat baru dalam tabel sejak operasi analisis sebelumnya.
adalah_latar belakang	karakter (1)	Nilai true (`t`) jika analisis dijalankan oleh operasi analisis otomatis. Jika tidak, nilainya false (`f`).
fase analisa otomatis	karakter (100)	Dicadangkan untuk penggunaan internal.

Nama kolom	Jenis data	Deskripsi
schema_name	karakter (128)	Nama skema untuk tabel.
table_name	karakter (136)	Nama tabel.

Kueri Sampel

Contoh berikut bergabung dengan STV_TBL_PERM untuk menampilkan nama tabel dan detail eksekusi.

```
select distinct a.xid, trim(t.name) as name, a.status, a.rows, a.modified_rows,
 a.starttime, a.endtime
from stl_analyze a
join stv_tbl_perm t  on t.id=a.table_id
where name = 'users'
order by starttime;

xid    | name  | status          | rows  | modified_rows | starttime           |
 endtime
-------+-------+-----------------+-------+---------------+---------------------
+---------------------
  1582 | users | Full            | 49990 |         49990 | 2016-09-22 22:02:23 |
 2016-09-22 22:02:28
244287 | users | Full            | 24992 |         74988 | 2016-10-04 22:50:58 |
 2016-10-04 22:51:01
244712 | users | Full            | 49984 |         24992 | 2016-10-04 22:56:07 |
 2016-10-04 22:56:07
245071 | users | Skipped         | 49984 |             0 | 2016-10-04 22:58:17 |
 2016-10-04 22:58:17
245439 | users | Skipped         | 49984 |          1982 | 2016-10-04 23:00:13 |
 2016-10-04 23:00:13
(5 rows)
```

STL_ANALISE_KOMPRESI

Merekam detail untuk operasi analisis kompresi selama perintah COPY atau ANALYSIS COMPRESSION.

STL_ANALYZE_COMPRESSION terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_ANALIS_RIWAYAT KOMPRESI. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
waktu mulai	timestamp	Waktu ketika operasi analisis kompresi dimulai.
xid	bigint	ID transaksi operasi analisis kompresi.
tbl	integer	ID tabel dari tabel yang dianalisis.
nama tabelnya	karakter (128)	Nama tabel yang dianalisis.
Col	integer	Indeks kolom dalam tabel yang dianalisis untuk menentukan pengkodean kompresi.
penyandian lama	karakter (15)	Jenis pengkodean sebelum analisis kompresi.
penyandian baru	karakter (15)	Jenis pengkodean setelah analisis kompresi.
Mode	karakter (14)	Nilai yang mungkin adalah: PRASETEL Menentukan bahwa `new_encoding` ditentukan oleh perintah Amazon Redshift COPY berdasarkan tipe data kolom. Tidak ada data yang diambil sampelnya.

Nama kolom	Jenis data	Deskripsi
		PADA Menentukan bahwa `new_encoding` ditentukan oleh perintah Amazon Redshift COPY berdasarkan analisis data sampel. HANYA MENGANALISIS Menentukan bahwa `new_encoding` ditentukan oleh perintah Amazon Redshift ANALYZE COMPRESSION berdasarkan analisis data sampel. Namun, jenis pengkodean kolom yang dianalisis tidak berubah.
pengkodean kompresi terbaik	karakter (15)	Jenis pengkodean yang memberikan rasio kompresi terbaik.
Byte_yang direkomendasikan	karakter (15)	Byte yang digunakan dengan mengadopsi encoding baru.
bit_kompresi_terbaik	karakter (15)	Byte yang digunakan dengan mengadopsi pengkodean kompresi terbaik.
ndv	bigint	Jumlah nilai yang berbeda dalam baris sampel.

Kueri Sampel

Contoh berikut memeriksa rincian analisis kompresi pada `lineitem` tabel dengan perintah COPY terakhir yang dijalankan dalam sesi yang sama.

```
select xid, tbl, btrim(tablename) as tablename, col, old_encoding, new_encoding,
 best_compression_encoding, mode
from stl_analyze_compression
where xid = (select xid from stl_query where query = pg_last_copy_id()) order by col;
```

```
 xid  |  tbl   | tablename | col |  old_encoding   |  new_encoding   |
 best_compression_encoding |      mode
------+--------+-----------+-----+-----------------+-----------------
+--------------------------+---------------
 5308 | 158961 | $lineitem |   0 | mostly32        | az64            | delta
          | ON
 5308 | 158961 | $lineitem |   1 | mostly32        | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   2 | lzo             | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   3 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   4 | bytedict        | az64            | bytedict
          | ON
 5308 | 158961 | $lineitem |   5 | mostly32        | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   6 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   7 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   8 | lzo             | lzo             | lzo
          | ON
 5308 | 158961 | $lineitem |   9 | runlength       | runlength       | runlength
          | ON
 5308 | 158961 | $lineitem |  10 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |  11 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |  12 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |  13 | bytedict        | bytedict        | bytedict
          | ON
 5308 | 158961 | $lineitem |  14 | bytedict        | bytedict        | bytedict
          | ON
 5308 | 158961 | $lineitem |  15 | text255         | text255         | text255
          | ON
(16 rows)
```

STL_BCAST

Mencatat informasi tentang aktivitas jaringan selama pelaksanaan langkah-langkah kueri yang menyiarkan data. Lalu lintas jaringan ditangkap oleh jumlah baris, byte, dan paket yang dikirim

melalui jaringan selama langkah tertentu pada irisan tertentu. Durasi langkah adalah perbedaan antara waktu mulai dan akhir yang dicatat.

Untuk mengidentifikasi langkah-langkah siaran dalam kueri, cari label bcast di tampilan SVL_QUERY_SUMMARY atau jalankan perintah EXPLAIN dan kemudian cari atribut langkah yang menyertakan bcast.

STL_BCAST terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_BCAST hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik

Nama kolom	Jenis data	Deskripsi
		pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
paket	integer	Jumlah total paket yang dikirim melalui jaringan.

Kueri Sampel

Contoh berikut mengembalikan informasi siaran untuk kueri di mana ada satu atau lebih paket, dan perbedaan antara awal dan akhir kueri adalah satu detik atau lebih.

```
select query, slice, step, rows, bytes, packets, datediff(seconds, starttime, endtime)
from stl_bcast
where packets>0 and datediff(seconds, starttime, endtime)>0;
```

```
 query | slice | step | rows | bytes | packets | date_diff
-------+-------+------+------+-------+---------+-----------
   453 |     2 |    5 |    1 |   264 |       1 |         1
   798 |     2 |    5 |    1 |   264 |       1 |         1
  1408 |     2 |    5 |    1 |   264 |       1 |         1
  2993 |     0 |    5 |    1 |   264 |       1 |         1
  5045 |     3 |    5 |    1 |   264 |       1 |         1
  8073 |     3 |    5 |    1 |   264 |       1 |         1
  8163 |     3 |    5 |    1 |   264 |       1 |         1
  9212 |     1 |    5 |    1 |   264 |       1 |         1
```

```
  9873 |     1 |    5 |    1 |   264 |       1 |         1
(9 rows)
```

STL_COMM_STATS

Menyediakan metrik yang terkait dengan kinerja komit, termasuk waktu berbagai tahapan komit dan jumlah blok yang dilakukan. Kueri STL_COMMIT_STATS untuk menentukan bagian transaksi yang dihabiskan untuk komit dan berapa banyak antrian yang terjadi.

STL_COMMIT_STATS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_RIWAYAT TRANSAKSI_. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
xid	bigint	ID transaksi sedang dilakukan.
node	integer	Nomor simpul. -1adalah node pemimpin.
startqueue	timestamp	Mulai antrian untuk komit.
startwork	timestamp	Mulai dari komitmen.
endflush	timestamp	Akhir fase siram blok kotor.
endstage	timestamp	Akhir fase pementasan metadata.
endlocal	timestamp	Akhir fase komit lokal.
startglobal	timestamp	Awal fase global.
endtime	timestamp	Akhir dari komitmen.

Nama kolom	Jenis data	Deskripsi
queuelen	bigint	Jumlah transaksi yang mendahului transaksi ini dalam antrean komit.
permblocks	bigint	Jumlah blok permanen yang ada pada saat komit ini.
newblocks	bigint	Jumlah blok permanen baru pada saat komit ini.
dirtyblocks	bigint	Jumlah blok yang harus ditulis sebagai bagian dari komit ini.
headers	bigint	Jumlah header blok yang harus ditulis sebagai bagian dari komit ini.
numxids	integer	Jumlah transaksi DML aktif.
oldestxid	bigint	XID dari transaksi DML aktif tertua.
extwritelatency	bigint	Informasi ini hanya untuk penggunaan internal.
metadatawritten	int	Informasi ini hanya untuk penggunaan internal.
tombstonedblocks	big point	Informasi ini hanya untuk penggunaan internal.
tossedblocks	bigint	Informasi ini hanya untuk penggunaan internal.
batched_by	bigint	Informasi ini hanya untuk penggunaan internal.

Contoh kueri

```
select node, datediff(ms,startqueue,startwork) as queue_time,
datediff(ms, startwork, endtime) as commit_time, queuelen
from stl_commit_stats
where xid = 2574
order by node;
```

```
node | queue_time   | commit_time | queuelen
-----+--------------+-------------+---------
  -1 |            0 |         617 |        0
   0 | 444950725641 |         616 |        0
   1 | 444950725636 |         616 |        0
```

STL_CONNECTION_LOG

Mencatat upaya otentikasi dan koneksi serta pemutusan.

STL_CONNECTION_LOG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_CONNECTION_LOG. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kejadian	karakter (50)	Acara koneksi atau otentikasi.
waktu rekor	timestamp	Waktu peristiwa itu terjadi.
host jarak jauh	karakter (45)	Nama atau alamat IP host jarak jauh.
pelabuhan jarak jauh	karakter (32)	Nomor port untuk host jarak jauh.
pid	integer	ID proses yang terkait dengan pernyataan.
nama db	karakter (50)	Nama database.
nama pengguna	karakter (50)	Nama pengguna.

Nama kolom	Jenis data	Deskripsi
metode otentikasi	karakter (32)	Metode otentikasi.
durasi	integer	Durasi koneksi dalam mikrodetik.
versi ssl	karakter (50)	Versi Secure Sockets Layer (SSL). Kolom ini kosong saat SSL dinonaktifkan.
sslcipher	karakter (128)	Sandi SSL.
mtu	integer	Unit transmisi maksimum (MTU).
kompresi ssl	karakter (64)	Jenis kompresi SSL.
sslexpansion	karakter (64)	Jenis ekspansi SSL.
Iamauthguide	karakter (36)	ID otentikasi IAM untuk CloudTrail permintaan.
application_name	karakter (250)	Nama awal atau yang diperbarui dari aplikasi untuk sesi.
os_version	karakter (64)	Versi sistem operasi yang ada di mesin klien yang terhubung ke cluster Amazon Redshift Anda.
versi_driver	karakter (64)	Versi driver ODBC atau JDBC yang terhubung ke cluster Amazon Redshift Anda dari alat klien SQL pihak ketiga Anda.
nama plugin	karakter (32)	Nama plugin yang digunakan untuk terhubung ke cluster Amazon Redshift Anda.

Nama kolom	Jenis data	Deskripsi
versi_protokol	integer	Versi protokol internal yang digunakan driver Amazon Redshift saat membuat koneksi dengan server. Versi protokol dinegosiasikan antara driver dan server. Versi ini menjelaskan fitur yang tersedia. Nilai yang valid meliputi: • 0 (VERSI_PROTOKOL_SERVER DASAR) • 1 (EXTENDED_RESULT_METADATA_SERVER_PROTOCOL_VERSION) — Untuk menyimpan perjalanan pulang pergi per kueri, server mengirimkan informasi metadata set hasil tambahan. • 2 (BINARY_PROTOCOL_VERSION) - Tergantung pada tipe data dari kumpulan hasil, server mengirimkan data dalam format biner. • 3 (EXTENDED2_RESULT_METADATA_SERVER_PROTOCOL_VERSION) - Server mengirimkan informasi sensitivitas huruf besar (kolasi) dari kolom.
sessionid	karakter (36)	Pengidentifikasi unik global untuk sesi saat ini. ID sesi tetap ada melalui kegagalan node restart.
Kompresi	karakter (16)	Algoritma kompresi yang digunakan untuk koneksi.

Kueri Sampel

Untuk melihat detail untuk koneksi terbuka, jalankan kueri berikut.

```
select recordtime, username, dbname, remotehost, remoteport
from stl_connection_log
where event = 'initiating session'
and pid not in
(select pid from stl_connection_log
where event = 'disconnecting session')
order by 1 desc;
```

```
recordtime          | username    | dbname     | remotehost    | remoteport

--------------------+-------------+------------+---------------+------------
2014-11-06 20:30:06 | rdsdb       | dev        | [local]       |

2014-11-06 20:29:37 | test001     | test       | 10.49.42.138  | 11111

2014-11-05 20:30:29 | rdsdb       | dev        | 10.49.42.138  | 33333

2014-11-05 20:28:35 | rdsdb       | dev        | [local]       |
(4 rows)
```

Contoh berikut mencerminkan upaya otentikasi yang gagal dan koneksi dan pemutusan yang berhasil.

```
select event, recordtime, remotehost, username
from stl_connection_log order by recordtime;

            event      |         recordtime        |  remotehost   | username

-----------------------+---------------------------+---------------+---------
authentication failure | 2012-10-25 14:41:56.96391 | 10.49.42.138 | john

authenticated          | 2012-10-25 14:42:10.87613 | 10.49.42.138 | john

initiating session     | 2012-10-25 14:42:10.87638 | 10.49.42.138 | john

disconnecting session  | 2012-10-25 14:42:19.95992 | 10.49.42.138 | john

(4 rows)
```

Contoh berikut menunjukkan versi driver ODBC, sistem operasi pada mesin klien, dan plugin yang digunakan untuk terhubung ke cluster Amazon Redshift. Dalam contoh ini, plugin yang digunakan adalah untuk otentikasi driver ODBC standar menggunakan nama login dan kata sandi.

```
select driver_version, os_version, plugin_name from stl_connection_log;

driver_version                          |  os_version                        |
 plugin_name
-----------------------------------------+------------------------------------
+--------------------
Amazon Redshift ODBC Driver 1.4.15.0001 | Darwin 18.7.0 x86_64               | none
```

```
Amazon Redshift ODBC Driver 1.4.15.0001 | Linux 4.15.0-101-generic x86_64   | none
```

Contoh berikut menunjukkan versi sistem operasi pada mesin klien, versi driver, dan versi protokol.

```
select os_version, driver_version, protocol_version from stl_connection_log;

os_version                     |  driver_version              | protocol_version
-------------------------------+------------------------------+-----------------
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
```

STL_DDLTEXT

Menangkap pernyataan DDL berikut yang dijalankan pada sistem.

Pernyataan DDL ini mencakup kueri dan objek berikut:

- BUAT SKEMA, TABEL, TAMPILAN
- JATUHKAN SKEMA, TABEL, TAMPILAN
- MENGUBAH SKEMA, TABEL

Lihat jugaSTL_QUERYTEXT,STL_UTILITYTEXT, danSVL_STATEMENTTEXT. Tampilan ini menyediakan garis waktu perintah SQL yang dijalankan pada sistem; riwayat ini berguna untuk pemecahan masalah dan untuk membuat jejak audit dari semua aktivitas sistem.

Gunakan kolom STARTTIME dan ENDTIME untuk mengetahui pernyataan mana yang dicatat selama periode waktu tertentu. Blok panjang teks SQL dipecah menjadi baris sepanjang 200 karakter; kolom SEQUENCE mengidentifikasi fragmen teks yang termasuk dalam satu pernyataan.

STL_DDLTEXT terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
pid	integer	ID proses yang terkait dengan pernyataan.
label	karakter (320)	Baik nama file yang digunakan untuk menjalankan query atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, bidang ini kosong.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
urutan	integer	Ketika pernyataan tunggal berisi lebih dari 200 karakter, baris tambahan dicatat untuk pernyataan itu. Urutan 0 adalah baris pertama, 1 adalah yang kedua, dan seterusnya.
text	karakter (200)	Teks SQL, dalam penambahan 200 karakter. Bidang ini mungkin berisi karakter khusus seperti backslash (\\) dan newline (). \n

Kueri Sampel

Kueri berikut mengembalikan catatan yang menyertakan pernyataan DDL yang dijalankan sebelumnya.

```
select xid, starttime, sequence, substring(text,1,40) as text
from stl_ddltext order by xid desc, sequence;
```

Berikut ini adalah contoh output yang menunjukkan empat pernyataan CREATE TABLE. Pernyataan DDL muncul di `text` kolom, yang dipotong untuk keterbacaan.

```
 xid  |         starttime          | sequence |                  text
------+----------------------------+----------
+----------------------------------------
 1806 | 2013-10-23 00:11:14.709851 |        0 | CREATE TABLE supplier ( s_suppkey int4
 N
 1806 | 2013-10-23 00:11:14.709851 |        1 |  s_comment varchar(101) NOT NULL )
 1805 | 2013-10-23 00:11:14.496153 |        0 | CREATE TABLE region ( r_regionkey int4
 N
 1804 | 2013-10-23 00:11:14.285986 |        0 | CREATE TABLE partsupp ( ps_partkey int8
 1803 | 2013-10-23 00:11:14.056901 |        0 | CREATE TABLE part ( p_partkey int8 NOT
 N
 1803 | 2013-10-23 00:11:14.056901 |        1 | ner char(10) NOT NULL , p_retailprice
 nu
(6 rows)
```

Merekonstruksi SQL Tersimpan

SQL berikut mencantumkan baris yang disimpan di `text` kolom STL_DDLTEXT. Baris diurutkan oleh `xid` dan`sequence`. Jika SQL asli lebih panjang dari 200 karakter beberapa baris, STL_DDLTEXT dapat berisi beberapa baris oleh. `sequence`

```
SELECT xid, sequence, LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text)
 END, '') WITHIN GROUP (ORDER BY sequence) as query_statement
FROM stl_ddltext GROUP BY xid, sequence ORDER BY xid, sequence;
```

```
xid     |  sequence | query_statement
--------+-----------+-----------------
7886671    0          create external schema schema_spectrum_uddh\nfrom data catalog
\ndatabase 'spectrum_db_uddh'\niam_role ''\ncreate external database if not exists;
```

```
7886752    0          CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league\n(\n
  league_rank smallint,\n  prev_rank   smallint,\n  club_name   varchar(15),\n
 league_name varchar(20),\n  league_off  decimal(6,2),\n  le
7886752    1          ague_def  decimal(6,2),\n  league_spi  decimal(6,2),\n
 league_nspi smallint\n)\nROW FORMAT DELIMITED \n    FIELDS TERMINATED BY ',' \n
 LINES TERMINATED BY '\\n\\l'\nstored as textfile\nLOCATION 's
7886752    2          3://mybucket-spectrum-uddh/'\ntable properties
 ('skip.header.line.count'='1');
...
```

Untuk merekonstruksi SQL yang disimpan di `text` kolom STL_DDLTEXT, jalankan pernyataan SQL berikut. Ini mengumpulkan pernyataan DDL dari satu atau lebih segmen di `text` kolom. Sebelum menjalankan SQL yang direkonstruksi, ganti karakter khusus (\n) dengan baris baru di klien SQL Anda. Hasil pernyataan SELECT berikut mengumpulkan tiga baris secara berurutan untuk merekonstruksi SQL, di `query_statement` lapangan.

```
SELECT LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END) WITHIN
 GROUP (ORDER BY sequence) as query_statement
FROM stl_ddltext GROUP BY xid, endtime order by xid, endtime;
```

```
query_statement
--------------
create external schema schema_spectrum_uddh\nfrom data catalog\ndatabase
 'spectrum_db_uddh'\niam_role ''\ncreate external database if not exists;
CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league\n(\n  league_rank smallint,
\n  prev_rank   smallint,\n  club_name   varchar(15),\n  league_name varchar(20),\n
 league_off  decimal(6,2),\n  league_def  decimal(6,2),\n  league_spi  decimal(6,2),
\n  league_nspi smallint\n)\nROW FORMAT DELIMITED \n    FIELDS TERMINATED BY ',' \n
  LINES TERMINATED BY '\\n\\l'\nstored as textfile\nLOCATION 's3://mybucket-spectrum-
uddh/'\ntable properties ('skip.header.line.count'='1');
```

STL_HAPUS

Analisis menghapus langkah-langkah eksekusi untuk kueri.

STL_DELETE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_DELETE hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.

Nama kolom	Jenis data	Deskripsi
baris	bigint	Jumlah total baris yang diproses.
tbl	integer	ID tabel.

Kueri Sampel

Untuk membuat baris di STL_DELETE, contoh berikut menyisipkan baris ke tabel EVENT dan kemudian menghapusnya.

Pertama, masukkan baris ke tabel EVENT dan verifikasi bahwa itu telah dimasukkan.

```
insert into event(eventid,venueid,catid,dateid,eventname)
values ((select max(eventid)+1 from event),95,9,1857,'Lollapalooza');
```

```
select * from event
where eventname='Lollapalooza'
order by eventid;
```

```
 eventid | venueid | catid | dateid |  eventname   |      starttime
---------+---------+-------+--------+--------------+---------------------
    4274 |     102 |     9 |   1965 | Lollapalooza | 2008-05-01 19:00:00
    4684 |     114 |     9 |   2105 | Lollapalooza | 2008-10-06 14:00:00
    5673 |     128 |     9 |   1973 | Lollapalooza | 2008-05-01 15:00:00
    5740 |      51 |     9 |   1933 | Lollapalooza | 2008-04-17 15:00:00
    5856 |     119 |     9 |   1831 | Lollapalooza | 2008-01-05 14:00:00
    6040 |     126 |     9 |   2145 | Lollapalooza | 2008-11-15 15:00:00
    7972 |      92 |     9 |   2026 | Lollapalooza | 2008-07-19 19:30:00
    8046 |      65 |     9 |   1840 | Lollapalooza | 2008-01-14 15:00:00
    8518 |      48 |     9 |   1904 | Lollapalooza | 2008-03-19 15:00:00
    8799 |      95 |     9 |   1857 | Lollapalooza |
(10 rows)
```

Sekarang, hapus baris yang Anda tambahkan ke tabel EVENT dan verifikasi bahwa itu telah dihapus.

```
delete from event
```

```
where eventname='Lollapalooza' and eventid=(select max(eventid) from event);
```

```
select * from event
where eventname='Lollapalooza'
order by eventid;
```

```
 eventid | venueid | catid | dateid |  eventname   |     starttime
---------+---------+-------+--------+--------------+---------------------
    4274 |     102 |     9 |   1965 | Lollapalooza | 2008-05-01 19:00:00
    4684 |     114 |     9 |   2105 | Lollapalooza | 2008-10-06 14:00:00
    5673 |     128 |     9 |   1973 | Lollapalooza | 2008-05-01 15:00:00
    5740 |      51 |     9 |   1933 | Lollapalooza | 2008-04-17 15:00:00
    5856 |     119 |     9 |   1831 | Lollapalooza | 2008-01-05 14:00:00
    6040 |     126 |     9 |   2145 | Lollapalooza | 2008-11-15 15:00:00
    7972 |      92 |     9 |   2026 | Lollapalooza | 2008-07-19 19:30:00
    8046 |      65 |     9 |   1840 | Lollapalooza | 2008-01-14 15:00:00
    8518 |      48 |     9 |   1904 | Lollapalooza | 2008-03-19 15:00:00
(9 rows)
```

Kemudian kueri stl_delete untuk melihat langkah-langkah eksekusi untuk penghapusan. Dalam contoh ini, kueri mengembalikan lebih dari 300 baris, sehingga output di bawah ini dipersingkat untuk tujuan tampilan.

```
select query, slice, segment, step, tasknum, rows, tbl from stl_delete order by query;
```

```
 query | slice | segment | step | tasknum | rows |  tbl
-------+-------+---------+------+---------+------+--------
     7 |     0 |       0 |    1 |       0 |    0 | 100000
     7 |     1 |       0 |    1 |       0 |    0 | 100000
     8 |     0 |       0 |    1 |       2 |    0 | 100001
     8 |     1 |       0 |    1 |       2 |    0 | 100001
     9 |     0 |       0 |    1 |       4 |    0 | 100002
     9 |     1 |       0 |    1 |       4 |    0 | 100002
    10 |     0 |       0 |    1 |       6 |    0 | 100003
    10 |     1 |       0 |    1 |       6 |    0 | 100003
    11 |     0 |       0 |    1 |       8 |    0 | 100253
    11 |     1 |       0 |    1 |       8 |    0 | 100253
    12 |     0 |       0 |    1 |       0 |    0 | 100255
    12 |     1 |       0 |    1 |       0 |    0 | 100255
    13 |     0 |       0 |    1 |       2 |    0 | 100257
    13 |     1 |       0 |    1 |       2 |    0 | 100257
```

```
    14 |     0 |       0 |    1 |       4 |    0 | 100259
    14 |     1 |       0 |    1 |       4 |    0 | 100259
    ...
```

STL_DISK_FULL_DIAG

Mencatat informasi tentang kesalahan yang direkam saat disk penuh.

STL_DISK_FULL_DIAG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu saat ini	bigint	Hari dan waktu kesalahan dihasilkan dalam mikrodetik sejak 1 Januari 2000.
nom_simpl	bigint	Pengidentifikasi untuk node.
Query_id	bigint	Pengidentifikasi untuk kueri yang menyebabkan kesalahan.
Temp_blok	bigint	Jumlah blok sementara yang dibuat oleh query.

Kueri Sampel

Contoh berikut mengembalikan detail tentang data yang disimpan ketika ada kesalahan disk-penuh.

```
select * from stl_disk_full_diag
```

Contoh berikut mengonversi `currenttime` ke stempel waktu.

```
select '2000-01-01'::timestamp + (currenttime/1000000.0)* interval '1 second' as
 currenttime,node_num,query_id,temp_blocks from pg_catalog.stl_disk_full_diag;
```

```
        currenttime         | node_num | query_id | temp_blocks
----------------------------+----------+----------+-------------
 2019-05-18 19:19:18.609338 |        0 |   569399 |       70982
 2019-05-18 19:37:44.755548 |        0 |   569580 |       70982
 2019-05-20 13:37:20.566916 |        0 |   597424 |       70869
```

STL_DIST

Mencatat informasi tentang aktivitas jaringan selama pelaksanaan langkah-langkah kueri yang mendistribusikan data. Lalu lintas jaringan ditangkap oleh jumlah baris, byte, dan paket yang dikirim melalui jaringan selama langkah tertentu pada irisan tertentu. Durasi langkah adalah perbedaan antara waktu mulai dan akhir yang dicatat.

Untuk mengidentifikasi langkah-langkah distribusi dalam kueri, cari label dist di tampilan QUERY_SUMMARY atau jalankan perintah EXPLAIN dan kemudian cari atribut langkah yang menyertakan dist.

STL_DIST dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_DIST hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
paket	integer	Jumlah total paket yang dikirim melalui jaringan.

Kueri Sampel

Contoh berikut mengembalikan informasi distribusi untuk kueri dengan satu atau lebih paket dan durasi lebih besar dari nol.

```
select query, slice, step, rows, bytes, packets,
datediff(seconds, starttime, endtime) as duration
from stl_dist
```

```
where packets>0 and datediff(seconds, starttime, endtime)>0
order by query
limit 10;
```

```
 query  | slice | step |  rows  |  bytes  | packets | duration
--------+-------+------+--------+---------+---------+----------
    567 |     1 |    4 |  49990 | 6249564 |     707 |        1
    630 |     0 |    5 |   8798 |  408404 |      46 |        2
    645 |     1 |    4 |   8798 |  408404 |      46 |        1
    651 |     1 |    5 | 192497 | 9226320 |    1039 |        6
    669 |     1 |    4 | 192497 | 9226320 |    1039 |        4
    675 |     1 |    5 |   3766 |  194656 |      22 |        1
    696 |     0 |    4 |   3766 |  194656 |      22 |        1
    705 |     0 |    4 |    930 |   44400 |       5 |        1
 111525 |     0 |    3 |     68 |   17408 |       2 |        1
(9 rows)
```

STL_ERROR

Merekam kesalahan pemrosesan internal yang dihasilkan oleh mesin database Amazon Redshift. STL_ERROR tidak merekam kesalahan SQL atau pesan. Informasi dalam STL_ERROR berguna untuk memecahkan masalah kesalahan tertentu. Tekn AWS isi dukungan mungkin meminta Anda untuk memberikan informasi ini sebagai bagian dari proses pemecahan masalah.

STL_ERROR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Untuk daftar kode kesalahan yang dapat dihasilkan saat memuat data dengan perintah Salin, lihatReferensi kesalahan muat.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
proses	karakter (12)	Proses yang melemparkan pengecualian.
waktu rekor	timestamp	Waktu kesalahan terjadi.
pid	integer	ID proses. STL_QUERYTabel berisi ID proses dan ID kueri unik untuk kueri yang diselesaikan.
kode kesalahan	integer	Kode kesalahan yang sesuai dengan kategori kesalahan.
file	karakter (90)	Nama file sumber tempat kesalahan terjadi.
linenum	integer	Nomor baris dalam file sumber tempat kesalahan terjadi.
context	karakter (100)	Penyebab kesalahan.
kesalahan	karakter (512)	Pesan kesalahan.

Kueri Sampel

Contoh berikut mengambil informasi kesalahan dari STL_ERROR.

```
select process, errcode, linenum as line,
trim(error) as err
from stl_error;

   process    | errcode | line |                                 err
--------------+---------+------
+-----------------------------------------------------------------
 padbmaster   |    8001 |  194 | Path prefix: s3://redshift-downloads/testnulls/
venue.txt*
 padbmaster   |    8001 |  529 | Listing bucket=redshift-downloads prefix=tests/
category-csv-quotes
```

```
 padbmaster   |       2 |  190 | database "template0" is not currently accepting
 connections
 padbmaster   |      32 | 1956 | pq_flush: could not send data to client: Broken pipe
(4 rows)
```

STL_JELASKAN

Menampilkan rencana EXPLAIN untuk kueri yang telah dikirimkan untuk dieksekusi.

STL_EXPLAIN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_EXPLAIN hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_JELASKAN Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
nodeid	integer	Pengidentifikasi simpul rencana, di mana node memetakan ke satu atau lebih langkah dalam pelaksanaan kueri.
parental	integer	Rencanakan pengenal simpul untuk node induk. Node induk memiliki sejumlah node anak. Misalnya, gabungan gabungan adalah induk dari pemindaian pada tabel yang digabungkan.

Nama kolom	Jenis data	Deskripsi
plannode	karakter (400)	Teks simpul dari output EXPLAIN. Node rencana yang mengacu pada eksekusi pada node komputasi diawali dengan **XN** dalam output EXPLAIN.
info	karakter (400)	Informasi kualifikasi dan filter untuk simpul rencana. Misalnya, kondisi bergabung dan batasan klausa WHERE disertakan dalam kolom ini.
user_query_id	bigint	Pengidentifikasi kueri dari kueri yang dikirimkan pengguna, seperti yang direkam di kolom query_id dari. SYS_QUERY_JELASKAN
urutan kueri anak	integer	Nomor urutan kueri anak yang ditulis ulang di bawah kueri pengguna, dimulai dengan 1.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Pertimbangkan output EXPLAIN berikut untuk kueri gabungan agregat:

```
explain select avg(datediff(day, listtime, saletime)) as avgwait
from sales, listing where sales.listid = listing.listid;
                                QUERY PLAN

------------------------------------------------------------------------------
 XN Aggregate  (cost=6350.30..6350.31 rows=1 width=16)
  ->  XN Hash Join DS_DIST_NONE  (cost=47.08..6340.89 rows=3766 width=16)
        Hash Cond: ("outer".listid = "inner".listid)
        -> XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=12)
        -> XN Hash  (cost=37.66..37.66 rows=3766 width=12)
              -> XN Seq Scan on sales  (cost=0.00..37.66 rows=3766 width=12)
(6 rows)
```

Jika Anda menjalankan kueri ini dan ID kuerinya adalah 10, Anda dapat menggunakan tabel STL_EXPLAIN untuk melihat jenis informasi yang sama dengan yang dikembalikan perintah EXPLAIN:

```
select query,nodeid,parentid,substring(plannode from 1 for 30),
substring(info from 1 for 20) from stl_explain
where query=10 order by 1,2;

query| nodeid |parentid|           substring            |    substring
-----+--------+--------+--------------------------------+------------------
10   |      1 |      0 |XN Aggregate  (cost=6717.61..6  |
10   |      2 |      1 |  -> XN Merge Join DS_DIST_NO   | Merge Cond:("outer"
10   |      3 |      2 |       -> XN Seq Scan on lis    |
10   |      4 |      2 |       -> XN Seq Scan on sal    |
(4 rows)
```

Pertimbangkan kueri berikut:

```
select event.eventid, sum(pricepaid)
from event, sales
where event.eventid=sales.eventid
group by event.eventid order by 2 desc;

eventid |   sum
--------+----------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
...
```

Jika ID kueri ini adalah 15, kueri tampilan sistem berikut mengembalikan node rencana yang telah selesai. Dalam hal ini, urutan node dibalik untuk menunjukkan urutan eksekusi yang sebenarnya:

```
select query,nodeid,parentid,substring(plannode from 1 for 56)
from stl_explain where query=15 order by 1, 2 desc;

query|nodeid|parentid|                          substring
-----+------+--------+-----------------------------------------------------------
15   |    8 |      7 |                                -> XN Seq Scan on eve
15   |    7 |      5 |                           -> XN Hash(cost=87.98..87.9
15   |    6 |      5 |                           -> XN Seq Scan on sales(cos
15   |    5 |      4 |                      -> XN Hash Join DS_DIST_OUTER(cos
15   |    4 |      3 |                 -> XN HashAggregate(cost=862286577.07..
15   |    3 |      2 |           -> XN Sort(cost=1000862287175.47..10008622871
```

```
15   |    2 |      1 |  -> XN Network(cost=1000862287175.47..1000862287197.
15   |    1 |      0 |XN Merge(cost=1000862287175.47..1000862287197.46 rows=87
(8 rows)
```

Kueri berikut mengambil ID kueri untuk setiap paket kueri yang berisi fungsi jendela:

```
select query, trim(plannode) from stl_explain
where plannode like '%Window%';

query|                                        btrim
-----+-------------------------------------------------------------------------
26   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
27   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
(2 rows)
```

STL_FILE_SCAN

Mengembalikan file yang dibaca Amazon Redshift saat memuat data dengan menggunakan perintah COPY.

Menanyakan tampilan ini dapat membantu memecahkan masalah kesalahan pemuatan data. STL_FILE_SCAN dapat sangat membantu dengan menentukan masalah dalam pemuatan data paralel, karena beban data paralel biasanya memuat banyak file dengan satu perintah COPY.

STL_FILE_SCAN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_FILE_SCAN hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_LOAD_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
name	karakter (90)	Jalur lengkap dan nama file yang dimuat.
lini	bigint	Jumlah baris yang dibaca dari file.
byte	bigint	Jumlah byte yang dibaca dari file.
waktu muat	bigint	Jumlah waktu yang dihabiskan untuk memuat file (dalam mikrodetik).
waktu luang	Stempel waktu	Stempel waktu yang mewakili waktu Amazon Redshift mulai memproses file.
adalah_sebagian	integer	Nilai yang jika benar (1) menunjukkan file input dibagi menjadi rentang selama operasi COPY. Jika nilai ini false (0), file input tidak dibagi.
awal_offset	bigint	Nilai yang, jika file input dibagi selama operasi COPY, menunjukkan nilai offset pemisahan (dalam byte). Jika file tidak dibagi, nilai ini adalah 0.

Kueri Sampel

Kueri berikut mengambil nama dan waktu muat file apa pun yang membutuhkan lebih dari 1.000.000 mikrodetik untuk dibaca Amazon Redshift.

```
select trim(name)as name, loadtime from stl_file_scan
where loadtime > 1000000;
```

Query ini mengembalikan contoh output berikut.

```
           name             | loadtime
----------------------------+----------
 listings_pipe.txt          |  9458354
 allusers_pipe.txt          |  2963761
 allevents_pipe.txt         |  1409135
 tickit/listings_pipe.txt   |  7071087
 tickit/allevents_pipe.txt  |  1237364
 tickit/allusers_pipe.txt   |  2535138
 listings_pipe.txt          |  6706370
 allusers_pipe.txt          |  3579461
 allevents_pipe.txt         |  1313195
 tickit/allusers_pipe.txt   |  3236060
 tickit/listings_pipe.txt   |  4980108
(11 rows)
```

STL_HASH

Menganalisis langkah-langkah eksekusi hash untuk kueri.

STL_HASH terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_HASH hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
slot	integer	Jumlah total bucket hash.
sibuk	integer	Jumlah total slot yang berisi catatan.
panjang maksimal	integer	Ukuran slot terbesar.

Nama kolom	Jenis data	Deskripsi
tbl	integer	ID tabel.
berbasis-disk	karakter (1)	Jika true (t), query dilakukan sebagai operasi berbasis disk. Jika false (f), query dilakukan dalam memori.
workmem	bigint	Jumlah total byte memori kerja yang ditetapkan untuk langkah.
nom_bagian	integer	Jumlah total partisi yang dibagi tabel hash selama langkah hash.
est_baris	bigint	Perkiraan jumlah baris yang akan di-hash.
nom_blocks_diizinkan	integer	Informasi ini hanya untuk penggunaan internal.
mengubah ukuran	integer	Informasi ini hanya untuk penggunaan internal.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.
ukuran filter runtime_	integer	Ukuran filter runtime dalam byte.
maksimum_runtime_filter_ukuran	integer	Ukuran maksimum filter runtime dalam byte.

Kueri Sampel

Contoh berikut mengembalikan informasi tentang jumlah partisi yang digunakan dalam hash untuk query 720, dan menunjukkan bahwa tidak ada langkah yang berjalan pada disk.

```
select slice, rows, bytes, occupied, workmem, num_parts, est_rows,
 num_blocks_permitted, is_diskbased
from stl_hash
where query=720 and segment=5
order by slice;
```

```
 slice | rows | bytes  | occupied | workmem  | num_parts | est_rows |
 num_blocks_permitted | is_diskbased
-------+------+--------+----------+----------+-----------+----------
+----------------------+--------------
     0 |  145 | 585800 |        1 | 88866816 |        16 |        1 |
 52                   f
     1 |    0 |      0 |        0 |        0 |        16 |        1 |
 52                   f
(2 rows)
```

STL_HASHJOIN

Menganalisis langkah-langkah eksekusi gabungan hash untuk kueri.

STL_HASHJOIN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_HASHJOIN hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
tbl	integer	ID tabel.
nom_bagian	integer	Jumlah total partisi yang dibagi tabel hash selama langkah hash.
tipe gabungan	integer	Jenis bergabung untuk langkah: • 0. Kueri menggunakan penggabungan dalam. • 1. Kueri menggunakan gabungan luar kiri.

Nama kolom	Jenis data	Deskripsi
		• 2. Kueri menggunakan gabungan luar penuh. • 3. Kueri menggunakan gabungan luar kanan. • 4. Kueri menggunakan operator UNION. • 5. Kueri menggunakan kondisi IN. • 6. Informasi ini hanya untuk penggunaan internal. • 7. Informasi ini hanya untuk penggunaan internal. • 8. Informasi ini hanya untuk penggunaan internal. • 9. Informasi ini hanya untuk penggunaan internal. • 10. Informasi ini hanya untuk penggunaan internal. • 11. Informasi ini hanya untuk penggunaan internal. • 12. Informasi ini hanya untuk penggunaan internal.
hash_looped	karakter (1)	Informasi ini hanya untuk penggunaan internal.
bagian-bagian yang dialihkan	karakter (1)	Menunjukkan apakah sisi build (atau luar) dan probe (atau bagian dalam) telah beralih.
used_prefetching	karakter (1)	Informasi ini hanya untuk penggunaan internal.
segmen_hash	integer	Segmen langkah hash yang sesuai.
langkah hash	integer	Nomor langkah langkah langkah hash yang sesuai.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.
distribusi	integer	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Contoh berikut mengembalikan jumlah partisi yang digunakan dalam hash join untuk query 720.

```
select query, slice, tbl, num_parts
from stl_hashjoin
where query=720 limit 10;
```

```
 query | slice | tbl | num_parts
-------+-------+-----+-----------
   720 |     0 | 243 |         1
   720 |     1 | 243 |         1
(2 rows)
```

STL_INSERT

Analisis menyisipkan langkah-langkah eksekusi untuk kueri.

STL_INSERT terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_INSERT hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
tbl	integer	ID tabel.
nilai_mega yang disisipkan	karakter (1)	Informasi ini hanya untuk penggunaan internal. Informasi ini menunjukkan apakah langkah penyisipan yang diberikan telah memasukkan nilai yang besar. Nilai besar akan disimpan dalam beberapa blok. Ukuran blok adalah 1 MB secara default, nilai besar lebih besar dari 1 MB dalam pengaturan default.

Kueri Sampel

Contoh berikut mengembalikan langkah eksekusi sisipkan untuk kueri terbaru.

```
select slice, segment, step, tasknum, rows, tbl
from stl_insert
where query=pg_last_query_id();
```

```
 slice | segment | step | tasknum | rows  |  tbl
-------+---------+------+---------+-------+--------
     0 |       2 |    2 |      15 | 24958 | 100548
     1 |       2 |    2 |      15 | 25032 | 100548
(2 rows)
```

STL_LIMIT

Menganalisis langkah-langkah eksekusi yang terjadi ketika klausa LIMIT digunakan dalam kueri SELECT.

STL_LIMIT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_LIMIT hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Untuk menghasilkan baris di STL_LIMIT, contoh ini pertama-tama menjalankan kueri berikut terhadap tabel VENUE menggunakan klausa LIMIT.

```
select * from venue
order by 1
limit 10;
```

```
 venueid |         venuename          |    venuecity    | venuestate | venueseats
---------+----------------------------+-----------------+------------+------------
       1 | Toyota Park                | Bridgeview      | IL         |          0
       2 | Columbus Crew Stadium      | Columbus        | OH         |          0
```

```
       3 | RFK Stadium                | Washington      | DC         |          0
       4 | CommunityAmerica Ballpark  | Kansas City     | KS         |          0
       5 | Gillette Stadium           | Foxborough      | MA         |      68756
       6 | New York Giants Stadium    | East Rutherford | NJ         |      80242
       7 | BMO Field                  | Toronto         | ON         |          0
       8 | The Home Depot Center      | Carson          | CA         |          0
       9 | Dick's Sporting Goods Park | Commerce City   | CO         |          0
      10 | Pizza Hut Park             | Frisco          | TX         |          0
(10 rows)
```

Selanjutnya, jalankan kueri berikut untuk menemukan ID kueri dari kueri terakhir yang Anda jalankan terhadap tabel VENUE.

```
select max(query)
from stl_query;
```

```
  max
--------
 127128
(1 row)
```

Secara opsional, Anda dapat menjalankan kueri berikut untuk memverifikasi bahwa ID kueri sesuai dengan kueri LIMIT yang Anda jalankan sebelumnya.

```
select query, trim(querytxt)
from stl_query
where query=127128;
```

```
 query  |                 btrim
--------+----------------------------------------
 127128 | select * from venue order by 1 limit 10;
(1 row)
```

Terakhir, jalankan query berikut untuk mengembalikan informasi tentang permintaan LIMIT dari tabel STL_LIMIT.

```
select slice, segment, step, starttime, endtime, tasknum
from stl_limit
where query=127128
```

```
order by starttime, endtime;
```

```
  slice | segment | step |         starttime          |          endtime           |
 tasknum
 -------+---------+------+----------------------------+----------------------------
+---------
      1 |       1 |    3 | 2013-09-06 22:56:43.608114 | 2013-09-06 22:56:43.609383 |
   15
      0 |       1 |    3 | 2013-09-06 22:56:43.608708 | 2013-09-06 22:56:43.609521 |
   15
  10000 |       2 |    2 | 2013-09-06 22:56:43.612506 | 2013-09-06 22:56:43.612668 |
    0
(3 rows)
```

STL_LOAD_COMMIT

Mengembalikan informasi untuk melacak atau memecahkan masalah pemuatan data.

Tampilan ini mencatat kemajuan setiap file data saat dimuat ke dalam tabel database.

STL_LOAD_COMMITS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_LOAD_COMMITS hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_LOAD_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Irisan dimuat untuk entri ini.
name	karakter (256)	System-defined nilai.
nama berkas	karakter (256)	Nama file yang dilacak.
Byte_offset	integer	Informasi ini hanya untuk penggunaan internal.
garis_dip indai	integer	Jumlah baris yang dipindai dari file muat. Nomor ini mungkin tidak cocok dengan jumlah baris yang sebenarnya dimuat. Misalnya, beban dapat memindai tetapi mentolerir sejumlah catatan buruk, berdasarkan opsi MAXERROR dalam perintah COPY.
kesalahan	integer	Informasi ini hanya untuk penggunaan internal.
waktu luang	timestamp	Waktu entri ini terakhir diperbarui.
status	integer	Informasi ini hanya untuk penggunaan internal.
file_format	karakter (16)	Format file muat. Kemungkinan nilainya adalah sebagai berikut: • Avro • JSON • ORC • Parquet • Teks
adalah_pa rsial	integer	Nilai yang jika benar (1) menunjukkan file input dibagi menjadi rentang selama operasi COPY. Jika nilai ini false (0), file input tidak dibagi.

Nama kolom	Jenis data	Deskripsi
awal_offset	bigint	Nilai yang, jika file input dibagi selama operasi COPY, menunjukkan nilai offset pemisahan (dalam byte). Setiap pemisahan file dicatat sebagai catatan terpisah dengan nilai start_offset yang sesuai. Jika file tidak dibagi, nilai ini adalah 0.
salin_job_id	bigint	Pengidentifikasi pekerjaan menyalin. A `0` menunjukkan tidak ada pengidentifikasi pekerjaan.

Kueri Sampel

Contoh berikut mengembalikan detail untuk operasi COPY terakhir.

```
select query, trim(filename) as file, curtime as updated
from stl_load_commits
where query = pg_last_copy_id();

 query |               file               |          updated
-------+----------------------------------+----------------------------
 28554 | s3://dw-tickit/category_pipe.txt | 2013-11-01 17:14:52.648486
(1 row)
```

Kueri berikut berisi entri untuk memuat tabel baru di database TICKIT:

```
select query, trim(filename), curtime
from stl_load_commits
where filename like '%tickit%' order by query;
```

```
 query |           btrim           |          curtime
-------+---------------------------+----------------------------
 22475 | tickit/allusers_pipe.txt  | 2013-02-08 20:58:23.274186
 22478 | tickit/venue_pipe.txt     | 2013-02-08 20:58:25.070604
 22480 | tickit/category_pipe.txt  | 2013-02-08 20:58:27.333472
 22482 | tickit/date2008_pipe.txt  | 2013-02-08 20:58:28.608305
 22485 | tickit/allevents_pipe.txt | 2013-02-08 20:58:29.99489
 22487 | tickit/listings_pipe.txt  | 2013-02-08 20:58:37.632939
```

```
 22593 | tickit/allusers_pipe.txt  | 2013-02-08 21:04:08.400491
 22596 | tickit/venue_pipe.txt     | 2013-02-08 21:04:10.056055
 22598 | tickit/category_pipe.txt  | 2013-02-08 21:04:11.465049
 22600 | tickit/date2008_pipe.txt  | 2013-02-08 21:04:12.461502
 22603 | tickit/allevents_pipe.txt | 2013-02-08 21:04:14.785124
 22605 | tickit/listings_pipe.txt  | 2013-02-08 21:04:20.170594

(12 rows)
```

Fakta bahwa catatan ditulis ke file log untuk tampilan sistem ini tidak berarti bahwa beban berhasil dilakukan sebagai bagian dari transaksi yang berisi. Untuk memverifikasi load commit, kueri tampilan STL_UTILITYTEXT dan cari catatan COMMIT yang sesuai dengan transaksi COPY. Misalnya, kueri ini bergabung dengan STL_LOAD_COMMITS dan STL_QUERY berdasarkan subquery terhadap STL_UTILITYTEXT:

```
select l.query,rtrim(l.filename),q.xid
from stl_load_commits l, stl_query q
where l.query=q.query
and exists
(select xid from stl_utilitytext where xid=q.xid and rtrim("text")='COMMIT');

 query |           rtrim           |  xid
-------+---------------------------+-------
 22600 | tickit/date2008_pipe.txt  | 68311
 22480 | tickit/category_pipe.txt  | 68066
  7508 | allusers_pipe.txt         | 23365
  7552 | category_pipe.txt         | 23415
  7576 | allevents_pipe.txt        | 23429
  7516 | venue_pipe.txt            | 23390
  7604 | listings_pipe.txt         | 23445
 22596 | tickit/venue_pipe.txt     | 68309
 22605 | tickit/listings_pipe.txt  | 68316
 22593 | tickit/allusers_pipe.txt  | 68305
 22485 | tickit/allevents_pipe.txt | 68071
  7561 | allevents_pipe.txt        | 23429
  7541 | category_pipe.txt         | 23415
  7558 | date2008_pipe.txt         | 23428
 22478 | tickit/venue_pipe.txt     | 68065
   526 | date2008_pipe.txt         |  2572
  7466 | allusers_pipe.txt         | 23365
 22482 | tickit/date2008_pipe.txt  | 68067
 22598 | tickit/category_pipe.txt  | 68310
 22603 | tickit/allevents_pipe.txt | 68315
```

```
 22475 | tickit/allusers_pipe.txt  | 68061
   547 | date2008_pipe.txt         |  2572
 22487 | tickit/listings_pipe.txt  | 68072
  7531 | venue_pipe.txt            | 23390
  7583 | listings_pipe.txt         | 23445
(25 rows)
```

Contoh berikut menyoroti nilai kolom is_partial dan start_offset.

```
-- Single large file copy without scan range
SELECT count(*) FROM stl_load_commits WHERE query = pg_last_copy_id();
1

-- Single large uncompressed, delimited file copy with scan range
SELECT count(*) FROM stl_load_commits WHERE query = pg_last_copy_id();
16

-- Scan range offset logging in the file at 64MB boundary.
SELECT start_offset FROM stl_load_commits
WHERE query = pg_last_copy_id() ORDER BY start_offset;
0
67108864
134217728
201326592
268435456
335544320
402653184
469762048
536870912
603979776
671088640
738197504
805306368
872415232
939524096
1006632960
```

STL_LOAD_ERROR

Menampilkan catatan semua kesalahan pemuatan Amazon Redshift.

STL_LOAD_ERRORS berisi riwayat semua kesalahan pemuatan Amazon Redshift. Lihat Referensi kesalahan muat untuk daftar lengkap kemungkinan kesalahan beban dan penjelasan.

Kueri STL_LOADERROR_DETAIL untuk detail tambahan, seperti baris data dan kolom yang tepat tempat kesalahan penguraian terjadi, setelah Anda menanyakan STL_LOAD_ERRORS untuk mengetahui informasi umum tentang kesalahan.

STL_LOAD_ERRORS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_LOAD_ERRORS hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_LOAD_ERROR_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
mengiris	integer	Iris tempat kesalahan terjadi.
tbl	integer	ID tabel.
waktu mulai	timestamp	Waktu mulai dalam UTC untuk beban.
sesi	integer	ID sesi untuk sesi yang melakukan pemuatan.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
nama berkas	karakter (256)	Jalur lengkap ke file input untuk beban.

Nama kolom	Jenis data	Deskripsi
bilangan garis	bigint	Nomor baris dalam file muat dengan kesalahan. Untuk COPY dari JSON, nomor baris dari baris terakhir objek JSON dengan kesalahan.
colname	karakter (127)	Bidang dengan kesalahan.
jenis	karakter (10)	Jenis data bidang.
col_length	karakter (10)	Panjang kolom, jika berlaku. Bidang ini diisi ketika tipe data memiliki batas panjang. Misalnya, untuk kolom dengan tipe data “karakter (3)”, kolom ini akan berisi nilai “3".
posisi	integer	Posisi kesalahan di lapangan.
raw_line	karakter (1024)	Data beban mentah yang berisi kesalahan. Karakter multibyte dalam data beban diganti dengan titik.
nilai_bidang raw_	karakter (1024)	Nilai pra-penguraian untuk bidang “colname” yang menyebabkan kesalahan penguraian.
kode kesalahan	integer	Kode kesalahan.
alasan_kesalahan	karakter (100)	Penjelasan untuk kesalahan tersebut.
adalah_sebagian	integer	Nilai yang jika benar (1) menunjukkan file input dibagi menjadi rentang selama operasi COPY. Jika nilai ini false (0), file input tidak dibagi.
awal_offset	bigint	Nilai yang, jika file input dibagi selama operasi COPY, menunjukkan nilai offset pemisahan (dalam byte). Jika nomor baris dalam file tidak diketahui, nomor baris adalah -1. Jika file tidak dibagi, nilai ini adalah 0.

Nama kolom	Jenis data	Deskripsi
salin_job_id	bigint	Pengidentifikasi pekerjaan menyalin. A `0` menunjukkan tidak ada pengidentifikasi pekerjaan.

Kueri Sampel

Kueri berikut menggabungkan STL_LOAD_ERRORS ke STL_LOADERROR_DETAIL untuk melihat detail kesalahan yang terjadi selama pemuatan terbaru.

```
select d.query, substring(d.filename,14,20),
d.line_number as line,
substring(d.value,1,16) as value,
substring(le.err_reason,1,48) as err_reason
from stl_loaderror_detail d, stl_load_errors le
where d.query = le.query
and d.query = pg_last_copy_id();

 query |    substring      | line |  value   |             err_reason
-------+-------------------+------+----------+----------------------------
   558| allusers_pipe.txt |  251 | 251      | String contains invalid or
                                              unsupported UTF8 code
   558| allusers_pipe.txt |  251 | ZRU29FGR | String contains invalid or
                                              unsupported UTF8 code
   558| allusers_pipe.txt |  251 | Kaitlin  | String contains invalid or
                                              unsupported UTF8 code
   558| allusers_pipe.txt |  251 | Walter   | String contains invalid or
                                              unsupported UTF8 code
```

Contoh berikut menggunakan STL_LOAD_ERRORS dengan STV_TBL_PERM untuk membuat tampilan baru, dan kemudian menggunakan tampilan itu untuk menentukan kesalahan apa yang terjadi saat memuat data ke tabel EVENT:

```
create view loadview as
(select distinct tbl, trim(name) as table_name, query, starttime,
trim(filename) as input, line_number, colname, err_code,
trim(err_reason) as reason
from stl_load_errors sl, stv_tbl_perm sp
where sl.tbl = sp.id);
```

Selanjutnya, query berikut benar-benar mengembalikan kesalahan terakhir yang terjadi saat memuat tabel EVENT:

```
select table_name, query, line_number, colname, starttime,
trim(reason) as error
from loadview
where table_name ='event'
order by line_number limit 1;
```

Kueri mengembalikan kesalahan pemuatan terakhir yang terjadi untuk tabel EVENT. Jika tidak terjadi kesalahan pemuatan, kueri mengembalikan nol baris. Dalam contoh ini, query mengembalikan kesalahan tunggal:

```
 table_name | query | line_number | colname | error | starttime
------+-----+----+----+-------------------------------------------------------
+----------------------
event | 309 |  0 |  5 | Error in Timestamp value or format [%Y-%m-%d %H:%M:%S] |
 2014-04-22 15:12:44

(1 row)
```

Dalam kasus di mana perintah COPY secara otomatis membagi data file besar, tidak terkompresi, dan dibatasi teks untuk memfasilitasi paralelisme, kolom line_number, is_partial, dan start_offset menampilkan informasi yang berkaitan dengan pem is ahan. (Nomor baris dapat tidak diketahui dalam kasus di mana nomor baris dari file asli tidak tersedia.)

```
--scan ranges information
SELECT line_number, POSITION, btrim(raw_line), btrim(raw_field_value),
btrim(err_reason), is_partial, start_offset FROM stl_load_errors
WHERE query = pg_last_copy_id();

--result
-1,51,"1008771|13463413|463414|2|28.00|38520.72|0.06|0.07|NO|1998-08-30|1998-09-25|
1998-09-04|TAKE BACK RETURN|RAIL|ans cajole sly","NO","Char length exceeds DDL
 length",1,67108864
```

STL_LOADERROR_DETAIL

Menampilkan log kesalahan penguraian data yang terjadi saat menggunakan perintah COPY untuk memuat tabel. Untuk menghemat ruang disk, maksimum 20 kesalahan per irisan node dicatat untuk setiap operasi pemuatan.

Kesalahan penguraian terjadi ketika Amazon Redshift tidak dapat mengurai bidang di baris data saat memuatnya ke dalam tabel. Misalnya, jika kolom tabel mengharapkan tipe data integer dan file data berisi string huruf di bidang itu, itu menyebabkan kesalahan parse.

Kueri STL_LOADERROR_DETAIL untuk detail tambahan, seperti baris data dan kolom yang tepat tempat kesalahan penguraian terjadi, setelah Anda menanyakan STL_LOAD_ERROR untuk mengetahui informasi umum tentang kesalahan.

Tampilan STL_LOADERROR_DETAIL berisi semua kolom data termasuk dan sebelum kolom tempat kesalahan penguraian terjadi. Gunakan bidang NILAI untuk melihat nilai data yang sebenarnya diurai di kolom ini, termasuk kolom yang diurai dengan benar hingga kesalahan.

Tampilan ini terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_LOADERROR_DETAIL hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_LOAD_ERROR_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
mengiris	integer	Iris tempat kesalahan terjadi.
sesi	integer	ID sesi untuk sesi yang melakukan pemuatan.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
nama berkas	karakter (256)	Jalur lengkap ke file input untuk beban.
bilangan garis	bigint	Nomor baris dalam file muat dengan kesalahan.
bidang	integer	Bidang dengan kesalahan.
colname	karakter (1024)	Nama kolom.
nilai	karakter (1024)	Nilai data yang diurai dari bidang. (Mungkin terpotong.) Karakter multibyte dalam data beban diganti dengan titik.
adalah_nol	integer	Apakah nilai yang diurai adalah null atau tidak.
jenis	karakter (10)	Jenis data bidang.
col_length	karakter (10)	Panjang kolom, jika berlaku. Bidang ini diisi ketika tipe data memiliki batas panjang. Misalnya, untuk kolom dengan tipe data “karakter (3)”, kolom ini akan berisi nilai “3".

Contoh kueri

Kueri berikut menggabungkan STL_LOAD_ERRORS ke STL_LOADERROR_DETAIL untuk melihat detail kesalahan penguraian yang terjadi saat memuat tabel EVENT, yang memiliki ID tabel 100133:

```
select d.query, d.line_number, d.value,
le.raw_line, le.err_reason
from stl_loaderror_detail d, stl_load_errors le
where
d.query = le.query
and tbl = 100133;
```

Output sampel berikut menunjukkan kolom yang berhasil dimuat, termasuk kolom dengan kesalahan. Dalam contoh ini, dua kolom berhasil dimuat sebelum kesalahan penguraian terjadi di kolom ketiga, di mana string karakter salah diurai untuk bidang yang mengharapkan bilangan bulat. Karena

bidang mengharapkan bilangan bulat, ia mengurai string “aaa”, yang merupakan data yang tidak diinisialisasi, sebagai nol dan menghasilkan kesalahan penguraian. Output menunjukkan nilai mentah, nilai yang diurai, dan alasan kesalahan:

```
query  | line_number | value | raw_line | err_reason
-------+-------------+-------+----------+----------------
4      |      3      |  1201 |  1201    | Invalid digit
4      |      3      |   126 |   126    | Invalid digit
4      |      3      |       |   aaa    | Invalid digit
(3 rows)
```

Ketika kueri bergabung dengan STL_LOAD_ERRORS dan STL_LOADERROR_DETAIL, itu menampilkan alasan kesalahan untuk setiap kolom di baris data, yang berarti bahwa kesalahan terjadi di baris itu. Baris terakhir dalam hasil adalah kolom aktual tempat kesalahan penguraian terjadi.

STL_MERGE

Analisis menggabungkan langkah-langkah eksekusi untuk kueri. Langkah-langkah ini terjadi ketika hasil operasi paralel (seperti pengurutan dan gabungan) digabungkan untuk pemrosesan selanjutnya.

STL_MERGE dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_MERGE hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.

Kueri Sampel

Contoh berikut mengembalikan 10 hasil eksekusi gabungan.

```
select query, step, starttime, endtime, tasknum, rows
from stl_merge
limit 10;
```

```
 query | step |       starttime     |        endtime      | tasknum | rows
-------+------+---------------------+---------------------+---------+------
     9 |    0 | 2013-08-12 20:08:14 | 2013-08-12 20:08:14 |       0 |    0
    12 |    0 | 2013-08-12 20:09:10 | 2013-08-12 20:09:10 |       0 |    0
    15 |    0 | 2013-08-12 20:10:24 | 2013-08-12 20:10:24 |       0 |    0
    20 |    0 | 2013-08-12 20:11:27 | 2013-08-12 20:11:27 |       0 |    0
    26 |    0 | 2013-08-12 20:12:28 | 2013-08-12 20:12:28 |       0 |    0
    32 |    0 | 2013-08-12 20:14:33 | 2013-08-12 20:14:33 |       0 |    0
    38 |    0 | 2013-08-12 20:16:43 | 2013-08-12 20:16:43 |       0 |    0
    44 |    0 | 2013-08-12 20:17:05 | 2013-08-12 20:17:05 |       0 |    0
    50 |    0 | 2013-08-12 20:18:48 | 2013-08-12 20:18:48 |       0 |    0
    56 |    0 | 2013-08-12 20:20:48 | 2013-08-12 20:20:48 |       0 |    0
(10 rows)
```

STL_MERGEJOIN

Analisis menggabungkan langkah-langkah eksekusi gabungan untuk kueri.

STL_MERGEJOIN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_MERGEJOIN hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
tbl	integer	ID tabel. Ini adalah ID untuk tabel dalam yang digunakan dalam gabungan gabungan.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Contoh berikut mengembalikan hasil gabungan gabungan untuk kueri terbaru.

```
select sum(s.qtysold), e.eventname
from event e, listing l, sales s
where e.eventid=l.eventid
and l.listid= s.listid
```

```
group by e.eventname;

select * from stl_mergejoin where query=pg_last_query_id();
```

```
 userid | query | slice | segment | step |         starttime   |          endtime    |
 tasknum | rows | tbl
--------+-------+-------+---------+------+---------------------+---------------------
+---------+------+-----
    100 | 27399 |     3 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |43428 | 240
    100 | 27399 |     0 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |43159 | 240
    100 | 27399 |     2 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |42778 | 240
    100 | 27399 |     1 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |43091 | 240
```

STL_MV_STATE

Tampilan STL_MV_STATE berisi baris untuk setiap transisi status dari tampilan yang terwujud.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

STL_MV_STATE dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_MV_STATUS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	bigint	ID pengguna yang membuat acara.
waktu mulai	timestamp	Waktu mulai acara.

Nama kolom	Jenis data	Deskripsi
xid	bigint	ID transaksi dari acara tersebut.
event_desc	karakter (500)	Peristiwa yang mendorong perubahan negara. Beberapa contoh nilai termasuk yang berikut: • Jenis kolom diubah • Kolom dijatuhkan • Kolom diganti namanya • Nama skema diubah • Konversi tabel kecil • MEMOTONG • Vakum Perhatikan bahwa ada kemungkinan nilai lain untuk kolom ini.
db_name	karakter (128)	Basis data yang berisi tampilan terwujud.
skema tabel_dasar	karakter (128)	Skema tabel dasar.
nama_tabel_dasar	karakter (128)	Nama tabel dasar.
mv_schema	karakter (128)	Skema tampilan yang terwujud.
mv_name	karakter (128)	Nama tampilan yang terwujud.
status	karakter (32)	Status perubahan tampilan yang terwujud sebagai berikut: • Menghitung ulang • Tidak bisa disegarkan

Tabel berikut menunjukkan contoh kombinasi `event_desc` dan`state`.

```
         event_desc     |     state
------------------------+---------------
 TRUNCATE               | Recompute
 TRUNCATE               | Recompute
 Small table conversion  | Recompute
 Vacuum                 | Recompute
 Column was renamed     | Unrefreshable
 Column was dropped     | Unrefreshable
 Table was renamed      | Unrefreshable
 Column type was changed | Unrefreshable
 Schema name was changed | Unrefreshable
```

Contoh kueri

Untuk melihat log transisi status dari tampilan yang terwujud, jalankan kueri berikut.

```
select * from stl_mv_state;
```

Kueri ini mengembalikan output sampel berikut:

```
 userid |         starttime          | xid  |            event_desc       | db_name |
 base_table_schema   |   base_table_name    |      mv_schema       | mv_name       |
  state
--------+----------------------------+------+-----------------------------+---------
+---------------------+---------------------+---------------------+--------------
+---------------
    138 | 2020-02-14 02:21:25.578885 | 5180 | TRUNCATE                    | dev     |
 public              | mv_base_table        | public              | mv_test       |
 Recompute
    138 | 2020-02-14 02:21:56.846774 | 5275 | Column was dropped          | dev     |
                     | mv_base_table        | public              | mv_test       |
 Unrefreshable
    100 | 2020-02-13 22:09:53.041228 | 1794 | Column was renamed          | dev     |
                     | mv_base_table        | public              | mv_test       |
 Unrefreshable
      1 | 2020-02-13 22:10:23.630914 | 1893 | ALTER TABLE ALTER SORTKEY   | dev     |
 public              | mv_base_table_sorted | public              | mv_test       |
 Recompute
      1 | 2020-02-17 22:57:22.497989 | 8455 | ALTER TABLE ALTER DISTSTYLE | dev     |
 public              | mv_base_table        | public              | mv_test       |
 Recompute
```

```
   173 | 2020-02-17 22:57:23.591434 | 8504 | Table was renamed           | dev     |
                    | mv_base_table        | public               | mv_test       |
Unrefreshable
   173 | 2020-02-17 22:57:27.229423 | 8592 | Column type was changed     | dev     |
                    | mv_base_table        | public               | mv_test       |
Unrefreshable
   197 | 2020-02-17 22:59:06.212569 | 9668 | TRUNCATE                    | dev     |
schemaf796e415850f4f | mv_base_table        | schemaf796e415850f4f | mv_test       |
Recompute
   138 | 2020-02-14 02:21:55.705655 | 5226 | Column was renamed          | dev     |
                    | mv_base_table        | public               | mv_test       |
Unrefreshable
     1 | 2020-02-14 02:22:26.292434 | 5325 | ALTER TABLE ALTER SORTKEY   | dev     |
public              | mv_base_table_sorted | public               | mv_test       |
Recompute
```

STL_NESTLOOP

Menganalisis langkah-langkah eksekusi gabungan loop bersarang untuk kueri.

STL_NESTLOOP dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_NESTLOOP hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
tbl	integer	ID tabel.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Karena kueri berikut mengabaikan untuk bergabung dengan tabel CATEGORY, ia menghasilkan produk Cartesian parsional, yang tidak disarankan. Hal ini ditampilkan di sini untuk mengilustrasikan loop bersarang.

```
select count(event.eventname), event.eventname, category.catname, date.caldate
from event, category, date
where event.dateid = date.dateid
group by event.eventname, category.catname, date.caldate;
```

Query berikut menunjukkan hasil dari query sebelumnya dalam tampilan STL_NESTLOOP.

```
select query, slice, segment as seg, step,
datediff(msec, starttime, endtime) as duration, tasknum, rows, tbl
from stl_nestloop
where query = pg_last_query_id();
```

```
 query | slice | seg | step | duration | tasknum | rows  | tbl
-------+-------+-----+------+----------+---------+-------+-----
  6028 |     0 |   4 |    5 |       41 |      22 | 24277 | 240
  6028 |     1 |   4 |    5 |       26 |      23 | 24189 | 240
  6028 |     3 |   4 |    5 |       25 |      23 | 24376 | 240
  6028 |     2 |   4 |    5 |       54 |      22 | 23936 | 240
```

STL_PARSE

Menganalisis langkah-langkah kueri yang mengurai string menjadi nilai biner untuk memuat.

STL_PARSE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_PARSE hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan

tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.

Kueri Sampel

Contoh berikut mengembalikan semua hasil langkah kueri untuk irisan 1 dan segmen 0 di mana string diurai menjadi nilai biner.

```
select query, step, starttime, endtime, tasknum, rows
from stl_parse
where slice=1 and segment=0;
```

```
 query | step |     starttime       |        endtime      | tasknum |  rows
-------+------+---------------------+---------------------+---------+--------
   669 |    1 | 2013-08-12 22:35:13 | 2013-08-12 22:35:17 |      32 | 192497
   696 |    1 | 2013-08-12 22:35:49 | 2013-08-12 22:35:49 |      32 |      0
   525 |    1 | 2013-08-12 22:32:03 | 2013-08-12 22:32:03 |      13 |  49990
   585 |    1 | 2013-08-12 22:33:18 | 2013-08-12 22:33:19 |      13 |    202
   621 |    1 | 2013-08-12 22:34:03 | 2013-08-12 22:34:03 |      27 |    365
   651 |    1 | 2013-08-12 22:34:47 | 2013-08-12 22:34:53 |      35 | 192497
   590 |    1 | 2013-08-12 22:33:28 | 2013-08-12 22:33:28 |      19 |      0
   599 |    1 | 2013-08-12 22:33:39 | 2013-08-12 22:33:39 |      31 |     11
   675 |    1 | 2013-08-12 22:35:26 | 2013-08-12 22:35:27 |      38 |   3766
   567 |    1 | 2013-08-12 22:32:47 | 2013-08-12 22:32:48 |      23 |  49990
   630 |    1 | 2013-08-12 22:34:17 | 2013-08-12 22:34:17 |      36 |      0
   572 |    1 | 2013-08-12 22:33:04 | 2013-08-12 22:33:04 |      29 |      0
   645 |    1 | 2013-08-12 22:34:37 | 2013-08-12 22:34:38 |      29 |   8798
   604 |    1 | 2013-08-12 22:33:47 | 2013-08-12 22:33:47 |      37 |      0
(14 rows)
```

STL_PLAN_INFO

Gunakan tampilan STL_PLAN_INFO untuk melihat output EXPLAIN untuk kueri dalam hal sekumpulan baris. Ini adalah cara alternatif untuk melihat rencana kueri.

STL_PLAN_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_PLAN_INFO hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
nodeid	integer	Pengidentifikasi simpul rencana, di mana node memetakan ke satu atau lebih langkah dalam pelaksanaan kueri.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Nomor yang mengidentifikasi langkah kueri.
lokus	integer	Lokasi di mana langkah berjalan. 0 jika pada node komputasi dan 1 jika pada node pemimpin.
plannode	integer	Nilai yang disebutkan dari simpul rencana. Lihat tabel berikut untuk enum untuk plannode. (Kolom PLANNODE di STL_JELASKAN berisi teks simpul rencana.)
biaya awal	double precision	Perkiraan biaya relatif pengembalian baris pertama untuk langkah ini.
total biaya	double precision	Perkiraan biaya relatif untuk melaksanakan langkah.
baris	bigint	Perkiraan jumlah baris yang akan diproduksi secara bertahap.
byte	bigint	Perkiraan jumlah byte yang akan dihasilkan oleh langkah.

Kueri Sampel

Contoh berikut membandingkan rencana kueri untuk kueri SELECT sederhana yang dikembalikan dengan menggunakan perintah EXPLAIN dan dengan menanyakan tampilan STL_PLAN_INFO.

```
explain select * from category;
QUERY PLAN
```

```
------------------------------------------------------------
XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
(1 row)

select * from category;
catid | catgroup | catname | catdesc
-------+----------+-----------+--------------------------------------------
1 | Sports | MLB | Major League Baseball
3 | Sports | NFL | National Football League
5 | Sports | MLS | Major League Soccer
...

select * from stl_plan_info where query=256;

query | nodeid | segment | step | locus | plannode | startupcost | totalcost
| rows | bytes
-------+--------+---------+------+-------+----------+-------------+-----------+------
+-------
256 | 1 | 0 | 1 | 0 | 104 | 0 | 0.11 | 11 | 539
256 | 1 | 0 | 0 | 0 | 104 | 0 | 0.11 | 11 | 539
(2 rows)
```

Dalam contoh ini, PLANNODE 104 mengacu pada pemindaian berurutan dari tabel CATEGORY.

```
select distinct eventname from event order by 1;

eventname
----------------------------------------------------------------------------
.38 Special
3 Doors Down
70s Soul Jam
A Bronx Tale
...

explain select distinct eventname from event order by 1;

QUERY PLAN
------------------------------------------------------------------------------------------
XN Merge (cost=1000000000136.38..1000000000137.82 rows=576 width=17)
Merge Key: eventname
-> XN Network (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Send to leader
```

```
-> XN Sort (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Sort Key: eventname
-> XN Unique (cost=0.00..109.98 rows=576 width=17)
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=17)
(8 rows)

select * from stl_plan_info where query=240 order by nodeid desc;

query | nodeid | segment | step | locus | plannode | startupcost |
totalcost | rows | bytes
-------+--------+---------+------+-------+----------+-----------------
+------------------+------+--------
240 | 5 | 0 | 0 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 5 | 0 | 1 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 4 | 0 | 2 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 0 | 3 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 0 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 1 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 3 | 1 | 2 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 3 | 2 | 0 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 2 | 2 | 1 | 0 | 123 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 1 | 3 | 0 | 0 | 122 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
(10 rows)
```

STL_PROJECT

Berisi baris untuk langkah-langkah kueri yang digunakan untuk mengevaluasi ekspresi.

STL_PROJECT terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_PROJECT hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan

tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Contoh berikut mengembalikan semua baris untuk langkah-langkah kueri yang digunakan untuk mengevaluasi ekspresi untuk irisan 0 dan segmen 1.

```
select query, step, starttime, endtime, tasknum, rows
from stl_project
where slice=0 and segment=1;
```

```
 query  | step |         starttime   |          endtime    | tasknum | rows
--------+------+---------------------+---------------------+---------+------
  86399 |    2 | 2013-08-29 22:01:21 | 2013-08-29 22:01:21 |      25 |   -1
  86399 |    3 | 2013-08-29 22:01:21 | 2013-08-29 22:01:21 |      25 |   -1
    719 |    1 | 2013-08-12 22:38:33 | 2013-08-12 22:38:33 |       7 |   -1
  86383 |    1 | 2013-08-29 21:58:35 | 2013-08-29 21:58:35 |       7 |   -1
    714 |    1 | 2013-08-12 22:38:17 | 2013-08-12 22:38:17 |       2 |   -1
  86375 |    1 | 2013-08-29 21:57:59 | 2013-08-29 21:57:59 |       2 |   -1
  86397 |    2 | 2013-08-29 22:01:20 | 2013-08-29 22:01:20 |      19 |   -1
    627 |    1 | 2013-08-12 22:34:13 | 2013-08-12 22:34:13 |      34 |   -1
  86326 |    2 | 2013-08-29 21:45:28 | 2013-08-29 21:45:28 |      34 |   -1
  86326 |    3 | 2013-08-29 21:45:28 | 2013-08-29 21:45:28 |      34 |   -1
  86325 |    2 | 2013-08-29 21:45:27 | 2013-08-29 21:45:27 |      28 |   -1
  86371 |    1 | 2013-08-29 21:57:42 | 2013-08-29 21:57:42 |       4 |   -1
 111100 |    2 | 2013-09-03 19:04:45 | 2013-09-03 19:04:45 |      12 |   -1
    704 |    2 | 2013-08-12 22:36:34 | 2013-08-12 22:36:34 |      37 |   -1
    649 |    2 | 2013-08-12 22:34:47 | 2013-08-12 22:34:47 |      38 |   -1
    649 |    3 | 2013-08-12 22:34:47 | 2013-08-12 22:34:47 |      38 |   -1
    632 |    2 | 2013-08-12 22:34:22 | 2013-08-12 22:34:22 |      13 |   -1
    705 |    2 | 2013-08-12 22:36:48 | 2013-08-12 22:36:49 |      13 |   -1
    705 |    3 | 2013-08-12 22:36:48 | 2013-08-12 22:36:49 |      13 |   -1
      3 |    1 | 2013-08-12 20:07:40 | 2013-08-12 20:07:40 |       3 |   -1
  86373 |    1 | 2013-08-29 21:57:58 | 2013-08-29 21:57:58 |       3 |   -1
 107976 |    1 | 2013-09-03 04:05:12 | 2013-09-03 04:05:12 |       3 |   -1
  86381 |    1 | 2013-08-29 21:58:35 | 2013-08-29 21:58:35 |       8 |   -1
  86396 |    1 | 2013-08-29 22:01:20 | 2013-08-29 22:01:20 |      15 |   -1
    711 |    1 | 2013-08-12 22:37:10 | 2013-08-12 22:37:10 |      20 |   -1
  86324 |    1 | 2013-08-29 21:45:27 | 2013-08-29 21:45:27 |      24 |   -1
(26 rows)
```

STL_QUERY

Mengembalikan informasi eksekusi tentang query database.

> **Note**
>
> Tampilan STL_QUERY dan STL_QUERYTEXT hanya berisi informasi tentang kueri, bukan perintah utilitas dan DDL lainnya. Untuk daftar dan informasi tentang semua pernyataan yang dijalankan oleh Amazon Redshift, Anda juga dapat menanyakan tampilan STL_DDLTEXT dan STL_UTILITYTEXT. Untuk daftar lengkap semua pernyataan yang dijalankan oleh Amazon Redshift, Anda dapat menanyakan tampilan SVL_STATEMENTTEXT.

STL_QUERY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Perhatikan bahwa ketika panjang teks kueri Anda lebih dari 4000 karakter, STL_QUERY hanya menampilkan data terpotong. Untuk mendapatkan teks kueri lengkap, Anda dapat menggunakan UNION pada teks kueri di seluruh baris.

> **Note**
>
> Untuk memverifikasi apakah transaksi yang berisi kueri yang dieksekusi berhasil dilakukan, Anda perlu melakukan operasi gabungan antara tabel sistem dan `sys_transaction_history` tabel. Contoh:

```
SELECT
    stlq.xid AS transaction_id,
    stlq.query AS query_id,
    TRIM(stlq.querytxt) AS query_text,
    th.status AS transaction_status
FROM
    stl_query stlq
LEFT JOIN
    sys_transaction_history th ON stlq.xid = th.transaction_id;
```

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
label	karakter (320)	Baik nama file yang digunakan untuk menjalankan query atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, nilai bidang ini adalah. `default`
xid	bigint	ID Transaksi.
pid	integer	ID proses. Biasanya, semua kueri dalam sesi dijalanka n dalam proses yang sama, jadi nilai ini biasanya tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama. Setelah kejadian internal tertentu, Amazon Redshift mungkin memulai ulang sesi aktif dan menetapkan PID baru. Untuk informasi selengkapnya, lihat STL_RESTARTED_SESSIONS.
basis data	karakter (32)	Nama database yang terhubung ke pengguna saat kueri dikeluarkan.
QueryText	karakter (4000)	Teks kueri aktual untuk kueri.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi

Nama kolom	Jenis data	Deskripsi
		untuk detik pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
dibatalkan	integer	Jika kueri dihentikan oleh sistem atau dibatalkan oleh pengguna, kolom ini berisi**1**. Jika kueri berjalan hingga selesai (termasuk mengembalikan hasil ke klien), kolom ini berisi**0**. Jika klien terputus sebelum menerima hasil, kueri akan ditandai sebagai (**1**), meskipun berhasil diselesaikan di backend.
insert_pristine	integer	Apakah kueri tulis are/were dapat dijalankan saat kueri saat ini is/was berjalan. 1 = tidak ada kueri penulisan yang diizinkan. 0 = kueri tulis diperbolehkan. Kolom ini dimaksudkan untuk digunakan dalam debugging.
status penskalaan_konkurensi	integer	Menunjukkan apakah kueri berjalan di cluster utama atau pada cluster penskalaan konkurensi. Kemungkinan nilainya adalah sebagai berikut: 0 - Berjalan di cluster utama 1 - Berjalan pada cluster penskalaan konkurensi Lebih besar dari 1 - Berjalan di cluster utama

Kueri Sampel

Kueri berikut mencantumkan lima kueri terbaru.

```
select query, trim(querytxt) as sqlquery
from stl_query
order by query desc limit 5;

query |                                  sqlquery
------+--------------------------------------------------
129 | select query, trim(querytxt) from stl_query order by query;
128 | select node from stv_disk_read_speeds;
```

```
127 | select system_status from stv_gui_status
126 | select * from systable_topology order by slice
125 | load global dict registry
(5 rows)
```

Kueri berikut mengembalikan waktu yang telah berlalu dalam urutan menurun untuk kueri yang berjalan pada 15 Februari 2013.

```
select query, datediff(seconds, starttime, endtime),
trim(querytxt) as sqlquery
from stl_query
where starttime >= '2013-02-15 00:00' and endtime < '2013-02-16 00:00'
order by date_diff desc;

 query | date_diff |  sqlquery
-------+-----------+-----------------------------------------
 55    |       119 | padb_fetch_sample: select count(*) from category
121    |         9 | select * from svl_query_summary;
181    |         6 | select * from svl_query_summary where query in(179,178);
172    |         5 | select * from svl_query_summary where query=148;
...
(189 rows)
```

Kueri berikut menunjukkan waktu antrian dan waktu eksekusi untuk kueri. Kueri yang `concurrency_scaling_status = 1` dijalankan pada cluster penskalaan konkurensi. Semua kueri lain berjalan di cluster utama.

```
SELECT w.service_class AS queue
     , q.concurrency_scaling_status
     , COUNT( * ) AS queries
     , SUM( q.aborted )  AS aborted
     , SUM( ROUND( total_queue_time::NUMERIC / 1000000,2 ) ) AS queue_secs
     , SUM( ROUND( total_exec_time::NUMERIC / 1000000,2 ) )  AS exec_secs
FROM stl_query q
     JOIN stl_wlm_query w
          USING (userid,query)
WHERE q.userid > 1
  AND service_class > 5
  AND q.starttime > '2019-03-01 16:38:00'
  AND q.endtime   < '2019-03-01 17:40:00'
GROUP BY 1,2
ORDER BY 1,2;
```

STL_QUERY_METRICS

Berisi informasi metrik, seperti jumlah baris yang diproses, penggunaan CPU, dan penggunaan disk input/output, untuk kueri yang telah selesai berjalan dalam antrian kueri yang ditentukan pengguna (kelas layanan). Untuk melihat metrik untuk kueri aktif yang sedang berjalan, lihat STV_QUERY_METRICS tampilan sistem.

Metrik kueri diambil sampelnya pada interval satu detik. Akibatnya, proses yang berbeda dari kueri yang sama mungkin mengembalikan waktu yang sedikit berbeda. Selain itu, segmen kueri yang berjalan dalam waktu kurang dari satu detik mungkin tidak direkam.

STL_QUERY_METRICS melacak dan menggabungkan metrik pada tingkat kueri, segmen, dan langkah. Untuk informasi tentang segmen kueri dan langkah-langkah, lihatPerencanaan kueri dan alur kerja pelaksanaan. Banyak metrik (sepertimax_rows,cpu_time, dan seterusnya) dijumlahkan di seluruh irisan node. Untuk informasi selengkapnya tentang irisan simpul, lihatArsitektur sistem gudang data.

Untuk menentukan tingkat di mana baris melaporkan metrik, periksa `step_type` kolom `segment` dan.

- Jika keduanya `segment` dan `step_type` ada`-1`, maka baris melaporkan metrik di tingkat kueri.
- Jika `segment` tidak `-1` dan `step_type` sedang`-1`, maka baris melaporkan metrik di tingkat segmen.
- Jika `step_type` keduanya `segment` dan tidak`-1`, maka baris melaporkan metrik di tingkat langkah.

T SVL_QUERY_METRICS ampilan dan SVL_QUERY_METRICS_SUMMARY tampilan menggabungkan data dalam tampilan ini dan menyajikan informasi dalam bentuk yang lebih mudah diakses.

STL_QUERY_METRICS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_DETAIL. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang menjalankan kueri yang menghasilkan entri.
Kelas layanan	integer	ID untuk kelas layanan. Antrian kueri didefinisikan dalam konfigurasi WLM. Metrik dilaporkan hanya untuk antrian yang ditentukan pengguna.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen query dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses. Jika nilai segmen adalah -1, nilai segmen metrik digulung ke tingkat kueri.
tipe_langkah	integer	Jenis langkah yang berjalan. Untuk deskripsi jenis langkah, lihatJenis langkah.
waktu mulai	timestamp	Waktu dalam UTC yang kueri mulai dijalankan, dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: `2009-06-12 11:29:19.131358` .
irisan	integer	Jumlah irisan untuk cluster.
max_baris	bigint	Jumlah maksimum output baris untuk satu langkah, digabungkan di semua irisan.
baris	bigint	Jumlah baris yang diproses dengan satu langkah.
maks_cpu_ waktu_	bigint	Waktu CPU maksimum yang digunakan, dalam mikrodetik. Pada tingkat segmen, waktu CPU maksimum yang digunakan oleh segmen di semua irisan. Pada tingkat kueri, waktu CPU maksimum yang digunakan oleh segmen kueri apa pun.

Nama kolom	Jenis data	Deskripsi
CPU_waktu	bigint	Waktu CPU yang digunakan, dalam mikrodetik. Pada tingkat segmen, total waktu CPU untuk segmen di semua irisan. Pada tingkat kueri, jumlah waktu CPU untuk kueri di semua irisan dan segmen.
max_blocks_read	integer	Jumlah maksimum blok 1 MB yang dibaca oleh segmen, digabungkan di semua irisan. Pada tingkat segmen, jumlah maksimum blok 1 MB yang dibaca untuk segmen di semua irisan. Pada tingkat kueri, jumlah maksimum blok 1 MB dibaca oleh segmen kueri apa pun.
blokir_baca	bigint	Jumlah blok 1 MB yang dibaca oleh kueri atau segmen.
waktu_lari maksimal	bigint	Waktu maksimum yang telah berlalu untuk segmen, dalam mikrodetik. Pada tingkat segmen, waktu proses maksimum untuk segmen di semua irisan. Pada tingkat kueri, waktu berjalan maksimum untuk segmen kueri apa pun.
waktu_lari	bigint	Total waktu berjalan, dijumlahkan di seluruh irisan. Waktu berjalan tidak termasuk waktu tunggu. Pada tingkat segmen, waktu berjalan untuk segmen, dijumlahkan di semua irisan. Pada tingkat kueri, waktu berjalan untuk kueri dijumlahkan di semua irisan dan segmen. Karena nilai ini adalah jumlah, waktu berjalan tidak terkait dengan waktu eksekusi kueri.
max_blocks_to_disk	bigint	Jumlah maksimum ruang disk yang digunakan untuk menulis hasil perantara, dalam blok MB. Pada tingkat segmen, jumlah maksimum ruang disk yang digunakan oleh segmen di semua irisan. Pada tingkat kueri, jumlah maksimum ruang disk yang digunakan oleh segmen kueri apa pun.
memblokir_ke_disk	bigint	Jumlah ruang disk yang digunakan oleh kueri atau segmen untuk menulis hasil perantara, dalam blok MB.
langkah	integer	Langkah kueri yang berjalan.

Nama kolom	Jenis data	Deskripsi
maksimum_kueri_scan_ukuran	bigint	Ukuran maksimum data yang dipindai oleh kueri, dalam MB. Pada tingkat segmen, ukuran maksimum data yang dipindai oleh segmen di semua irisan. Pada tingkat kueri, ukuran maksimum data yang dipindai oleh segmen kueri apa pun.
kueri_scan_ukuran	bigint	Ukuran data yang dipindai oleh kueri, dalam MB.
Query_Prioritas	integer	Prioritas kueri. Nilai yang mungkin adalah -10,1,2,3,, dan4, yang -1 berarti bahwa prioritas kueri tidak didukung.
Query_Queue_Time	bigint	Jumlah waktu dalam mikrodetik saat kueri diantri.
nama_kelas_layanan	karakter (64)	Nama kelas layanan.

Contoh kueri

Untuk menemukan kueri dengan waktu CPU tinggi (lebih dari 1.000 detik), jalankan kueri berikut.

```
Select query, cpu_time / 1000000 as cpu_seconds
from stl_query_metrics where segment = -1 and cpu_time > 1000000000
order by cpu_time;

query | cpu_seconds
------+------------
25775 |        9540
```

Untuk menemukan kueri aktif dengan penggabungan loop bersarang yang mengembalikan lebih dari satu juta baris, jalankan kueri berikut.

```
select query, rows
from stl_query_metrics
where step_type = 15 and rows > 1000000
order by rows;

query | rows
```

```
------+-----------
25775 | 2621562702
```

Untuk menemukan kueri aktif yang telah berjalan selama lebih dari 60 detik dan telah menggunakan kurang dari 10 detik waktu CPU, jalankan kueri berikut.

```
select query, run_time/1000000 as run_time_seconds
from stl_query_metrics
where segment = -1 and run_time > 60000000 and cpu_time < 10000000;

query | run_time_seconds
------+-----------------
25775 |              114
```

STL_QUERYTEXT

Menangkap teks kueri untuk perintah SQL.

Kueri tampilan STL_QUERYTEXT untuk menangkap SQL yang dicatat untuk pernyataan berikut:

- PILIH, PILIH KE
- MASUKKAN, PERBARUI, HAPUS
- MENYALIN
- MEMBONGKAR
- Kueri yang dihasilkan dengan menjalankan VACUUM dan ANALYZE
- BUAT TABEL SEBAGAI (CTAS)

Untuk menanyakan aktivitas untuk pernyataan ini selama periode waktu tertentu, gabungkan tampilan STL_QUERYTEXT dan STL_QUERY.

> **Note**
>
> Tampilan STL_QUERY dan STL_QUERYTEXT hanya berisi informasi tentang kueri, bukan perintah utilitas dan DDL lainnya. Untuk daftar dan informasi tentang semua pernyataan yang dijalankan oleh Amazon Redshift, Anda juga dapat menanyakan tampilan STL_DDLTEXT dan STL_UTILITYTEXT. Untuk daftar lengkap semua pernyataan yang dijalankan oleh Amazon Redshift, Anda dapat menanyakan tampilan SVL_STATEMENTTEXT.

Lihat jugaSTL_DDLTEXT,STL_UTILITYTEXT, danSVL_STATEMENTTEXT.

STL_QUERYTEXT terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_TEKS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Perhatikan bahwa ketika panjang teks kueri Anda lebih dari 4000 karakter, STL_QUERYTEXT hanya menampilkan data terpotong. Untuk mendapatkan teks kueri lengkap, Anda dapat menggunakan UNION pada teks kueri di seluruh baris.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID Transaksi.
pid	integer	ID proses. Biasanya, semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini biasanya tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama. Setelah kejadian internal tertentu, Amazon Redshift mungkin memulai ulang sesi aktif dan menetapkan PID baru. Untuk informasi selengkapnya, lihat STL_RESTARTED_SESSIONS. Anda dapat menggunakan kolom ini untuk bergabung dengan STL_ERROR tampilan.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
urutan	integer	Ketika pernyataan tunggal berisi lebih dari 200 karakter, baris tambahan dicatat untuk pernyataan itu. Urutan 0

Nama kolom	Jenis data	Deskripsi
		adalah baris pertama, 1 adalah yang kedua, dan seterusnya.
text	karakter (200)	Teks SQL, dalam penambahan 200 karakter. Bidang ini mungkin berisi karakter khusus seperti backslash (\\) dan newline (). \n
user_query_id	bigint	Pengidentifikasi kueri dari kueri yang dikirimkan pengguna, seperti yang direkam di kolom query_id dari. SYS_QUERY_TEKS
urutan kueri anak	integer	Nomor urutan kueri anak yang ditulis ulang di bawah kueri pengguna, dimulai dengan 1.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Anda dapat menggunakan fungsi PG_BACKEND_PID () untuk mengambil informasi untuk sesi saat ini. Misalnya, kueri berikut mengembalikan ID kueri dan sebagian teks kueri untuk kueri yang diselesaikan dalam sesi saat ini.

```
select query, substring(text,1,60)
from stl_querytext
where pid = pg_backend_pid()
order by query desc;

 query |                          substring
-------+-----------------------------------------------------------
 28262 | select query, substring(text,1,80) from stl_querytext where
 28252 | select query, substring(path,0,80) as path from stl_unload_l
 28248 | copy category from 's3://dw-tickit/manifest/category/1030_ma
 28247 | Count rows in target table
 28245 | unload ('select * from category') to 's3://dw-tickit/manifes
 28240 | select query, substring(text,1,40) from stl_querytext where
(6 rows)
```

Merekonstruksi SQL yang disimpan

Untuk merekonstruksi SQL yang disimpan di `text` kolom STL_QUERYTEXT, jalankan pernyataan SELECT untuk membuat SQL dari 1 atau lebih bagian di kolom. `text` Sebelum menjalankan SQL yang direkonstruksi, ganti karakter khusus (\n) dengan baris baru. Hasil dari pernyataan SELECT berikut adalah baris SQL yang direkonstruksi di `query_statement` lapangan.

```
SELECT query, LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END)
 WITHIN GROUP (ORDER BY sequence) as query_statement, COUNT(*) as row_count
FROM stl_querytext GROUP BY query ORDER BY query desc;
```

Misalnya, kueri berikut memilih 3 kolom. Kueri itu sendiri lebih panjang dari 200 karakter dan disimpan dalam beberapa bagian di STL_QUERYTEXT.

```
select
1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,
3 AS b01234567890123456789012345678901234567890123456789012345678901234
FROM stl_querytext;
```

Dalam contoh ini, query disimpan dalam 2 bagian (baris) di `text` kolom STL_QUERYTEXT.

```
select query, sequence, text
from stl_querytext where query=pg_last_query_id() order by query desc, sequence limit
 10;
```

```
query  | sequence |
                        text

-------+----------
+----------------------------------------------------------------------------------------------------------------
    45 |        0 | select\n1 AS
 a01234567890123456789012345678901234567890123456789012345678901234567890,\n2 AS
 b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b01234567890123456789012345678901234567890123456789012345678901234
    45 |        1 | \nFROM stl_querytext;
```

Untuk merekonstruksi SQL yang disimpan di STL_QUERYTEXT, jalankan SQL berikut.

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS text
from stl_querytext where query=pg_last_query_id();
```

Untuk menggunakan SQL yang direkonstruksi yang dihasilkan di klien Anda, ganti karakter khusus (\n) dengan baris baru.

```
                    text

----------------------------------------------------------------------------------------------------
 select\n1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
\n2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b0123456789012345678901234567890123456789012345678901234\nFROM stl_querytext;
```

STL_PENGGANTIAN

Menampilkan log yang mencatat ketika UTF-8 karakter tidak valid diganti oleh MENYALIN perintah dengan opsi ACCEPTINVCHARS. Entri log ditambahkan ke STL_REPLACEMENTS untuk masing-masing dari 100 baris pertama pada setiap irisan node yang memerlukan setidaknya satu penggantian.

STL_REPLACEMENTS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_NESTLOOP hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_COPY_PENGGANTIAN Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor irisan simpul tempat penggantian terjadi.
tbl	integer	ID tabel.
waktu mulai	timestamp	Waktu mulai dalam UTC untuk perintah COPY.
sesi	integer	ID sesi untuk sesi yang melakukan perintah COPY.
nama berkas	karakter (256)	Lengkapi jalur ke file input untuk perintah COPY.
bilang_baris	bigint	Nomor baris dalam file data input yang berisi UTF-8 karakter yang tidak valid. A -1 menunjukkan bahwa nomor baris tidak tersedia, seperti, saat menyalin dari file data kolumnar.
colname	karakter (127)	Bidang pertama yang berisi UTF-8 karakter yang tidak valid.
raw_line	karakter (1024)	Data pemuatan mentah yang berisi UTF-8 karakter yang tidak valid.
user_query_id	bigint	Pengidentifikasi kueri dari kueri yang dikirimkan pengguna, seperti yang direkam di kolom query_id dari. SYS_COPY_PENGGANTIAN

Kueri Sampel

Contoh berikut mengembalikan pengganti untuk operasi COPY terbaru.

```
select query, session, filename, line_number, colname
from stl_replacements
```

```
where query = pg_last_copy_id();

 query | session |   filename                                          | line_number | colname
 ------+---------+-----------------------------------------------------+-------------+--------
    96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt   |         251 | city
    96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt   |         317 | city
    96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt   |         569 | city
    96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt   |         623 | city
    96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt   |         694 | city
...
```

STL_RESTARTED_SESSIONS

Untuk menjaga ketersediaan berkelanjutan setelah peristiwa internal tertentu, Amazon Redshift mungkin memulai ulang sesi aktif dengan ID proses (PID) baru. Saat Amazon Redshift memulai ulang sesi, STL_RESTARTED_SESSIONS merekam PID baru dan PID lama.

Untuk informasi lebih lanjut, lihat contoh berikut di bagian ini.

STL_RESTARTED_SESSIONS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_SESSION_RIWAYAT. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu saat ini	timestamp	Waktu acara.
nama db	karakter (50)	Nama database yang terkait dengan sesi.
newpid	integer	ID proses untuk sesi yang dimulai ulang.
oldpid	integer	ID proses untuk sesi asli.

Nama kolom	Jenis data	Deskripsi
nama pengguna	karakter (50)	Nama pengguna yang terkait dengan sesi.
host jarak jauh	karakter (45)	Nama atau alamat IP host jarak jauh.
pelabuhan jarak jauh	karakter (32)	Nomor port host jarak jauh.
waktu parkir	timestamp	Informasi ini hanya untuk penggunaan internal.
session_vars	karakter (2000)	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Contoh berikut bergabung dengan STL_RESTARTED_SESSIONS dengan STL_SESSIONS untuk menampilkan nama pengguna untuk sesi yang telah dimulai ulang.

```
select process, stl_restarted_sessions.newpid, user_name
from stl_sessions
inner join stl_restarted_sessions on stl_sessions.process =
 stl_restarted_sessions.oldpid
order by process;

...
```

STL_RETURN

Berisi detail untuk langkah pengembalian dalam kueri. Langkah pengembalian mengembalikan hasil kueri yang diselesaikan pada node komputasi ke node pemimpin. Node pemimpin kemudian menggabungkan data dan mengembalikan hasilnya ke klien yang meminta. Untuk kueri yang diselesaikan pada node pemimpin, langkah pengembalian mengembalikan hasil ke klien.

Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Untuk informasi selengkapnya, lihat Pemrosesan kueri.

STL_RETURN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_RETURN hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .

Nama kolom	Jenis data	Deskripsi
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
paket	integer	Jumlah total paket yang dikirim melalui jaringan.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Kueri berikut menunjukkan langkah mana dalam kueri terbaru yang dilakukan pada setiap irisan.

```
SELECT query, slice, segment, step, endtime, rows, packets
from stl_return where query = pg_last_query_id();

 query |  slice | segment | step |          endtime           | rows | packets
-------+--------+---------+------+----------------------------+------+---------
     4 |      2 |       3 |    2 | 2013-12-27 01:43:21.469043 |    3 |       0
     4 |      3 |       3 |    2 | 2013-12-27 01:43:21.473321 |    0 |       0
     4 |      0 |       3 |    2 | 2013-12-27 01:43:21.469118 |    2 |       0
     4 |      1 |       3 |    2 | 2013-12-27 01:43:21.474196 |    0 |       0
     4 |      4 |       3 |    2 | 2013-12-27 01:43:21.47704  |    2 |       0
     4 |      5 |       3 |    2 | 2013-12-27 01:43:21.478593 |    0 |       0
     4 |   12811|       4 |    1 | 2013-12-27 01:43:21.480755 |    0 |       0
(7 rows)
```

STL_S3KLIEN

Mencatat waktu transfer dan metrik kinerja lainnya.

Gunakan tabel STL_S3CLIENT untuk menemukan waktu yang dihabiskan untuk mentransfer data dari Amazon S3.

STL_S3CLIENT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
waktu rekor	timestamp	Waktu catatan dicatat.
pid	integer	ID proses. Semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama.
http_metode	karakter (64)	Nama metode HTTP sesuai dengan permintaan Amazon S3.
bucket	karakter (64)	Nama bucket S3.
kunci	karakter (256)	Kunci yang sesuai dengan objek Amazon S3.
ukuran transfer	bigint	Jumlah byte yang ditransfer.
ukuran data	bigint	Jumlah byte data. Nilai ini sama dengan transfer_size untuk data yang tidak terkompresi. Jika kompresi digunakan, ini adalah ukuran data yang tidak terkompresi.
waktu mulai	bigint	Waktu ketika transfer dimulai (dalam mikrodetik sejak 1 Januari 2000).

Nama kolom	Jenis data	Deskripsi
akhir_waktu	bigint	Waktu transfer berakhir (dalam mikrodetik sejak 1 Januari 2000).
waktu transfer	bigint	Waktu yang dibutuhkan oleh transfer (dalam mikrodetik).
waktu_kompresi	bigint	Bagian dari waktu transfer yang dihabiskan untuk mengompresi data (dalam mikrodetik).
sambungan_waktu	bigint	Waktu dari awal hingga koneksi ke server jarak jauh selesai (dalam mikrodetik).
app_connect_time	bigint	Waktu dari awal hingga SSL connect/handshake dengan host jarak jauh selesai (dalam mikrodetik).
mencoba lagi	bigint	Berapa kali transfer dicoba lagi.
id permintaan	karakter (32)	Minta ID dari header respons HTTP Amazon S3
id permintaan_diperpanjang	karakter (128)	ID permintaan diperpanjang dari respons header HTTP Amazon S3 (x-amz-id-2).
alamat ip	karakter (64)	Alamat IP server (ip V4 atau V6).
adalah_sebagian	integer	Nilai yang jika benar (1) menunjukkan file input dibagi menjadi rentang selama operasi COPY. Jika nilai ini false (0), file input tidak dibagi.
awal_offset	bigint	Nilai yang, jika file input dibagi selama operasi COPY, menunjukkan nilai offset pemisahan (dalam byte). Jika file tidak dibagi, nilai ini adalah 0.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Contoh kueri

Kueri berikut mengembalikan waktu yang dibutuhkan untuk memuat file menggunakan perintah COPY.

```
select slice, key, transfer_time
from stl_s3client
where query = pg_last_copy_id();
```

Hasil

```
 slice |   key                         | transfer_time
 ------+-------------------------------+---------------
     0 | listing10M0003_part_00        |    16626716
     1 | listing10M0001_part_00        |    12894494
     2 | listing10M0002_part_00        |    14320978
     3 | listing10M0000_part_00        |    11293439
  3371 | prefix=listing10M;marker=     |       99395
```

Kueri berikut mengonversi `start_time` dan `end_time` ke stempel waktu.

```
select userid,query,slice,pid,recordtime,start_time,end_time,
'2000-01-01'::timestamp + (start_time/1000000.0)* interval '1 second' as start_ts,
'2000-01-01'::timestamp + (end_time/1000000.0)* interval '1 second' as end_ts
from stl_s3client where query> -1 limit 5;
```

```
 userid | query | slice |  pid  |         recordtime         |   start_time    |
 end_time     |          start_ts          |           end_ts
--------+-------+-------+-------+----------------------------+-----------------
+-----------------+----------------------------+----------------------------
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.207839 | 616436837154256 |
 616436837207838 | 2019-07-14 16:27:17.154256 | 2019-07-14 16:27:17.207838
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.252521 | 616436837208208 |
 616436837252520 | 2019-07-14 16:27:17.208208 | 2019-07-14 16:27:17.25252
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.284376 | 616436837208460 |
 616436837284374 | 2019-07-14 16:27:17.20846  | 2019-07-14 16:27:17.284374
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.285307 | 616436837208980 |
 616436837285306 | 2019-07-14 16:27:17.20898  | 2019-07-14 16:27:17.285306
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.353853 | 616436837302216 |
 616436837353851 | 2019-07-14 16:27:17.302216 | 2019-07-14 16:27:17.353851
```

STL_S3CLIENT_ERROR

Merekam kesalahan yang ditemui oleh irisan saat memuat file dari Amazon S3.

Gunakan STL_S3CLIENT_ERROR untuk menemukan detail kesalahan yang ditemui saat mentransfer data dari Amazon S3 sebagai bagian dari perintah COPY.

STL_S3CLIENT_ERROR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya. ID kueri -1 adalah untuk penggunaan internal.
Sliceid	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
waktu rekor	timestamp	Waktu catatan dicatat.
pid	integer	ID proses. Semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama.
http_metode	karakter (64)	Nama metode HTTP sesuai dengan permintaan Amazon S3.
bucket	karakter (64)	Nama bucket Amazon S3.
kunci	karakter (256)	Kunci yang sesuai dengan objek Amazon S3.

Nama kolom	Jenis data	Deskripsi
kesalahan	karakter (1024)	Pesan kesalahan.
adalah_sebagian	integer	Nilai yang, jika benar (1), menunjukkan file input dibagi menjadi rentang selama operasi COPY. Jika nilai ini false (0), file input tidak dibagi.
awal_offset	bigint	Nilai yang, jika file input dibagi selama operasi COPY, menunjukkan nilai offset pemisahan (dalam byte). Jika file tidak dibagi, nilai ini adalah 0.

Catatan penggunaan

Jika Anda melihat beberapa kesalahan dengan “Koneksi habis waktu”, Anda mungkin mengalami masalah jaringan. Jika Anda menggunakan Perutean VPC yang Ditingkatkan, verifikasi bahwa Anda memiliki jalur jaringan yang valid antara VPC cluster dan sumber daya data Anda. Untuk informasi selengkapnya, lihat Perut ean VPC yang Ditingkatkan Amazon Redshift.

Contoh kueri

Kueri berikut mengembalikan kesalahan dari perintah COPY selesai selama sesi saat ini.

```
select query, sliceid, substring(key from 1 for 20) as file,
substring(error from 1 for 35) as error
from stl_s3client_error
where pid = pg_backend_pid()
order by query desc;
```

Hasil

```
 query  | sliceid |     file           |              error

--------+---------+--------------------+-----------------------------------
 362228 |      12 | part.tbl.25.159.gz | transfer closed with 1947655 bytes
 362228 |      24 | part.tbl.15.577.gz | transfer closed with 1881910 bytes
 362228 |       7 | part.tbl.22.600.gz | transfer closed with 700143 bytes r
 362228 |      22 | part.tbl.3.34.gz   | transfer closed with 2334528 bytes
```

```
 362228 |      11 | part.tbl.30.274.gz | transfer closed with 699031 bytes r
 362228 |      30 | part.tbl.5.509.gz  | Unknown SSL protocol error in conne
 361999 |      10 | part.tbl.23.305.gz | transfer closed with 698959 bytes r
 361999 |      19 | part.tbl.26.582.gz | transfer closed with 1881458 bytes
 361999 |       4 | part.tbl.15.629.gz | transfer closed with 2275907 bytes
 361999 |      20 | part.tbl.6.456.gz  | transfer closed with 692162 bytes r
(10 rows)
```

STL_SAVE

Berisi detail untuk langkah menyimpan dalam kueri. Langkah simpan menyimpan aliran input ke tabel transien. Tabel transien adalah tabel sementara yang menyimpan hasil perantara selama eksekusi kueri.

Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Untuk informasi selengkapnya, lihat Pemrosesan kueri.

STL_SAVE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_SAVE hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
tbl	integer	ID tabel transien yang terwujud.
berbasis-disk	karakter (1)	Apakah langkah kueri ini dilakukan sebagai operasi berbasis disk: true (`t`) atau false (`f`).
workmem	bigint	Jumlah byte memori kerja yang ditetapkan untuk langkah.

Kueri Sampel

Kueri berikut menunjukkan langkah penyimpanan mana dalam kueri terbaru yang dilakukan pada setiap irisan.

```
select query, slice, segment, step, tasknum, rows,  tbl
from stl_save where query = pg_last_query_id();
```

```
 query | slice | segment | step | tasknum | rows | tbl
-------+-------+---------+------+---------+------+-----
 52236 |     3 |       0 |    2 |      21 |    0 | 239
 52236 |     2 |       0 |    2 |      20 |    0 | 239
 52236 |     2 |       2 |    2 |      20 |    0 | 239
 52236 |     3 |       2 |    2 |      21 |    0 | 239
 52236 |     1 |       0 |    2 |      21 |    0 | 239
 52236 |     0 |       0 |    2 |      20 |    0 | 239
 52236 |     0 |       2 |    2 |      20 |    0 | 239
 52236 |     1 |       2 |    2 |      21 |    0 | 239
(8 rows)
```

STL_SCAN

Menganalisis langkah-langkah pemindaian tabel untuk kueri. Nomor langkah untuk baris dalam tabel ini selalu 0 karena pemindaian adalah langkah pertama dalam segmen.

STL_SCAN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_SCAN hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.
mengambil	bigint	Informasi ini hanya untuk penggunaan internal.
jenis	integer	ID jenis pemindaian. Untuk daftar nilai yang valid, lihat tabel berikut.
tbl	integer	ID tabel.
is_rrscan	karakter (1)	Jika true (t), menunjukkan bahwa pemindaian terbatas rentang digunakan pada langkah tersebut.
is_tertunda_scan	karakter (1)	Informasi ini hanya untuk penggunaan internal.

Nama kolom	Jenis data	Deskripsi
baris pra_filter	bigint	Untuk pemindaian tabel permanen, jumlah total baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna.
baris pra_filter_pengguna	bigint	Untuk pemindaian tabel permanen, jumlah baris yang diproses setelah memfilter baris ditandai untuk dihapus (baris hantu) tetapi sebelum menerapkan filter kueri yang ditentukan pengguna.
Perm_nama_tabel_	karakter (136)	Untuk pemindaian tabel permanen, nama tabel dipindai.
is_rlf_scan	karakter (1)	Jika true (t), menunjukkan bahwa pemfilteran tingkat baris digunakan pada langkah tersebut.
is_rlf_scan_reason	integer	Informasi ini hanya untuk penggunaan internal.
nom_em_blok	integer	Informasi ini hanya untuk penggunaan internal.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.
penyaringan runtime_	karakter (1)	Jika true (t), menunjukkan bahwa filter runtime diterapkan.
memindai_wilayah	integer	Informasi ini hanya untuk penggunaan internal.
nom_sortkey_as_predikat	integer	Informasi ini hanya untuk penggunaan internal.

Nama kolom	Jenis data	Deskripsi
row_fetcher_state	integer	Informasi ini hanya untuk penggunaan internal.
rentangan_pemindaian yang dikonsumsi	bigint	Informasi ini hanya untuk penggunaan internal.
pekerjaan_mencuri_alasan_kerja	bigint	Informasi ini hanya untuk penggunaan internal.
is_divektorisasi_scan	karakter (1)	Informasi ini hanya untuk penggunaan internal.
alasan_pemindaian_divektorisasi	integer	Informasi ini hanya untuk penggunaan internal.
row_fetcher_reason	bigint	Informasi ini hanya untuk penggunaan internal.
topology_signature	bigint	Informasi ini hanya untuk penggunaan internal.
gunakan_tpm_partisi	karakter (1)	Informasi ini hanya untuk penggunaan internal.
is_rrscan_expr	karakter (1)	Informasi ini hanya untuk penggunaan internal.

Nama kolom	Jenis data	Deskripsi
memindai_mega_nilai-nilai	karakter (1)	Informasi ini hanya untuk penggunaan internal. Informasi ini menunjukkan apakah langkah pemindaian yang diberikan telah memindai nilai yang besar. Nilai besar akan disimpan dalam beberapa blok. Ukuran blok adalah 1 MB secara default, nilai besar lebih besar dari 1 MB dalam pengaturan default.

Jenis pemindaian

Ketik ID	Deskripsi
1	Data dari jaringan.
2	Tabel pengguna permanen dalam memori bersama terkompresi.
3	Tabel baris sementara.
21	Muat file dari Amazon S3.
22	Muat tabel dari Amazon DynamoDB.
23	Muat data dari koneksi SSH jarak jauh.
24	Muat data dari cluster jarak jauh (wilayah yang diurutkan). Ini digunakan untuk mengubah ukuran.
25	Muat data dari cluster jarak jauh (wilayah yang tidak disortir). Ini digunakan untuk mengubah ukuran.
28	Baca data dari tampilan deret waktu dengan UNION ALL pada beberapa tabel.
29	Baca data dari tabel eksternal Amazon S3.
30	Baca informasi partisi dari tabel eksternal Amazon S3.
33	Membaca data dari tabel Postgres jarak jauh.

Ketik ID	Deskripsi
36	Membaca data dari tabel MySQL jarak jauh.
37	Membaca data dari aliran Kinesis jarak jauh.

Catatan penggunaan

Idealnya `rows` harus relatif dekat dengan`rows_pre_filter`. Perbedaan besar antara `rows` dan `rows_pre_filter` merupakan indikasi bahwa mesin eksekusi memindai baris yang kemudian dibuang, yang tidak efisien. Perbedaan antara `rows_pre_filter` dan `rows_pre_user_filter` adalah jumlah baris hantu dalam pemindaian. Jalankan VACUUM untuk menghapus baris yang ditandai untuk dihapus. Perbedaan antara `rows` dan `rows_pre_user_filter` adalah jumlah baris yang difilter oleh kueri. Jika banyak baris dibuang oleh filter pengguna, tinjau kolom pengurutan pilihan Anda atau, jika ini karena wilayah besar yang tidak disortir, jalankan penyedot debu.

Kueri Sampel

Contoh berikut menunjukkan bahwa `rows_pre_filter` lebih besar daripada `rows_pre_user_filter` karena tabel telah menghapus baris yang belum disedot (baris hantu).

```
SELECT query, slice, segment,step,rows, rows_pre_filter, rows_pre_user_filter
from stl_scan where query = pg_last_query_id();

 query |  slice | segment | step | rows  | rows_pre_filter | rows_pre_user_filter
-------+--------+---------+------+-------+-----------------+----------------------
 42915 |      0 |       0 |    0 | 43159 |           86318 |                43159
 42915 |      0 |       1 |    0 |     1 |               0 |                    0
 42915 |      1 |       0 |    0 | 43091 |           86182 |                43091
 42915 |      1 |       1 |    0 |     1 |               0 |                    0
 42915 |      2 |       0 |    0 | 42778 |           85556 |                42778
 42915 |      2 |       1 |    0 |     1 |               0 |                    0
 42915 |      3 |       0 |    0 | 43428 |           86856 |                43428
 42915 |      3 |       1 |    0 |     1 |               0 |                    0
 42915 |  10000 |       2 |    0 |     4 |               0 |                    0
(9 rows)
```

PELANGGARAN STL_SCHEMA_KUOTA_

Merekam kejadian, stempel waktu, XID, dan informasi berguna lainnya saat kuota skema terlampaui.

STL_SCHEMA_QUOTA_VIOLATIONS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_SCHEMA_KUOTA_PELANGGARAN. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
pemilik	integer	ID pemilik skema.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
pid	integer	ID proses yang terkait dengan pernyataan.
ID pengguna	integer	ID pengguna yang membuat entri.
Schema_id	integer	Namespace atau ID skema.
schema_name	karakter (128)	Namespace atau nama skema.
kuota	integer	Jumlah ruang disk (dalam MB) yang dapat digunakan skema.
penggunaan disk	integer	Ruang disk (dalam MB) yang saat ini digunakan oleh skema.
disk_penggunaan_pct	double precision	Persentase ruang disk yang saat ini digunakan oleh skema di luar kuota yang dikonfigurasi.

Nama kolom	Jenis data	Deskripsi
timestamp	stempel waktu tanpa zona waktu	Waktu ketika pelanggaran terjadi.

Kueri Sampel

Kueri berikut menunjukkan hasil pelanggaran kuota:

```
SELECT userid, TRIM(SCHEMA_NAME) "schema_name", quota, disk_usage, disk_usage_pct,
 timestamp FROM
stl_schema_quota_violations WHERE SCHEMA_NAME = 'sales_schema' ORDER BY timestamp DESC;
```

Kueri ini mengembalikan output sampel berikut untuk skema yang ditentukan:

```
userid | schema_name  | quota | disk_usage | disk_usage_pct |timestamp
-------+--------------+-------+------------+----------------
+---------------------------
104    | sales_schema | 2048  | 2798       |  136.62        | 2020-04-20
 20:09:25.494723
(1 row)
```

STL_SESSIONS

Mengembalikan informasi tentang riwayat sesi pengguna.

STL_SESSIONS berbeda dari STV_SESSIONS karena STL_SESSIONS berisi riwayat sesi, di mana STV_SESSIONS berisi sesi aktif saat ini.

STL_SESSIONS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_SESSION_RIWAYAT. Data dalam tampilan pemantauan SYS diformat agar lebih mudah

digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
waktu mulai	timestamp	Waktu dalam UTC saat sesi dimulai.
waktu akhir	timestamp	Waktu dalam UTC sesi berakhir.
proses	integer	ID proses untuk sesi.
user_name	karakter (50)	Nama pengguna yang terkait dengan sesi.
db_name	karakter (50)	Nama database yang terkait dengan sesi.
batas waktu_detik	int	Waktu maksimum dalam detik sesi tetap tidak aktif atau idle sebelum waktu habis. 0 menunjukkan bahwa tidak ada batas waktu yang ditetapkan.
habis waktu	int	Nilai yang menunjukkan mengapa koneksi dihentikan. Ini dapat memiliki nilai-nilai berikut: • 0: Koneksi dihentikan karena kesalahan yang tidak diketahui. • 1: Waktu koneksi habis. • 2: Sisi klien menghentikan koneksi. • 3: Kesalahan internal backend Amazon Redshift menghentikan koneksi.

Kueri Sampel

Untuk melihat riwayat sesi untuk database TICKIT, ketik kueri berikut:

```
select starttime, process, user_name, timeout_sec, timed_out
from stl_sessions
where db_name='tickit' order by starttime;
```

Kueri ini mengembalikan output sampel berikut:

```
    starttime              | process |  user_name              | timeout_sec | timed_out
---------------------------+---------+-------------------------+-------------
+-------------
2008-09-15 09:54:06.746705 |   32358 | dwuser                  | 120         | 1
2008-09-15 09:56:34.30275  |   32744 | dwuser                  | 60          | 1
2008-09-15 11:20:34.694837 |   14906 | dwuser                  | 0           | 0
2008-09-15 11:22:16.749818 |   15148 | dwuser                  | 0           | 0
2008-09-15 14:32:44.66112  |   14031 | dwuser                  | 0           | 0
2008-09-15 14:56:30.22161  |   18380 | dwuser                  | 0           | 0
2008-09-15 15:28:32.509354 |   24344 | dwuser                  | 0           | 0
2008-09-15 16:01:00.557326 |   30153 | dwuser                  | 120         | 1
2008-09-15 17:28:21.419858 |   12805 | dwuser                  | 0           | 0
2008-09-15 20:58:37.601937 |   14951 | dwuser                  | 60          | 1
2008-09-16 11:12:30.960564 |   27437 | dwuser                  | 60          | 1
2008-09-16 14:11:37.639092 |   23790 | dwuser                  | 3600        | 1
2008-09-16 15:13:46.02195  |    1355 | dwuser                  | 120         | 1
2008-09-16 15:22:36.515106 |    2878 | dwuser                  | 120         | 1
2008-09-16 15:44:39.194579 |    6470 | dwuser                  | 120         | 1
2008-09-16 16:50:27.02138  |   17254 | dwuser                  | 120         | 1
2008-09-17 12:05:02.157208 |    8439 | dwuser                  | 3600        | 0
(17 rows)
```

STL_SORT

Menampilkan langkah-langkah eksekusi pengurutan untuk kueri, seperti langkah-langkah yang menggunakan pemrosesan ORDER BY.

STL_SORT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_SORT hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama

tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
byte	bigint	Ukuran, dalam byte, dari semua baris keluaran untuk langkah tersebut.

Nama kolom	Jenis data	Deskripsi
tbl	integer	ID tabel.
berbasis-disk	karakter (1)	Jika true (t), query dilakukan sebagai operasi berbasis disk. Jika false (f), query dilakukan dalam memori.
workmem	bigint	Jumlah total byte dalam memori kerja yang ditetapkan untuk langkah.
checksum	bigint	Informasi ini hanya untuk penggunaan internal.

Kueri Sampel

Contoh berikut mengembalikan hasil pengurutan untuk irisan 0 dan segmen 1.

```
select query, bytes, tbl, is_diskbased, workmem
from stl_sort
where slice=0 and segment=1;
```

```
 query |  bytes  | tbl | is_diskbased |  workmem
-------+---------+-----+--------------+-----------
   567 | 3126968 | 241 | f            | 383385600
   604 |    5292 | 242 | f            | 383385600
   675 |  104776 | 251 | f            | 383385600
   525 | 3126968 | 251 | f            | 383385600
   585 |    5068 | 241 | f            | 383385600
   630 |  204808 | 266 | f            | 383385600
   704 |       0 | 242 | f            |         0
   669 | 4606416 | 241 | f            | 383385600
   696 |  104776 | 241 | f            | 383385600
   651 | 4606416 | 254 | f            | 383385600
   632 |       0 | 256 | f            |         0
   599 |     396 | 241 | f            | 383385600
 86397 |       0 | 242 | f            |         0
   621 |    5292 | 241 | f            | 383385600
 86325 |       0 | 242 | f            |         0
   572 |    5068 | 242 | f            | 383385600
   645 |  204808 | 241 | f            | 383385600
   590 |     396 | 242 | f            | 383385600
```

```
(18 rows)
```

STL_SSHCLIENT_ERROR

Merekam semua kesalahan yang dilihat oleh klien SSH.

STL_SSHCLIENT_ERROR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
waktu rekor	timestamp	Waktu kesalahan dicatat.
pid	integer	Proses yang mencatat kesalahan.
ssh_nama pengguna	karakter (1024)	Nama pengguna SSH.
titik akhir	karakter (1024)	Titik akhir SSH.
perintah	karakter (4096)	Perintah SSH lengkap.
kesalahan	karakter (1024)	Pesan kesalahan.

STL_STREAM_SEGS

Mencantumkan hubungan antara aliran dan segmen bersamaan.

Streaming dalam konteks ini adalah aliran Amazon Redshift. Tampilan sistem ini tidak berkaitan Streaming konsumsi ke tampilan yang terwujud dengan.

STL_STREAM_SEGS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_STREAM_SEGS hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
aliran	integer	Kumpulan segmen bersamaan dari kueri.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Untuk melihat hubungan antara aliran dan segmen bersamaan untuk kueri terbaru, ketik kueri berikut:

```
select *
from stl_stream_segs
where query = pg_last_query_id();

 query | stream | segment
-------+--------+---------
    10 |      1 |       2
    10 |      0 |       0
    10 |      2 |       4
    10 |      1 |       3
    10 |      0 |       1
(5 rows)
```

STL_TR_KONFLIK

Menampilkan informasi untuk mengidentifikasi dan menyelesaikan konflik transaksi dengan tabel database.

Konflik transaksi terjadi ketika dua atau lebih pengguna menanyakan dan memodifikasi baris data dari tabel sehingga transaksi mereka tidak dapat diserialkan. Transaksi yang menjalankan pernyataan yang akan merusak serialisasi dihentikan dan diputar kembali. Setiap kali terjadi konflik transaksi, Amazon Redshift menulis baris data ke tabel sistem STL_TR_CONFLICT yang berisi detail tentang transaksi yang dibatalkan. Untuk informasi selengkapnya, lihat Tingkat isolasi di Amazon Redshift.

STL_TR_CONFLICT hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_RIWAYAT TRANSAKSI_. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
xact_id	bigint	ID transaksi untuk transaksi yang dibatalkan.
Proses_id	bigint	Proses yang terkait dengan transaksi yang digulirkan kembali.
xact_start_ts	timestamp	Timestamp (UTC) saat transaksi dimulai.
waktu abort_	timestamp	Timestamp (UTC) saat transaksi dihentikan.
tabel_id	bigint	ID tabel untuk tabel tempat konflik terjadi.

Contoh kueri

Untuk mengembalikan informasi tentang konflik yang melibatkan tabel tertentu, jalankan kueri yang menentukan ID tabel:

```
select * from stl_tr_conflict where table_id=100234
order by xact_start_ts;

xact_id|process_|      xact_start_ts       |        abort_time        |table_
       |id      |                          |                          |id
-------+--------+--------------------------+--------------------------+------
  1876 |  8551  |2010-03-30 09:19:15.852326|2010-03-30 09:20:17.582499|100234
  1928 | 15034  |2010-03-30 13:20:00.636045|2010-03-30 13:20:47.766817|100234
  1991 | 23753  |2010-04-01 13:05:01.220059|2010-04-01 13:06:06.94098 |100234
  2002 | 23679  |2010-04-01 13:17:05.173473|2010-04-01 13:18:27.898655|100234
(4 rows)
```

Anda bisa mendapatkan ID tabel dari bagian DETAIL pesan kesalahan untuk pelanggaran serialisasi (kesalahan 1023).

STL_UNDONE

Menampilkan informasi tentang transaksi yang telah dibatalkan.

STL_UNDONE dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_RIWAYAT TRANSAKSI_. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xact_id	bigint	ID untuk transaksi undo.
xact_id_dibatalkan	bigint	ID untuk transaksi yang dibatalkan.
undo_start_ts	timestamp	Waktu mulai untuk transaksi undo.
undo_end_ts	timestamp	Waktu akhir untuk transaksi undo.
tabel_id	bigint	ID untuk tabel yang dipengaruhi oleh transaksi undo.

Contoh kueri

Untuk melihat log ringkas dari semua transaksi yang dibatalkan, ketik perintah berikut:

```
select xact_id, xact_id_undone, table_id from stl_undone;
```

Perintah ini mengembalikan output sampel berikut:

```
 xact_id | xact_id_undone | table_id
---------+----------------+----------
```

```
1344 |           1344 |   100192
1326 |           1326 |   100192
1551 |           1551 |   100192
(3 rows)
```

STL_UNIK

Menganalisis langkah-langkah eksekusi yang terjadi ketika fungsi DISTINCT digunakan dalam daftar SELECT atau ketika duplikat dihapus dalam kueri UNION atau INTERSECT.

STL_UNIQUE dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_UNIQUE hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.

Nama kolom	Jenis data	Deskripsi
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
`type`	karakter (6)	Jenis langkah. Nilai yang valid adalah: • DI-HASH. Menunjukkan bahwa langkah yang digunakan dikelompokkan, agregasi tidak disortir. • POLOS. Menunjukkan bahwa langkah yang digunakan agregasi skalar yang tidak dikelompokkan. • DISORTIR. Menunjukkan bahwa langkah yang digunakan dikelompokkan, diurutkan agregasi.
berbasis-disk	karakter (1)	Jika true (t), query dilakukan sebagai operasi berbasis disk. Jika false (f), query dilakukan dalam memori.
slot	integer	Jumlah total bucket hash.
workmem	bigint	Jumlah total byte dalam memori kerja yang ditetapkan untuk langkah.
max_buffers_digunakan	bigint	Jumlah maksimum buffer yang digunakan dalam tabel hash sebelum pergi ke disk.

Nama kolom	Jenis data	Deskripsi
mengubah ukuran	integer	Informasi ini hanya untuk penggunaan internal.
sibuk	integer	Informasi ini hanya untuk penggunaan internal.
dapat dibilas	integer	Informasi ini hanya untuk penggunaan internal.
menggunakan prefetching yang unik	karakter (1)	Informasi ini hanya untuk penggunaan internal.
byte	biginit	Jumlah byte dari semua baris keluaran untuk langkah tersebut.

Kueri Sampel

Misalkan Anda menjalankan query berikut:

```
select distinct eventname
from event order by 1;
```

Dengan asumsi ID untuk kueri sebelumnya adalah 6313, contoh berikut menunjukkan jumlah baris yang dihasilkan oleh langkah unik untuk setiap irisan di segmen 0 dan 1.

```
select query, slice, segment, step, datediff(msec, starttime, endtime) as msec,
 tasknum, rows
from stl_unique where query = 6313
order by query desc, slice, segment, step;
```

```
 query | slice | segment | step | msec | tasknum | rows
-------+-------+---------+------+------+---------+------
  6313 |     0 |       0 |    2 |    0 |      22 |  550
  6313 |     0 |       1 |    1 |  256 |      20 |  145
  6313 |     1 |       0 |    2 |    1 |      23 |  540
```

```
  6313 |     1 |       1 |    1 |   42 |      21 |  127
  6313 |     2 |       0 |    2 |    1 |      22 |  540
  6313 |     2 |       1 |    1 |  255 |      20 |  158
  6313 |     3 |       0 |    2 |    1 |      23 |  542
  6313 |     3 |       1 |    1 |   38 |      21 |  146
(8 rows)
```

STL_UNLOAD_LOG

Merekam detail untuk operasi bongkar muat.

STL_UNLOAD_LOG merekam satu baris untuk setiap file yang dibuat oleh pernyataan UNLOAD. Misalnya, jika UNLOAD membuat 12 file, STL_UNLOAD_LOG akan berisi 12 baris yang sesuai.

STL_UNLOAD_LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_UNLOAD_LOG hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS dan. SYS_UNLOAD_HISTORY SYS_UNLOAD_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
pid	integer	ID proses yang terkait dengan pernyataan query.

Nama kolom	Jenis data	Deskripsi
path	karakter (1280)	Jalur objek Amazon S3 lengkap untuk file.
waktu mulai	timestamp	Waktu mulai untuk transaksi.
akhir_waktu	timestamp	Waktu akhir untuk transaksi.
hitung_garis	bigint	Jumlah baris (baris) yang dibongkar ke file.
ukuran transfer	bigint	Jumlah byte yang ditransfer.
file_format	karakter (10)	Format file yang tidak dimuat.
user_query_id	bigint	Pengidentifikasi kueri dari kueri yang dikirimkan pengguna, seperti yang direkam di kolom query_id dari. SYS_UNLOAD_DETAIL
id transaksi_	bigint	Pengidentifikasi transaksi yang terkait dengan pernyataan.

Contoh kueri

Untuk mendapatkan daftar file yang ditulis ke Amazon S3 dengan perintah UNLOAD, Anda dapat memanggil operasi daftar Amazon S3 setelah UNLOAD selesai. Anda juga dapat menanyakan STL_UNLOAD_LOG.

Kueri berikut mengembalikan pathname untuk file yang dibuat oleh UNLOAD untuk kueri terakhir yang diselesaikan:

```
select query, substring(path,0,40) as path
from stl_unload_log
where query = pg_last_query_id()
order by path;
```

Perintah ini mengembalikan output sampel berikut:

```
 query |             path
-------+-----------------------------------
  2320 | s3://amzn-s3-demo-bucket/venue0000_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0001_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0002_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0003_part_00
(4 rows)
```

STL_PENGGUNAAN_KONTROL

Tampilan STL_USAGE_CONTROL berisi informasi yang dicatat saat batas penggunaan tercapai. Untuk informasi selengkapnya tentang batas penggunaan, lihat Mengelola B atas Penggunaan di Panduan Manajemen Amazon Redshift.

STL_USAGE_CONTROL hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu acara	timestamp	Waktu (UTC) saat kueri melebihi batas penggunaan.
kueri	integer	Pengidentifikasi kueri. Anda dapat menggunakan ID ini untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
xid	bigint	Pengidentifikasi transaksi.
pid	integer	Pengidentifikasi proses yang terkait dengan query.
pemakaian_batas_id	karakter (40)	Pengidentifikasi unik universal (UUID) yang dihasilkan oleh Amazon Redshift, misalnya. `25d9297e-3e7b-41c8-9f4d-c4b6eb731c09`
tipe_fitur	karakter (30)	Fitur yang batas penggunaannya terlampaui. Nilai yang mungkin termasuk `CONCURRENCY_SCALING` dan`SPECTRUM`.

Contoh kueri

Contoh SQL berikut mengembalikan beberapa informasi yang dicatat ketika batas penggunaan tercapai.

```
select query, pid, eventtime, feature_type
from stl_usage_control
order by eventtime desc
limit 5;
```

STL_USERLOG

Merekam detail untuk perubahan berikut pada pengguna database:

- Buat pengguna
- Jatuhkan pengguna
- Ubah pengguna (ganti nama)
- Ubah pengguna (ubah properti)

STL_USERLOG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_USERLOG. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang terpengaruh oleh perubahan.
nama pengguna	karakter (50)	Nama pengguna pengguna yang terpengaruh oleh perubahan.
nama lama	karakter (50)	Untuk tindakan ganti nama, nama pengguna asli. Untuk tindakan lain, bidang ini kosong.

Nama kolom	Jenis data	Deskripsi
tindakan	karakter (10)	Tindakan yang terjadi. Nilai valid: • Mengubah • Buat • Jatuhkan • Ubah Nama
gunakancreatedb	integer	Jika benar (1), menunjukkan bahwa pengguna telah membuat hak istimewa database.
usesuper	integer	Jika benar (1), menunjukkan bahwa pengguna adalah superuser.
menggunakan catupd	integer	Jika benar (1), menunjukkan bahwa pengguna dapat memperbarui katalog sistem.
valuntil	timestamp	Tanggal kedaluwarsa kata sandi.
pid	integer	ID proses.
xid	bigint	ID Transaksi.
waktu rekor	timestamp	Waktu dalam UTC saat kueri dimulai.

Kueri Sampel

Contoh berikut melakukan empat tindakan pengguna, lalu menanyakan tampilan STL_USERLOG.

```
create user userlog1 password 'Userlog1';
alter user userlog1 createdb createuser;
alter user userlog1 rename  to userlog2;
drop user userlog2;

select userid, username, oldusername, action, usecreatedb, usesuper from stl_userlog
 order by recordtime desc;
```

```
 userid |  username | oldusername |  action | usecreatedb | usesuper
--------+-----------+-------------+---------+-------------+----------
    108 | userlog2  |             | drop    |           1 |   1
    108 | userlog2  | userlog1    | rename  |           1 |   1
    108 | userlog1  |             | alter   |           1 |   1
    108 | userlog1  |             | create  |           0 |   0
 (4 rows)
```

STL_UTILITYTEXT

Menangkap teks perintah SQL non-pilih yang dijalankan pada database.

Kueri tampilan STL_UTILITYTEXT untuk menangkap subset pernyataan SQL berikut yang dijalankan pada sistem:

- BATALKAN, MULAI, KOMIT, AKHIRI, KEMBALIKAN
- MENGANALISA
- PANGGILAN
- CANCEL (BATALKAN)
- KOMENTAR
- MEMBUAT, MENGUBAH, MENGHAPUS DATABASE
- BUAT, UBAH, JATUHKAN PENGGUNA
- EXPLAIN
- MEMBERIKAN, MENCABUT
- KUNCI
- MENGATUR ULANG
- SET
- MENUNJUKKAN
- MEMOTONG

Lihat jugaSTL_DDLTEXT,STL_QUERYTEXT, danSVL_STATEMENTTEXT.

Gunakan kolom STARTTIME dan ENDTIME untuk mengetahui pernyataan mana yang dicatat selama periode waktu tertentu. Blok panjang teks SQL dipecah menjadi baris sepanjang 200 karakter; kolom SEQUENCE mengidentifikasi fragmen teks yang termasuk dalam satu pernyataan.

Tabel sistem STL_UTILITYTEXT mendukung operasi MANUAL dan AUTO REFRESH dari tampilan yang terwujud. Untuk mengidentifikasi AUTO REFRESH pada tampilan yang terwujud, temukan kolom`label`. Semua kueri AUTO REFRESH memiliki label dengan nilai`maintenance`.

STL_UTILITYTEXT terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID Transaksi.
pid	integer	ID proses yang terkait dengan pernyataan kueri.
label	karakter (320)	Baik nama file yang digunakan untuk menjalankan query atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, bidang ini kosong.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .

Nama kolom	Jenis data	Deskripsi
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .
urutan	integer	Ketika pernyataan tunggal berisi lebih dari 200 karakter, baris tambahan dicatat untuk pernyataan itu. Urutan 0 adalah baris pertama, 1 adalah yang kedua, dan seterusnya.
text	karakter (200)	Teks SQL, dalam penambahan 200 karakter. Bidang ini mungkin berisi karakter khusus seperti backslash (\\) dan newline (). \n

Kueri Sampel

Kueri berikut mengembalikan teks untuk perintah “utilitas” yang dijalankan pada 26 Januari 2012. Dalam hal ini, beberapa perintah SET dan perintah SHOW ALL dijalankan:

```
select starttime, sequence, rtrim(text)
from stl_utilitytext
where starttime like '2012-01-26%'
order by starttime, sequence;

starttime          | sequence |              rtrim
---------------------------+-----+---------------------------------
2012-01-26 13:05:52.529235 |   0 | show all;
2012-01-26 13:20:31.660255 |   0 | SET query_group to ''
2012-01-26 13:20:54.956131 |   0 | SET query_group to 'soldunsold.sql'
...
```

Merekonstruksi SQL Tersimpan

Untuk merekonstruksi SQL yang disimpan di `text` kolom STL_UTILITYTEXT, jalankan pernyataan SELECT untuk membuat SQL dari 1 atau lebih bagian di kolom. `text` Sebelum menjalankan SQL

yang direkonstruksi, ganti karakter khusus (\n) dengan baris baru. Hasil dari pernyataan SELECT berikut adalah baris SQL yang direkonstruksi di `query_statement` lapangan.

```
SELECT LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END) WITHIN
 GROUP (ORDER BY sequence) as query_statement
FROM stl_utilitytext GROUP BY xid order by xid;
```

Misalnya, kueri berikut menetapkan query_group ke string nol. Kueri itu sendiri lebih panjang dari 200 karakter dan disimpan dalam beberapa bagian di STL_UTILITYTEXT.

```
set query_group to
 '00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             000000';
```

Dalam contoh ini, query disimpan dalam 2 bagian (baris) di `text` kolom STL_UTILITYTEXT.

```
select query, sequence, text
from stl_utilitytext where query=pg_last_query_id() order by query desc, sequence limit
 10;
```

```
         starttime          | sequence |
                                                text

----------------------------+----------
+-----------------------------------------------------------------------------------------------
 2019-07-23 22:55:34.926198 |        0 | set query_group to
 '00000000000000000000000000000000000000000000000000000000000000000000000000000000000000\n00000000000
 2019-07-23 22:55:34.926198 |        1 |                   000000';
```

Untuk merekonstruksi SQL yang disimpan di STL_UTILITYTEXT, jalankan SQL berikut.

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS query_statement
from stl_utilitytext where query=pg_last_query_id();
```

Untuk menggunakan SQL yang direkonstruksi yang dihasilkan di klien Anda, ganti karakter khusus (\n) dengan baris baru.

```
                                                query_statement


-------------------------------------------------------------------------------------------------
 set query_group to
 '0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000\n00000000000
                  000000';
```

STL_VACUUM

Menampilkan statistik baris dan blok untuk tabel yang telah disedot.

Tampilan menunjukkan informasi spesifik kapan setiap operasi vakum dimulai dan selesai, dan menunjukkan manfaat menjalankan operasi. Untuk informasi tentang persyaratan untuk menjalankan perintah ini, lihat deskripsi VAKUM perintah.

STL_VACUUM hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_VACUUM_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID transaksi untuk pernyataan VACUUM. Anda dapat menggabungkan tabel ini ke tampilan STL_QUERY untuk melihat pernyataan SQL individual yang dijalankan untuk transaksi VACUUM tertentu. Jika Anda menyedot seluruh database, setiap tabel disedot dalam transaksi terpisah.
tabel_id	integer	ID tabel.

Nama kolom	Jenis data	Deskripsi
status	karakter (30)	Status operasi VACUUM untuk setiap tabel. Nilai yang mungkin adalah sebagai berikut: • **Started** • **Started Delete Only** • **Started Delete Only (Sorted >= nn%)** Hanya fase penghapusan yang dimulai untuk VACUUM FULL. Fase pengurutan dilewati karena tabel sudah diurutkan pada atau di atas ambang pengurutan. • **Started Sort Only** • **Started Ranged Partition** • **Started Reindex** • **Finished** Waktu operasi selesai untuk tabel. Untuk mengetahui berapa lama operasi vakum berlangsung pada tabel tertentu, kurangi Waktu Mulai dari Waktu Selesai untuk ID transaksi dan ID tabel tertentu. • **Skipped** Tabel dilewati karena tabel diurutkan sepenuhnya dan tidak ada baris yang ditandai untuk dihapus. • **Skipped (delete only)** Tabel dilewati karena DELETE ONLY ditentukan dan tidak ada baris yang ditandai untuk dihapus. • **Skipped (sort only)** Tabel dilewati karena SORT ONLY ditentukan dan tabel sudah diurutkan sepenuhnya diurutkan. • **Skipped (sort only, sorted>=xx%)**

Nama kolom	Jenis data	Deskripsi
		Tabel dilewati karena SORT ONLY ditentukan dan tabel sudah diurutkan pada atau di atas ambang pengurutan. • **`Skipped (0 rows)`** Meja itu dilewati karena kosong. • **`VacuumBG`** Operasi vakum otomatis dilakukan di latar belakang. Status ini ditambahkan ke status lain ketika mereka dilakukan secara otomatis. Misalnya, penghapusan hanya vakum yang dilakukan secara otomatis akan memiliki baris awal dengan status`[VacuumBG] Started Delete Only`. Untuk informasi selengkapnya tentang pengaturan ambang batas pengurutan VACUUM, lihatVAKUM.
baris	bigint	Jumlah sebenarnya dari baris dalam tabel ditambah baris yang dihapus yang masih disimpan di disk (menunggu untuk disedot). Kolom ini menunjukkan jumlah sebelum vakum dimulai untuk baris dengan **`Started`** status, dan hitungan setelah vakum untuk baris dengan **`Finished`** status.
baris yang diurutkan	integer	Jumlah baris dalam tabel yang diurutkan. Kolom ini menunjukkan jumlah sebelum vakum dimulai untuk baris dengan **`Started`** kolom Status, dan hitungan setelah vakum untuk baris dengan **`Finished`** di kolom Status.
blok	integer	Jumlah total blok data yang digunakan untuk menyimpan data tabel sebelum operasi vakum (baris dengan **`Started`** status) dan setelah operasi vakum (**`Finished`**kolom). Setiap blok data menggunakan 1 MB.

Nama kolom	Jenis data	Deskripsi
max_merge_partisi	integer	Kolom ini digunakan untuk analisis kinerja dan mewakili jumlah maksimum partisi yang dapat diproses vakum untuk tabel per iterasi fase penggabungan. (Vacuum mengurutkan wilayah yang tidak disortir menjadi satu atau lebih partisi yang diurutkan. Bergantung pada jumlah kolom dalam tabel dan konfigurasi Amazon Redshift saat ini, fase penggabungan dapat memproses jumlah maksimum partisi dalam satu iterasi penggabungan. Fase penggabungan akan tetap berfungsi jika jumlah partisi yang diurutkan melebihi jumlah maksimum partisi penggabungan, tetapi lebih banyak iterasi penggabungan akan diperlukan.)
waktu acara	timestamp	Ketika operasi vakum dimulai atau selesai.
baris yang dapat dipulihkan	bigint	Jumlah baris yang dapat dipulihkan untuk cutoff_xid saat ini. Kolom ini menunjukkan perkiraan jumlah baris yang dapat direklamasi Redshift sebelum vakum dimulai untuk baris dengan **`Started`** status, dan jumlah sebenarnya dari baris yang dapat direklamasi yang tersisa setelah vakum untuk baris dengan status. **`Finished`**
reclaimable_space_mb	bigint	Ruang yang dapat dipulihkan dalam MB untuk cutoff_xid saat ini. Kolom ini menunjukkan perkiraan jumlah ruang yang dapat direklamasi Redshift sebelum vakum dimulai untuk baris dengan **`Started`** status, dan jumlah aktual ruang yang dapat dipulihkan yang tersisa setelah vakum untuk baris dengan status. **`Finished`**
cutoff_xid	bigint	ID transaksi cutoff untuk operasi VACUUM. Setiap transaksi setelah cutoff tidak termasuk dalam operasi VACUUM.
is_recluster	integer	Jika 1 (benar), operasi VACUUM mengeksekusi algoritma recluster, Jika 0 (false), ternyata tidak.

Kueri Sampel

Kueri berikut melaporkan statistik vakum untuk tabel 108313. Tabel disedot setelah serangkaian sisipan dan penghapusan.

```
select xid, table_id, status, rows, sortedrows, blocks, eventtime,
                reclaimable_rows, reclaimable_space_mb
from stl_vacuum where table_id=108313 order by eventtime;

 xid   | table_id | status                  | rows | sortedrows | blocks |  eventtime
         | reclaimable_rows | reclaimable_space_mb
-------+----------+-------------------------+------+------------+--------
+----------------------+------------------+----------------------
 14294 |   108313 | Started                 | 1950 |        408 |     28 |  2016-05-19
 17:36:01 |              984 |                   17
 14294 |   108313 | Finished                |  966 |        966 |     11 |  2016-05-19
 18:26:13 |                0 |                    0
 15126 |   108313 | Skipped(sorted>=95%)    |  966 |        966 |     11 |  2016-05-19
 18:26:38 |                0 |                    0
```

Pada awal VACUUM, tabel berisi 1.950 baris yang disimpan dalam 28 blok 1 MB. Amazon Redshift memperkirakan dapat merebut kembali 984, atau 17 blok ruang disk, dengan operasi vakum.

Di baris untuk status Selesai, kolom ROWS menunjukkan nilai 966, dan nilai kolom BLOCKS adalah 11, turun dari 28. Vakum merebut kembali perkiraan jumlah ruang disk, tanpa baris atau ruang yang dapat dipulihkan setelah operasi vakum selesai.

Pada fase pengurutan (transaksi 15126), vakum dapat melewati tabel karena baris dimasukkan dalam urutan kunci pengurutan.

Contoh berikut menunjukkan statistik untuk vakum SORT ONLY pada tabel SALES (tabel 110116 dalam contoh ini) setelah operasi INSERT besar:

```
vacuum sort only sales;

select xid, table_id, status, rows, sortedrows, blocks, eventtime
from stl_vacuum order by xid, table_id, eventtime;

xid |table_id|     status      | rows  |sortedrows|blocks|     eventtime
----+--------+-----------------+-------+----------+------+--------------------
...
```

```
2925| 110116 |Started Sort Only|1379648|   172456 |  132 | 2011-02-24 16:25:21...
2925| 110116 |Finished         |1379648|  1379648 |  132 | 2011-02-24 16:26:28...
```

STL_WINDOW

Menganalisis langkah-langkah kueri yang melakukan fungsi jendela.

STL_WINDOW terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STL_WINDOW hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses rencana penjelasan untuk kueri yang dijalankan pada klaster utama, cluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor yang mengidentifikasi irisan tempat kueri berjalan.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik

Nama kolom	Jenis data	Deskripsi
		pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
waktu akhir	timestamp	Waktu dalam UTC saat kueri selesai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **2009-06-12 11:29:19.131358** .
tasknum	integer	Jumlah proses tugas kueri yang ditugaskan untuk menjalankan langkah.
baris	bigint	Jumlah total baris yang diproses.
berbasis-disk	karakter (1)	Jika true (t), query dilakukan sebagai operasi berbasis disk. Jika false (f), query dilakukan dalam memori.
workmem	bigint	Jumlah total byte dalam memori kerja yang ditetapkan untuk langkah.

Kueri Sampel

Contoh berikut mengembalikan hasil fungsi jendela untuk slice 0 dan segmen 3.

```
select query, tasknum, rows, is_diskbased, workmem
from stl_window
where slice=0 and segment=3;
```

```
 query | tasknum | rows | is_diskbased | workmem
-------+---------+------+--------------+----------
 86326 |      36 | 1857 | f            | 95256616
   705 |      15 | 1857 | f            | 95256616
 86399 |      27 | 1857 | f            | 95256616
   649 |      10 |    0 | f            | 95256616
(4 rows)
```

STL_WLM_ERROR

Catat semua WLM-related kesalahan saat terjadi.

STL_WLM_ERROR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
waktu rekor	timestamp	Waktu kesalahan terjadi.
pid	integer	ID untuk proses yang menghasilkan kesalahan.
error_string	karakter (256)	Deskripsi kesalahan.

STL_WLM_RULE_ACTION

Merekam detail tentang tindakan yang dihasilkan dari aturan pemantauan kueri WLM yang terkait dengan antrian yang ditentukan pengguna. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

STL_WLM_RULE_ACTION terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	Pengguna yang menjalankan query.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri.
kelas_layanan	integer	ID untuk kelas layanan. Antrian kueri didefinisikan dalam konfigurasi WLM. Kelas layanan yang lebih besar dari 5 adalah antrian yang ditentukan pengguna.
aturan	karakter (256)	Nama aturan pemantauan kueri.
tindakan	karakter (256)	Tindakan yang dihasilkan. Kemungkinan nilainya adalah sebagai berikut: • log • hop (menetapkan kembali) • hop (mulai ulang) • membatalkan • ubah_kueri_prioritas • none Nilai none menunjukkan bahwa predikat aturan terpenuhi tetapi tindakan digantikan oleh aturan lain dengan tindakan tingkat keparahan yang lebih tinggi.
waktu rekor	timestamp	Waktu tindakan dicatat di UTC.
nilai_tindakan	karakter (256)	Jika `action` ya`change_query_priority` , maka nilai yang mungkin adalah`highest`,`high`,`normal`,`low`, dan`lowest`. Jika `action` ya `log`hop,, `abort` atau nilainya kosong.
nama_kelas layanan	karakter (64)	Nama kelas layanan.

Kueri Sampel

Contoh berikut menemukan kueri yang dihentikan oleh aturan pemantauan kueri.

```
Select query, rule
from stl_wlm_rule_action
where action = 'abort'
order by query;
```

STL_WLM_QUERY

Berisi catatan dari setiap percobaan eksekusi kueri di kelas layanan yang ditangani oleh WLM.

STL_WLM_QUERY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	integer	ID transaksi dari query atau subquery.
tugas	integer	ID digunakan untuk melacak kueri melalui pengelola beban kerja. Dapat dikaitkan dengan beberapa ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
kueri	integer	ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
kelas_layanan	integer	ID untuk kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

Nama kolom	Jenis data	Deskripsi
Slot_count	integer	Jumlah slot kueri WLM yang digunakan kueri sesuai dengan tingkat konkurensi yang ditetapkan untuk antrian. Default-nya adalah 1. Untuk informasi selengkapnya, lihat wlm _query_slot_count.
waktu mulai layanan_kelas_layanan	timestamp	Waktu kueri ditugaskan ke kelas layanan. Waktu ini berada di zona waktu UTC.
waktu mulai antrean	timestamp	Waktu kueri memasuki antrian untuk kelas layanan. Waktu ini berada di zona waktu UTC.
Antrian_end_time	timestamp	Waktu ketika kueri meninggalkan antrian untuk kelas layanan. Waktu ini berada di zona waktu UTC.
total_antre_waktu_	bigint	Jumlah total mikrodetik yang dihabiskan kueri dalam antrian
waktu mulai eksekusi	timestamp	Waktu kueri mulai dijalankan di kelas layanan. Waktu ini berada di zona waktu UTC.
waktu akhir eksekusi	timestamp	Waktu kueri menyelesaikan eksekusi di kelas layanan. Waktu ini berada di zona waktu UTC.
total_waktu_eksekusi	bigint	Jumlah mikrodetik yang dihabiskan kueri untuk mengeksekusi.
waktu_akhir_kelas_layanan	timestamp	Waktu kueri meninggalkan kelas layanan. Waktu ini berada di zona waktu UTC.
keadaan akhir	karakter (16)	Dicadangkan untuk penggunaan sistem.
est_peak_mem	bigint	Dicadangkan untuk penggunaan sistem.
kueri_prioritas	arang (20)	Prioritas kueri. Nilai yang mungkin adalah n/alowest,low,normal,high,, danhighest, yang n/a berarti bahwa prioritas kueri tidak didukung.

Nama kolom	Jenis data	Deskripsi
nama_kelas layanan	karakter (64)	Nama kelas layanan. Untuk informasi selengkap nya tentang kelas layanan, lihat tabel dan tampilan sistem WLM.

Kueri Sampel

Lihat kueri rata-rata Waktu dalam antrian dan eksekusi

Kueri berikut menampilkan konfigurasi saat ini untuk kelas layanan yang lebih besar dari 4. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

Kueri berikut mengembalikan waktu rata-rata (dalam mikrodetik) yang dihabiskan setiap kueri dalam antrian kueri dan dieksekusi untuk setiap kelas layanan.

```
select service_class as svc_class, count(*),
avg(datediff(microseconds, queue_start_time, queue_end_time)) as avg_queue_time,
avg(datediff(microseconds, exec_start_time, exec_end_time )) as avg_exec_time
from stl_wlm_query
where service_class > 4
group by service_class
order by service_class;
```

Kueri ini mengembalikan output sampel berikut:

```
 svc_class | count | avg_queue_time | avg_exec_time
-----------+-------+----------------+---------------
         5 | 20103 |              0 |         80415
         5 |  3421 |          34015 |        234015
         6 |    42 |              0 |        944266
         7 |   196 |           6439 |       1364399
(4 rows)
```

Lihat waktu kueri maksimum dalam antrian dan eksekusi

Kueri berikut mengembalikan jumlah waktu maksimum (dalam mikrodetik) yang dihabiskan kueri dalam antrian kueri dan dieksekusi untuk setiap kelas layanan.

```
select service_class as svc_class, count(*),
max(datediff(microseconds, queue_start_time, queue_end_time)) as max_queue_time,
```

```
max(datediff(microseconds, exec_start_time, exec_end_time )) as max_exec_time
from stl_wlm_query
where svc_class > 5
group by service_class
order by service_class;
```

```
 svc_class | count | max_queue_time | max_exec_time
-----------+-------+----------------+---------------
         6 |    42 |              0 |       3775896
         7 |   197 |          37947 |      16379473
(4 rows)
```

Tabel STV untuk data snapshot

Tabel STV adalah tabel sistem virtual yang berisi snapshot dari data sistem saat ini.

Topik

- STV_ACTIVE_KURSOR
- STV_BLOCKLIST
- STV_KURSOR_KONFIGURASI
- STV_DB_LEVEL_ISOLASI
- STV_EXEC_STATE
- STV_INFLIGHT
- STV_LOAD_STATE
- STV_LOCKS
- STV_ML_MODEL_INFO
- STV_MV_DEPS
- STV_MV_INFO
- STV_NODE_STORAGE_KAPASITAS
- STV_PARTISI
- STV_QUERY_METRICS
- STV_TERBARU
- STV_SESSION
- STV_SLICE
- STV_STARTUP_RECOVERY_STATE

- STV_TBL_PERM
- STV_TBL_TRANS
- STV_WLM_CLASSICATION_CONFIG
- STV_WLM_QMR_CONFIG
- STV_WLM_QUERY_QUEUE_STATE
- STV_WLM_QUERY_STATE
- STV_WLM_QUERY_TASKING_STATE
- STV_WLM_SERVICE_CLASS_CONFIG
- STV_WLM_SERVICE_CLASS_STATE
- STV_XRESTORE_ALTER_QUEUE_STATE

STV_ACTIVE_KURSOR

STV_ACTIVE_CURSORS menampilkan detail untuk kursor yang sedang terbuka. Untuk informasi selengkapnya, lihat MENYATAKAN.

STV_ACTIVE_CURSORS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem. Pengguna hanya dapat melihat kursor yang dibuka oleh pengguna tersebut. Seorang superuser dapat melihat semua kursor.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
name	karakter (256)	Nama kursor.
xid	bigint	Konteks transaksi.
pid	integer	Proses pemimpin menjalankan kueri.
waktu mulai	timestam	Waktu ketika kursor dideklarasikan.

Nama kolom	Jenis data	Deskripsi
hitung_baris	bigint	Jumlah baris dalam set hasil kursor.
hitung_byte	bigint	Jumlah byte dalam set hasil kursor.
ditampat_baris	bigint	Jumlah baris yang saat ini diambil dari kumpulan hasil kursor.

STV_BLOCKLIST

STV_BLOCKLIST berisi jumlah blok disk 1 MB yang digunakan oleh setiap irisan, tabel, atau kolom dalam database.

Gunakan kueri agregat dengan STV_BLOCKLIST, seperti yang ditunjukkan contoh berikut, untuk menentukan jumlah blok disk 1 MB yang dialokasikan per database, tabel, irisan, atau kolom. Anda juga dapat menggunakan STV_PARTISI untuk melihat informasi ringkasan tentang pemanfaatan disk.

STV_BLOCKLIST hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> STV_BLOCKLIST hanya merekam blok yang dimiliki oleh cluster yang disediakan atau namespace tanpa server. Jika database menyertakan blok yang dibagikan dari produsen datashare, blok tersebut tidak disertakan dalam STV_BLOCKLIST. Untuk informasi lebih lanjut tentang berbagi data, buka. Berbagi data di Amazon Redshift

Kolom tabel

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Irisan simpul.

Nama kolom	Jenis data	Deskripsi
Col	integer	Zero-based indeks untuk kolom. Setiap tabel yang Anda buat memiliki tiga kolom tersembunyi yang ditambahkan padanya: INSERT_XID, DELETE_XID, dan ROW_ID (OID). Tabel dengan 3 kolom yang ditentukan pengguna berisi 6 kolom aktual, dan kolom yang ditentukan pengguna diberi nomor internal sebagai 0, 1, dan 2. Kolom INSERT_XID, DELETE_XID, dan ROW_ID diberi nomor 3, 4, dan 5, masing-masing, dalam contoh ini.
tbl	integer	ID tabel untuk tabel database.
Blocknum	integer	ID untuk blok data.
nilai_angka	integer	Jumlah nilai yang terkandung di blok.
batas_diperpanjang	integer	Untuk penggunaan internal.
minvalue	bigint	Nilai data minimum blok. Menyimpan delapan karakter pertama sebagai bilangan bulat 64-bit untuk data non-numerik. Digunakan untuk pemindaian disk.
nilai maksimum	bigint	Nilai data maksimum blok. Menyimpan delapan karakter pertama sebagai bilangan bulat 64-bit untuk data non-numerik. Digunakan untuk pemindaian disk.
sb_pos	integer	Pengidentifikasi internal Amazon Redshift untuk posisi blok super pada disk.
disematkan	integer	Apakah blok disematkan ke dalam memori sebagai bagian dari pra-pemuatan. 0 = salah; 1 = benar. Defaultnya false.
on_disk	integer	Apakah blok disimpan secara otomatis pada disk atau tidak. 0 = salah; 1 = benar. Defaultnya false.
dimodifikasi	integer	Apakah blok telah dimodifikasi atau tidak. 0 = salah; 1 = benar. Defaultnya false.

Nama kolom	Jenis data	Deskripsi
hdr_dimodifikasi	integer	Apakah header blok telah dimodifikasi atau tidak. 0 = false; 1 = true. Defaultnya false.
tidak disortir	integer	Apakah blok tidak disortir atau tidak. 0 = salah; 1 = benar. Defaultnya benar.
batu nisan	integer	Untuk penggunaan internal.
preferred_diskno	integer	Nomor disk yang harus diaktifkan blok, kecuali disk telah gagal. Setelah disk diperbaiki, blok akan pindah kembali ke disk ini.
sementara	integer	Apakah blok berisi data sementara atau tidak, seperti dari tabel sementara atau hasil kueri perantara. 0 = false; 1 = true. Defaultnya false.
blok baru	integer	Menunjukkan apakah blok baru (benar) atau tidak pernah diikat ke disk (false). 0 = false; 1 = true.
pembaca_nomor	integer	Jumlah referensi di setiap blok.
bendera	integer	Bendera Amazon Redshift internal untuk header blok.

Kueri Sampel

STV_BLOCKLIST berisi satu baris per blok disk yang dialokasikan, sehingga kueri yang memilih semua baris berpotensi mengembalikan sejumlah besar baris. Sebaiknya gunakan hanya kueri agregat dengan STV_BLOCKLIST.

T SVV_DISKUSI ampilan memberikan informasi serupa dalam format yang lebih ramah pengguna; namun, contoh berikut menunjukkan satu penggunaan tabel STV_BLOCKLIST.

Untuk menentukan jumlah blok 1 MB yang digunakan oleh setiap kolom dalam tabel VENUE, ketik kueri berikut:

```
select col, count(*)
from stv_blocklist, stv_tbl_perm
```

```
where stv_blocklist.tbl = stv_tbl_perm.id
and stv_blocklist.slice = stv_tbl_perm.slice
and stv_tbl_perm.name = 'venue'
group by col
order by col;
```

Kueri ini mengembalikan jumlah blok 1 MB yang dialokasikan untuk setiap kolom dalam tabel VENUE, ditunjukkan oleh data sampel berikut:

```
 col | count
-----+-------
   0 |  4
   1 |  4
   2 |  4
   3 |  4
   4 |  4
   5 |  4
   7 |  4
   8 |  4
(8 rows)
```

Kueri berikut menunjukkan apakah data tabel benar-benar didistribusikan ke semua irisan atau tidak:

```
select trim(name) as table, stv_blocklist.slice, stv_tbl_perm.rows
from stv_blocklist,stv_tbl_perm
where stv_blocklist.tbl=stv_tbl_perm.id
and stv_tbl_perm.slice=stv_blocklist.slice
and stv_blocklist.id > 10000 and name not like '%#m%'
and name not like 'systable%'
group by name, stv_blocklist.slice, stv_tbl_perm.rows
order by 3 desc;
```

Kueri ini menghasilkan output sampel berikut, menunjukkan distribusi data genap untuk tabel dengan baris terbanyak:

```
table   | slice | rows
----------+-------+-------
listing  |    13 | 10527
listing  |    14 | 10526
listing  |     8 | 10526
```

```
listing  |     9 | 10526
listing  |     7 | 10525
listing  |     4 | 10525
listing  |    17 | 10525
listing  |    11 | 10525
listing  |     5 | 10525
listing  |    18 | 10525
listing  |    12 | 10525
listing  |     3 | 10525
listing  |    10 | 10525
listing  |     2 | 10524
listing  |    15 | 10524
listing  |    16 | 10524
listing  |     6 | 10524
listing  |    19 | 10524
listing  |     1 | 10523
listing  |     0 | 10521
...
(180 rows)
```

Kueri berikut menentukan apakah ada blok tombstoned yang dikomit ke disk:

```
select slice, col, tbl, blocknum, newblock
from stv_blocklist
where  tombstone > 0;

slice | col |   tbl  | blocknum | newblock
-------+-----+--------+----------+----------
4     |  0  | 101285 |    0     |   1
4     |  2  | 101285 |    0     |   1
4     |  4  | 101285 |    1     |   1
5     |  2  | 101285 |    0     |   1
5     |  0  | 101285 |    0     |   1
5     |  1  | 101285 |    0     |   1
5     |  4  | 101285 |    1     |   1
...
(24 rows)
```

STV_KURSOR_KONFIGURASI

STV_CURSOR_CONFIGURATION menampilkan batasan konfigurasi kursor. Untuk informasi selengkapnya, lihat Kendala kursor.

STV_CURSOR_CONFIGURATION hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kursor_hitun_saat ini	integer	Jumlah kursor yang saat ini terbuka.
max_diskspace_dapat digunakan	integer	Jumlah ruang disk yang tersedia untuk kursor, dalam megabyte. Kendala ini didasarkan pada ukuran set hasil kursor maksimum untuk cluster.
ruang disk_digunakan saat ini	integer	Jumlah ruang disk yang saat ini digunakan oleh kursor, dalam megabyte.

STV_DB_LEVEL_ISOLASI

STV_DB_ISOLATION_LEVEL menampilkan tingkat isolasi saat ini untuk database. Untuk informasi selengkapnya tentang tingkat isolasi, lihatTingkat isolasi di Amazon Redshift.

STV_DB_ISOLATION_LEVEL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
db_name	karakter (128)	Nama database.

Nama kolom	Jenis data	Deskripsi
tingkat isolasi	karakter (20)	Tingkat isolasi database. Nilai yang mungkin termasuk `Serializable` dan`Snapshot Isolation` .

STV_EXEC_STATE

Gunakan tabel STV_EXEC_STATE untuk mengetahui informasi tentang kueri dan langkah-langkah kueri yang secara aktif berjalan pada node komputasi.

Informasi ini biasanya digunakan hanya untuk memecahkan masalah teknik. Tampilan SVV_QUERY_STATE dan SVL_QUERY_SUMMARY mengekstrak informasi mereka dari STV_EXEC_STATE.

STV_EXEC_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_DETAIL. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
mengiris	integer	Irisan simpul di mana langkah selesai.
segmen	integer	Segmen kueri yang dijalankan. Segmen query adalah serangkaian langkah.

Nama kolom	Jenis data	Deskripsi
langkah	integer	Langkah segmen query yang selesai. Langkah adalah unit terkecil yang dilakukan kueri.
waktu mulai	timestamp	Waktu langkah itu berjalan.
waktu saat ini	timestamp	Waktu saat ini.
tasknum	integer	Proses tugas kueri yang ditugaskan untuk menyelesaikan langkah.
baris	bigint	Jumlah baris yang diproses.
byte	bigint	Jumlah byte yang diproses.
label	karakter (256)	Label langkah, yang terdiri dari nama langkah kueri dan, bila berlaku, ID tabel dan nama tabel (misalnya,`scan tbl=100448 name =user`). Three-digit ID tabel biasanya mengacu pada pemindaian tabel transien. Ketika Anda melihat`tbl=0`, biasanya mengacu pada pemindaian nilai konstan.
berbasis-disk	haru (1)	Apakah langkah kueri ini diselesaikan sebagai operasi berbasis disk: true (**`t`**) atau false (**`f`**). Hanya langkah-l angkah tertentu, seperti langkah hash, sort, dan agregat, yang dapat masuk ke disk. Banyak jenis langkah selalu diselesaikan dalam memori.
workmem	bigint	Jumlah byte memori kerja yang ditetapkan untuk langkah.
nom_bagian	integer	Jumlah partisi yang dibagi menjadi tabel hash selama langkah hash. Angka positif dalam kolom ini tidak menyiratkan bahwa langkah hash berjalan sebagai operasi berbasis disk. Periksa nilai di kolom IS_DISKBASED untuk melihat apakah langkah hash berbasis disk.

Nama kolom	Jenis data	Deskripsi
is_rrscan	harimau (1)	Jika true (**t**), menunjukkan bahwa pemindaian terbatas rentang digunakan pada langkah tersebut. Defaultnya adalah false (**f**).
is_tertunda_scan	harimau (1)	Jika true (**t**), menunjukkan bahwa pemindaian tertunda digunakan pada langkah tersebut. Defaultnya adalah false (**f**).
Query_uuid	karakter (36)	Pengidentifikasi unik global (UUID) dari kueri.

Kueri Sampel

Alih-alih menanyakan STV_EXEC_STATE secara langsung, Amazon Redshift merekomendasikan untuk menanyakan SVL_QUERY_SUMMARY atau SVV_QUERY_STATE untuk mendapatkan informasi dalam STV_EXEC_STATE dalam format yang lebih ramah pengguna. Lihat dokumentasi SVV_QUERY_STATE tabel SVL_QUERY_SUMMARY atau untuk lebih jelasnya.

STV_INFLIGHT

Gunakan tabel STV_INFLIGHT untuk menentukan kueri apa yang sedang berjalan di cluster. Jika Anda memecahkan masalah, akan sangat membantu untuk memeriksa status kueri yang berjalan lama.

STV_INFLIGHT tidak menampilkan kueri hanya leader-node. Untuk informasi selengkapnya, lihat Node—hanya fungsi utama. STV_INFLIGHT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Pemecahan masalah dengan STV_INFLIGHT

Jika Anda menggunakan STV_INFLIGHT untuk memecahkan masalah kinerja untuk kueri, atau kumpulan kueri, perhatikan hal berikut:

- Long-running transaksi terbuka umumnya meningkatkan beban. Transaksi terbuka ini dapat menghasilkan waktu berjalan yang lebih lama untuk kueri lain.
- Long-running Pekerjaan COPY dan ETL dapat memengaruhi kueri lain yang berjalan di cluster, jika mereka mengambil banyak sumber daya komputasi. Dalam kebanyakan kasus, memindahkan pekerjaan jangka panjang ini ke waktu penggunaan rendah meningkatkan kinerja untuk beban kerja pelaporan atau analitik.
- Ada pandangan yang memberikan informasi terkait dengan STV_INFLIGHT. Ini termasukSTL_QUERYTEXT, yang menangkap teks kueri untuk perintah SQL, danSVV_QUERY_DALAM PENERBANGAN, yang menggabungkan STV_INFLIGHT ke STL_QUERYTEXT. Anda juga dapat menggunakan STV_TERBARU dengan STV_INFLIGHT untuk pemecahan masalah. Misalnya, STV_RECENTS dapat menunjukkan apakah kueri tertentu berada dalam status Running atau Done. Menggabungkan informasi ini dengan hasil dari STV_INFLIGHT dapat memberi Anda informasi lebih lanjut tentang properti kueri dan dampak sumber daya komputasi.

Anda juga dapat memantau kueri yang sedang berjalan menggunakan konsol Amazon Redshift.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
mengiris	integer	Iris tempat kueri berjalan.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
label	karakter (320)	Baik nama file yang digunakan untuk menjalankan query atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau

Nama kolom	Jenis data	Deskripsi
		parameter QUERY_GROUP tidak disetel, bidang ini kosong.
xid	bigint	ID Transaksi.
pid	integer	ID proses. Semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama. Anda dapat menggunakan kolom ini untuk bergabung dengan STL_ERROR tabel.
waktu mulai	timestamp	Waktu permintaan dimulai.
text	karakter (100)	Teks kueri, dipotong menjadi 100 karakter jika pernyataan melebihi batas itu.
tergantung	integer	Apakah kueri ditangguhkan atau tidak. 0 = salah; 1 = benar.
insert_pristine	integer	Apakah kueri tulis are/were dapat dijalankan saat kueri saat ini is/was berjalan. 1 = tidak ada kueri penulisan yang diizinkan. 0 = kueri tulis diperbolehkan. Kolom ini dimaksudkan untuk digunakan dalam debugging.
status penskalaan_konkurensi	integer	Menunjukkan apakah kueri berjalan di cluster utama atau pada cluster penskalaan konkurensi, Nilai yang mungkin adalah sebagai berikut: 0 - Berjalan di cluster utama 1 - Berjalan pada cluster penskalaan konkurensi

Kueri Sampel

Untuk melihat semua kueri aktif yang sedang berjalan di database, ketik kueri berikut:

```
select * from stv_inflight;
```

Contoh output di bawah ini menunjukkan dua kueri yang sedang berjalan, termasuk kueri STV_INFLIGHT itu sendiri dan kueri yang dijalankan dari skrip yang disebut: `avgwait.sql`

```
select slice, query, trim(label) querylabel, pid,
starttime, substring(text,1,20) querytext
from stv_inflight;

slice|query|querylabel | pid |        starttime         |      querytext
-----+-----+-----------+-----+--------------------------+--------------------
1011 |  21 |           | 646 |2012-01-26 13:23:15.645503|select slice, query,
1011 |  20 |avgwait.sql| 499 |2012-01-26 13:23:14.159912|select avg(datediff(
(2 rows)
```

Kueri berikut memilih beberapa kolom, termasuk concurrency_scaling_status. Kolom ini menunjukkan apakah kueri sedang dikirim ke cluster penskalaan bersamaan. Jika nilainya 1 untuk beberapa hasil, ini merupakan indikasi bahwa sumber daya komputasi penskalaan bersamaan sedang digunakan. Untuk informasi selengkapnya, lihat Penskalaan konkurensi.

```
select userid,
query,
pid,
starttime,
text,
suspended,
concurrency_scaling_status
 from STV_INFLIGHT;
```

Output sampel menunjukkan satu kueri yang dikirim ke cluster penskalaan konkurensi.

```
 query  | pid     |        starttime           |   text                | suspended
 |  concurrency_scaling_status
--------+---------
+---------------------------|-----------------------|--------------|-----------------------
1234567 | 123456  | 2012-01-26 13:23:15.645503 | select userid, query...  0
    1
2345678 | 234567  | 2012-01-26 13:23:14.159912 | select avg(datediff(...  0
    0
(2 rows)
```

Untuk kiat selengkapnya tentang pemecahan masalah kinerja kueri, lihatPemecahan masalah kueri.

STV_LOAD_STATE

Gunakan tabel STV_LOAD_STATE untuk menemukan informasi tentang keadaan saat ini dari pernyataan COPY yang sedang berlangsung.

Perintah COPY memperbarui tabel ini setelah setiap juta catatan dimuat.

STV_LOAD_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
sesi	integer	Sesi PID proses melakukan pemuatan.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
mengiris	integer	Nomor irisan simpul.
pid	integer	ID proses. Semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini tetap konstan jika Anda menjalank an serangkaian kueri dalam sesi yang sama.
waktu rekor	timestamp	Waktu catatan dicatat.
Bytes_to_load	bigint	Jumlah total byte yang akan dimuat oleh irisan ini. Ini adalah 0 jika data yang dimuat dikompresi
Byte_dimuat	bigint	Jumlah byte yang dimuat oleh irisan ini. Jika data yang dimuat dikompresi, ini adalah jumlah byte yang dimuat setelah data tidak dikompresi.

Nama kolom	Jenis data	Deskripsi
Bytes_ke_muat_terkompresi	bigint	Jumlah total byte data terkompresi yang akan dimuat oleh irisan ini. Ini adalah 0 jika data yang dimuat tidak dikompresi.
byte_dimuat_terkompresi	bigint	Jumlah byte data terkompresi yang dimuat oleh irisan ini. Ini adalah 0 jika data yang dimuat tidak dikompresi.
lini	integer	Jumlah baris yang dimuat oleh irisan ini.
nom_file	integer	Jumlah file yang akan dimuat oleh irisan ini.
nom_file_lengkap	integer	Jumlah file yang dimuat oleh irisan ini.
berkas_saat ini	karakter (256)	Nama file yang dimuat oleh irisan ini.
pct_lengkap	integer	Persentase beban data yang diselesaikan oleh irisan ini.

Contoh kueri

Untuk melihat kemajuan setiap irisan untuk perintah SALIN, ketik kueri berikut. Contoh ini menggunakan fungsi PG_LAST_COPY_ID () untuk mengambil informasi untuk perintah COPY terakhir.

```
select slice , bytes_loaded, bytes_to_load , pct_complete from stv_load_state where
 query = pg_last_copy_id();

 slice | bytes_loaded | bytes_to_load | pct_complete
-------+--------------+---------------+--------------
     2 |            0 |             0 |            0
     3 |     12840898 |      39104640 |           32
(2 rows)
```

STV_LOCKS

Gunakan tabel STV_LOCKS untuk melihat pembaruan terkini pada tabel dalam database.

Amazon Redshift mengunci tabel untuk mencegah dua pengguna memperbarui tabel yang sama secara bersamaan. Sementara tabel STV_LOCKS menunjukkan semua pembaruan tabel saat ini, kueri STL_TR_KONFLIK tabel untuk melihat log konflik kunci. Gunakan SVV_TRANSAKSI tampilan untuk mengidentifikasi transaksi terbuka dan mengunci masalah pertikaian.

STV_LOCKS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
tabel_id	bigint	ID tabel untuk tabel yang memperoleh kunci.
komit terakhir	timestamp	Stempel waktu untuk komit terakhir dalam tabel.
pembaruan _terakhir	timestamp	Stempel waktu untuk pembaruan terakhir untuk tabel.
pengunci_ pemilik	bigint	ID transaksi yang terkait dengan kunci.
lock_owner_pid	bigint	ID proses yang terkait dengan kunci.
lock_owne r_start_ts	timestamp	Timestamp untuk waktu mulai transaksi.
lock_owne r_end_ts	timestamp	Timestamp untuk waktu akhir transaksi.
kunci_status	karakter (22)	Status proses baik menunggu atau memegang kunci.

Contoh kueri

Untuk melihat semua kunci yang terjadi dalam transaksi saat ini, ketik perintah berikut:

```
select table_id, last_update, lock_owner, lock_owner_pid from stv_locks;
```

Kueri ini mengembalikan output sampel berikut, yang menampilkan tiga kunci yang saat ini berlaku:

```
 table_id |        last_update         | lock_owner | lock_owner_pid
----------+----------------------------+------------+----------------
100004  | 2008-12-23 10:08:48.882319 |       1043 |           5656
100003  | 2008-12-23 10:08:48.779543 |       1043 |           5656
100140  | 2008-12-23 10:08:48.021576 |       1043 |           5656
(3 rows)
```

STV_ML_MODEL_INFO

Nyatakan informasi tentang keadaan model pembelajaran mesin saat ini.

STV_ML_MODEL_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
schema_name	karakter (128)	Namespace dari model.
user_name	karakter (128)	Pemilik model.
nama_model	karakter (128)	Nama modul.
siklus hidup	arang (20)	Status siklus hidup model.
dapat diperbaharui	integer	Status model apakah dapat disegarkan jika tabel dan kolom asli dalam kueri pelatihan masih ada dan pengguna masih memiliki izin untuk mereka. Nilai yang mungkin adalah: 1 (dapat disegarkan) dan 0 (tidak dapat disegarkan).
model_status	karakter (128)	Keadaan model saat ini.

Contoh kueri

Kueri berikut menampilkan keadaan model pembelajaran mesin saat ini.

```
SELECT schema_name, model_name, model_state
```

```
FROM stv_ml_model_info;

 schema_name |        model_name            |             model_state
-------------+------------------------------+-------------------------------------
 public      | customer_churn_auto_model    | Train Model On SageMaker In Progress
 public      | customer_churn_xgboost_model | Model is Ready
(2 row)
```

STV_MV_DEPS

Tabel STV_MV_DEPS menunjukkan dependensi tampilan yang terwujud pada tampilan terwujud lainnya dalam Amazon Redshift.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

STV_MV_DEPS terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat mencantumkan tampilan terwujud yang berada dalam skema yang mereka akses. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
db_name	karakter (128)	Basis data yang berisi tampilan terwujud yang ditentukan.
skema	karakter (128)	Skema tampilan yang terwujud.
name	karakter (128)	Nama tampilan yang terwujud.
Ref_Schema	karakter (128)	Skema tampilan terwujud yang menjadi sandaran tampilan terwujud ini.
nama referensi	karakter (128)	Nama pandangan terwujud yang menjadi sandaran pandangan terwujud ini.
ref_nama_database	karakter (128)	Nama database tempat tampilan terwujud ini bergantung.

Contoh kueri

Kueri berikut mengembalikan baris keluaran yang menunjukkan bahwa tampilan terwujud `mv_over_foo` menggunakan tampilan terwujud `mv_foo` dalam definisinya sebagai ketergantungan.

```
CREATE SCHEMA test_ivm_setup;
CREATE TABLE test_ivm_setup.foo(a INT);
CREATE MATERIALIZED VIEW test_ivm_setup.mv_foo AS SELECT * FROM test_ivm_setup.foo;
CREATE MATERIALIZED VIEW test_ivm_setup.mv_over_foo AS SELECT * FROM
 test_ivm_setup.mv_foo;

SELECT * FROM stv_mv_deps;

 db_name | schema          | name        |   ref_schema   | ref_name |
 ref_database_name
---------+-----------------+-------------+----------------+----------
+------------------
 dev     | test_ivm_setup  | mv_over_foo | test_ivm_setup | mv_foo   | dev
```

STV_MV_INFO

Tabel STV_MV_INFO berisi baris untuk setiap tampilan yang terwujud, apakah data basi, dan informasi status.

Untuk informasi selengkapnya tentang tampilan yang terwujud, lihatTampilan yang terwujud di Amazon Redshift.

STV_MV_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat mencantumkan tampilan terwujud yang berada dalam skema yang mereka akses. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
db_name	karakter (128)	Basis data yang berisi tampilan terwujud.
skema	karakter (128)	Skema database.
name	karakter (128)	Nama tampilan yang terwujud.

Nama kolom	Jenis data	Deskripsi
upto_xid yang diperbarui	bigint	Dicadangkan untuk penggunaan internal.
is_basi	harimau (1)	A `t` menunjukkan bahwa pandangan yang terwujud sudah basi. Tampilan terwujud basi adalah tampilan di mana tabel dasar telah diperbarui tetapi tampilan yang terwujud belum diperbarui. An `f` menunjukkan bahwa tampilan yang terwujud diperbarui dengan perubahan tabel dasar. A u menunjukkan kebuntuan yang tidak diketahui (misalnya, dalam kasus MV pada tabel eksternal biasa). Informasi mungkin tidak akurat jika penyegaran belum dijalankan sejak restart terakhir. `is_stale`Kolom selalu disetel ke `t` jika tampilan yang terwujud bergantung pada fungsi yang dapat berubah. Fungsi yang dapat berubah mengembal ikan hasil yang berbeda ketika diberikan argumen atau argumen yang sama. Misalnya, sebagian besar fungsi yang mengembalikan tanggal atau stempel waktu adalah fungsi yang dapat diubah.
nama_pengguna pemilik	karakter (128)	Pengguna yang memiliki tampilan terwujud.

Nama kolom	Jenis data	Deskripsi
status	integer	Keadaan pandangan yang terwujud sebagai berikut: • 0 — Tampilan terwujud sepenuhnya dihitung ulang saat di-refresh. • 1 — Pandangan yang terwujud bersifat inkremental. • 101 - Tampilan yang terwujud tidak dapat disegarkan karena kolom yang dijatuhkan. Kendala ini berlaku bahkan jika kolom tidak digunakan dalam tampilan yang terwujud. • 102 - Tampilan yang terwujud tidak dapat disegarkan karena jenis kolom yang diubah. Kendala ini berlaku bahkan jika kolom tidak digunakan dalam tampilan yang terwujud. • 103 — Tampilan terwujud tidak dapat disegarkan karena tabel berganti nama. • 104 - Tampilan terwujud tidak dapat disegarkan karena kolom berganti nama. Kendala ini berlaku bahkan jika kolom tidak digunakan dalam tampilan yang terwujud. • 105 - Tampilan terwujud tidak dapat disegarkan karena skema berganti nama.
penulisan ulang otomatis	harimau (1)	A `t` menunjukkan bahwa tampilan yang terwujud memenuhi syarat untuk penulisan ulang kueri secara otomatis.
Refresh otomatis	harimau (1)	A `t` menunjukkan bahwa tampilan yang terwujud dapat disegarkan secara otomatis.

Contoh kueri

Untuk melihat keadaan semua tampilan yang terwujud, jalankan kueri berikut.

```
select * from stv_mv_info;
```

Kueri ini mengembalikan output sampel berikut.

```
 db_name |       schema       |   name  | updated_upto_xid | is_stale | owner_user_name
 | state | autorefresh | autorewrite
---------+--------------------+---------+------------------+----------
+-----------------+-------+-------------+------------
 dev     | test_ivm_setup     | mv      |             1031 | f        | catch-22
 |     1 |           1 |           0
 dev     | test_ivm_setup     | old_mv  |              988 | t        | lotr
 |     1 |           0 |           1
```

STV_NODE_STORAGE_KAPASITAS

Tabel STV_NODE_STORAGE_CAPACITY menunjukkan rincian kapasitas penyimpanan total dan total kapasitas yang digunakan untuk setiap node dalam sebuah cluster. Ini berisi baris untuk setiap node.

STV_NODE_STORAGE_CAPACITY hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
simpul	integer	Nomor simpul.
digunakan	integer	Jumlah blok disk 1 MB yang saat ini digunakan pada node. Untuk tipe node RG dan RA3, blok yang digunakan mencakup blok cache lokal dan blok yang disimpan di Amazon S3.
kapasitas	integer	Total kapasitas penyimpanan yang disediaka n untuk node dalam blok 1 MB. Kapasitas mencakup ruang yang disediakan oleh Amazon Redshift pada tipe node DC2 untuk penggunaa n internal. Kapasitas lebih besar dari kapasitas

Nama kolom	Jenis data	Deskripsi
		node nominal, yang merupakan jumlah ruang simpul yang tersedia untuk data pengguna. Untuk tipe node RG dan RA3, kapasitas ini sama dengan total kuota penyimpanan terkelola untuk cluster. Untuk informasi selengkapnya tentang kapasitas menurut jenis simpul, lihat Detail jenis Node di Panduan Manajemen Amazon Redshift.

Kueri Sampel

> **Note**
>
> Hasil dari contoh berikut bervariasi berdasarkan spesifikasi node cluster Anda. Tambahkan kolom `capacity` ke SQL SELECT Anda untuk mengambil kapasitas cluster Anda.

Kueri berikut mengembalikan ruang yang digunakan dan kapasitas total dalam blok disk 1 MB. Contoh ini berjalan pada cluster dc2.8xlarge dua simpul.

```
select node, used from stv_node_storage_capacity order by node;
```

Kueri ini mengembalikan output sampel berikut.

```
 node | used
------+-------
    0 | 30597
    1 | 27089
```

Kueri berikut mengembalikan ruang yang digunakan dan kapasitas total dalam blok disk 1 MB. Contoh ini berjalan pada cluster ra3.16xlarge dua simpul.

```
select node, used from stv_node_storage_capacity order by node;
```

Kueri ini mengembalikan output sampel berikut.

```
 node | used
------+-------
    0 | 30591
    1 | 27103
```

STV_PARTISI

Gunakan tabel STV_PARTITIONS untuk mengetahui kinerja kecepatan disk dan pemanfaatan disk untuk Amazon Redshift.

STV_PARTITIONS berisi satu baris per node per volume disk logis.

STV_PARTITIONS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
owner	integer	Node disk yang memiliki partisi.
host	integer	Node yang secara fisik melekat pada partisi.
disk	integer	Disk yang berisi partisi.
bagian_mulai	bigint	Offset partisi. Perangkat mentah secara logis dipartisi untuk membuka ruang untuk blok cermin.
bagian_akhir	bigint	Akhir dari partisi.
digunakan	integer	Jumlah blok disk 1 MB yang saat ini digunakan pada partisi.
terlempar	integer	Jumlah blok yang siap dihapus tetapi belum dihapus karena tidak aman untuk membebaskan alamat disk mereka. Jika alamat dibebaskan segera, transaksi yang tertunda dapat menulis ke lokasi yang sama pada disk. Oleh karena itu, blok yang dilemparkan ini dirilis pada komit berikutnya. Blok disk mungkin ditandai sebagai lempar, misalnya, ketika

Nama kolom	Jenis data	Deskripsi
		kolom tabel dijatuhkan, selama operasi INSERT, atau selama operasi kueri berbasis disk.
kapasitas	integer	Total kapasitas partisi dalam blok disk 1 MB.
membaca	bigint	Jumlah pembacaan yang telah terjadi sejak restart cluster terakhir.
menulis	bigint	Jumlah penulisan yang telah terjadi sejak restart cluster terakhir.
mencari-maju	integer	Berapa kali permintaan bukan untuk alamat berikutnya yang diberikan alamat permintaan sebelumnya.
mencari-kembali	integer	Berapa kali permintaan bukan untuk alamat sebelumnya yang diberikan alamat berikutnya.
is_san	integer	Apakah partisi milik SAN. Nilai yang valid adalah **0** (false) atau **1** (true).
gagal	integer	Kolom ini tidak digunakan lagi.
mbps	integer	Kecepatan disk dalam megabyte per detik.
menaiki	karakter (256)	Jalur direktori ke perangkat.

Contoh kueri

Kueri berikut mengembalikan ruang disk yang digunakan dan kapasitas, dalam blok disk 1 MB, dan menghitung pemanfaatan disk sebagai persentase ruang disk mentah. Ruang disk mentah mencakup ruang yang disediakan oleh Amazon Redshift untuk penggunaan internal, sehingga lebih besar dari kapasitas disk nominal, yang merupakan jumlah ruang disk yang tersedia untuk pengguna. Metrik Per sentase Ruang Disk yang Digunakan pada tab Kin erja di Amazon Redshift Management Console melaporkan persentase kapasitas disk nominal yang digunakan oleh cluster Anda. Sebaiknya Anda memantau metrik Per sentase Ruang Disk yang Digunakan untuk mempertahankan penggunaan Anda dalam kapasitas disk nominal cluster Anda.

> **Important**
>
> Kami sangat menyarankan agar Anda tidak melebihi kapasitas disk nominal cluster Anda. Meskipun mungkin secara teknis mungkin dalam keadaan tertentu, melebihi kapasitas disk nominal Anda mengurangi toleransi kesalahan cluster Anda dan meningkatkan risiko kehilangan data.

Contoh ini dijalankan pada cluster dua node dengan enam partisi disk logis per node. Ruang digunakan dengan sangat merata di seluruh disk, dengan sekitar 25% dari setiap disk yang digunakan.

```
select owner, host, diskno, used, capacity,
(used-tossed)/capacity::numeric *100 as pctused
from stv_partitions order by owner;

 owner | host | diskno |  used  | capacity | pctused
-------+------+--------+--------+----------+---------
   0   |  0   |    0   | 236480 |  949954  | 24.9
   0   |  0   |    1   | 236420 |  949954  | 24.9
   0   |  0   |    2   | 236440 |  949954  | 24.9
   0   |  1   |    2   | 235150 |  949954  | 24.8
   0   |  1   |    1   | 237100 |  949954  | 25.0
   0   |  1   |    0   | 237090 |  949954  | 25.0
   1   |  1   |    0   | 236310 |  949954  | 24.9
   1   |  1   |    1   | 236300 |  949954  | 24.9
   1   |  1   |    2   | 236320 |  949954  | 24.9
   1   |  0   |    2   | 237910 |  949954  | 25.0
   1   |  0   |    1   | 235640 |  949954  | 24.8
   1   |  0   |    0   | 235380 |  949954  | 24.8

(12 rows)
```

STV_QUERY_METRICS

Berisi informasi metrik, seperti jumlah baris yang diproses, penggunaan CPU, dan penggunaan disk input/output, untuk kueri aktif yang berjalan dalam antrian kueri yang ditentukan pengguna (kelas layanan). Untuk melihat metrik untuk kueri yang telah selesai, lihat tabel STL_QUERY_METRICS sistem.

Metrik kueri diambil sampelnya pada interval satu detik. Akibatnya, proses yang berbeda dari kueri yang sama mungkin mengembalikan waktu yang sedikit berbeda. Selain itu, segmen kueri yang berjalan dalam waktu kurang dari 1 detik mungkin tidak direkam.

STV_QUERY_METRICS melacak dan menggabungkan metrik pada tingkat kueri, segmen, dan langkah. Untuk informasi tentang segmen kueri dan langkah-langkah, lihatPerencanaan kueri dan alur kerja pelaksanaan. Banyak metrik (sepertimax_rows,cpu_time, dan seterusnya) dijumlahkan di seluruh irisan node. Untuk informasi selengkapnya tentang irisan simpul, lihatArsitektur sistem gudang data.

Untuk menentukan tingkat di mana baris melaporkan metrik, periksa `step_type` kolom `segment` dan:

- Jika keduanya `segment` dan `step_type` ada`-1`, maka baris melaporkan metrik di tingkat kueri.
- Jika `segment` tidak `-1` dan `step_type` sedang`-1`, maka baris melaporkan metrik di tingkat segmen.
- Jika `step_type` keduanya `segment` dan tidak`-1`, maka baris melaporkan metrik di tingkat langkah.

STV_QUERY_METRICS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_DETAIL. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang menjalankan kueri yang menghasilkan entri.
Kelas layanan	integer	ID untuk antrian kueri WLM (kelas layanan). Antrian kueri didefinisikan dalam konfigurasi WLM. Metrik dilaporkan hanya untuk antrian yang ditentukan pengguna.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
waktu mulai	timestamp	Waktu dalam UTC yang kueri mulai dijalankan, dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: `2009-06-12 11:29:19.131358` .
irisan	integer	Jumlah irisan untuk cluster.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen query dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses. Jika nilai segmen adalah -1, nilai segmen metrik digulung ke tingkat kueri.
tipe_langkah	integer	Jenis langkah yang berjalan. Untuk deskripsi jenis langkah, lihatJenis langkah.
baris	bigint	Jumlah baris yang diproses dengan satu langkah.
max_baris	bigint	Jumlah maksimum output baris untuk satu langkah, digabungkan di semua irisan.
CPU_waktu	bigint	Waktu CPU yang digunakan, dalam mikrodetik. Pada tingkat segmen, total waktu CPU untuk segmen di semua irisan. Pada tingkat kueri, jumlah waktu CPU untuk kueri di semua irisan dan segmen.
maks_cpu_ waktu_	bigint	Waktu CPU maksimum yang digunakan, dalam mikrodetik. Pada tingkat segmen, waktu CPU maksimum yang digunakan oleh segmen di semua irisan. Pada tingkat kueri, waktu CPU maksimum yang digunakan oleh segmen kueri apa pun.
blokir_baca	bigint	Jumlah blok 1 MB yang dibaca oleh kueri atau segmen.

Nama kolom	Jenis data	Deskripsi
max_blocks_read	bigint	Jumlah maksimum blok 1 MB yang dibaca oleh segmen, digabungkan di semua irisan. Pada tingkat segmen, jumlah maksimum blok 1 MB yang dibaca untuk segmen di semua irisan. Pada tingkat kueri, jumlah maksimum blok 1 MB dibaca oleh segmen kueri apa pun.
waktu_lari	bigint	Total waktu berjalan, dijumlahkan di seluruh irisan. Waktu berjalan tidak termasuk waktu tunggu. Pada tingkat segmen, waktu berjalan untuk segmen, dijumlahkan di semua irisan. Pada tingkat kueri, waktu berjalan untuk kueri dijumlahkan di semua irisan dan segmen. Karena nilai ini adalah jumlah, waktu berjalan tidak terkait dengan waktu eksekusi kueri.
waktu_lari maksimal	bigint	Waktu maksimum yang telah berlalu untuk segmen, dalam mikrodetik. Pada tingkat segmen, waktu proses maksimum untuk segmen di semua irisan. Pada tingkat kueri, waktu berjalan maksimum untuk segmen kueri apa pun.
max_blocks_to_disk	bigint	Jumlah maksimum ruang disk yang digunakan untuk menulis hasil perantara, dalam blok 1 MB. Pada tingkat segmen, jumlah maksimum ruang disk yang digunakan oleh segmen di semua irisan. Pada tingkat kueri, jumlah maksimum ruang disk yang digunakan oleh segmen kueri apa pun.
memblokir_ke_disk	bigint	Jumlah ruang disk yang digunakan oleh kueri atau segmen untuk menulis hasil perantara, dalam blok 1 MB.
langkah	integer	Langkah kueri yang berjalan.
maksimum_kueri_scan_ukuran	bigint	Ukuran maksimum data yang dipindai oleh kueri, dalam MB. Pada tingkat segmen, ukuran maksimum data yang dipindai oleh segmen di semua irisan. Pada tingkat kueri, ukuran maksimum data yang dipindai oleh segmen kueri apa pun.

Nama kolom	Jenis data	Deskripsi
kueri_scan_ukuran	bigint	Ukuran data yang dipindai oleh kueri, dalam MB.
Query_Prioritas	integer	Prioritas kueri. Nilai yang mungkin adalah -10,1,2,3,, dan4, yang -1 berarti bahwa prioritas kueri tidak didukung.
Query_Queue_Time	bigint	Jumlah waktu dalam mikrodetik saat kueri diantri.

Jenis langkah

Tabel berikut mencantumkan jenis langkah yang relevan dengan pengguna database. Tabel tidak mencantumkan jenis langkah yang hanya untuk penggunaan internal. Jika tipe langkah adalah -1, metrik tidak dilaporkan pada tingkat langkah.

Jenis langkah	Deskripsi
1	Pindai tabel
2	Sisipkan baris
3	Baris agregat
6	Urutkan langkah
7	Menggabungkan langkah
8	Langkah distribusi
9	Langkah distribusi siaran
10	Bergabunglah dengan hash
11	Gabungkan bergabung
12	Simpan langkah
14	Hash

Jenis langkah	Deskripsi
15	Gabung loop bersarang
16	Bidang dan ekspresi proyek
17	Batasi jumlah baris yang dikembalikan
18	Unik
20	Hapus baris
26	Batasi jumlah baris yang diurutkan yang dikembalikan
29	Hitung fungsi jendela
32	UDF
33	Unik
37	Kembalikan baris dari node pemimpin ke klien
38	Kembalikan baris dari node komputasi ke node pemimpin
40	Pemindaian spektrum.

Contoh kueri

Untuk menemukan kueri aktif dengan waktu CPU tinggi (lebih dari 1.000 detik), jalankan kueri berikut.

```
select query, cpu_time / 1000000 as cpu_seconds
from stv_query_metrics where segment = -1 and cpu_time > 1000000000
order by cpu_time;

query | cpu_seconds
------+------------
25775 |        9540
```

Untuk menemukan kueri aktif dengan penggabungan loop bersarang yang mengembalikan lebih dari satu juta baris, jalankan kueri berikut.

```
select query, rows
from stv_query_metrics
where step_type = 15 and rows > 1000000
order by rows;

query | rows
------+-----------
25775 | 1580225854
```

Untuk menemukan kueri aktif yang telah berjalan selama lebih dari 60 detik dan telah menggunakan kurang dari 10 detik waktu CPU, jalankan kueri berikut.

```
select query, run_time/1000000 as run_time_seconds
from stv_query_metrics
where segment = -1 and run_time > 60000000 and cpu_time < 10000000;

query | run_time_seconds
------+-----------------
25775 |              114
```

STV_TERBARU

Gunakan tabel STV_RECENTS untuk mengetahui informasi tentang kueri yang saat ini aktif dan baru-baru ini dijalankan terhadap database.

STV_RECENTS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Pemecahan masalah dengan STV_RECENTS

STV_RECENTS sangat membantu untuk menentukan apakah kueri atau kumpulan kueri saat ini sedang berjalan atau selesai. Ini juga menunjukkan durasi kueri telah berjalan. Ini sangat membantu untuk mengetahui kueri mana yang berjalan lama.

Anda dapat menggabungkan STV_RECENTS ke tampilan sistem lain, seperti STV_INFLIGHT, untuk mengumpulkan metadata tambahan tentang menjalankan kueri. (Ada contoh yang menunjukkan cara melakukan ini di bagian kueri sampel.) Anda juga dapat menggunakan catatan yang dikembalikan dari tampilan ini bersama dengan fitur pemantauan di konsol Amazon Redshift untuk pemecahan masalah secara real time.

Tampilan sistem yang melengkapi STV_RECENTS termasuk STL_QUERYTEXT, yang mengambil teks kueri untuk perintah SQL, dan, yang menggabungkan STV_INFLIGHT ke STL_QU SVV_QUERY_DALAM PENERBANGAN ERYTEXT.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
status	karakter (20)	Status kueri. Nilai yang valid adalah **`Running`**,**`Done`**.
waktu mulai	timestamp	Waktu permintaan dimulai.
durasi	integer	Jumlah mikrodetik sejak sesi dimulai.
user_name	karakter (50)	Nama pengguna yang menjalankan proses.
db_name	karakter (50)	Nama basis data.
kueri	karakter (600)	Teks kueri, hingga 600 karakter. Setiap karakter tambahan terpotong.
pid	integer	ID proses untuk sesi yang terkait dengan kueri, yang selalu -1 untuk kueri yang telah selesai.

Kueri Sampel

Untuk menentukan kueri mana yang sedang berjalan terhadap database, jalankan kueri berikut:

```
select user_name, db_name, pid, query
from stv_recents
```

```
where status = 'Running';
```

Contoh output di bawah ini menunjukkan satu kueri yang berjalan pada database TICKIT:

```
user_name | db_name |   pid   | query
----------+---------+---------+-------------
dwuser    | tickit  |  19996  |select venuename, venueseats from
venue where venueseats > 50000 order by venueseats desc;
```

Contoh berikut mengembalikan daftar kueri (jika ada) yang sedang berjalan atau menunggu dalam antrian untuk dijalankan:

```
select * from stv_recents where status<>'Done';

status |    starttime        | duration |user_name|db_name| query     | pid
-------+---------------------+----------+---------+-------+-----------+------
Running| 2010-04-21 16:11... | 281566454| dwuser  |tickit | select ...| 23347
```

Kueri ini tidak mengembalikan hasil kecuali Anda menjalankan sejumlah kueri bersamaan dan beberapa kueri tersebut berada dalam antrian.

Contoh berikut memperluas contoh sebelumnya. Dalam hal ini, kueri yang benar-benar “dalam penerbangan” (berjalan, tidak menunggu) dikecualikan dari hasil:

```
select * from stv_recents where status<>'Done'
and pid not in (select pid from stv_inflight);
...
```

Untuk kiat selengkapnya tentang pemecahan masalah kinerja kueri, lihatPemecahan masalah kueri.

STV_SESSION

Gunakan tabel STV_SESSIONS untuk melihat informasi tentang sesi pengguna aktif untuk Amazon Redshift.

Untuk melihat riwayat sesi, gunakan STL_SESSIONS tabel, bukan STV_SESSIONS.

STV_SESSIONS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_SESSION_RIWAYAT. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu mulai	timestamp	Waktu sesi dimulai.
proses	integer	ID proses untuk sesi.
user_name	karakter (50)	Pengguna yang terkait dengan sesi.
db_name	karakter (50)	Nama database yang terkait dengan sesi.
batas waktu_detik	int	Waktu maksimum dalam detik sesi tetap tidak aktif atau idle sebelum waktu habis. 0 menunjukkan bahwa tidak ada batas waktu yang ditetapkan.

Kueri Sampel

Untuk melakukan pemeriksaan cepat untuk melihat apakah ada pengguna lain yang saat ini masuk ke Amazon Redshift, ketik kueri berikut:

```
select count(*)
from stv_sessions;
```

Jika hasilnya lebih besar dari satu, maka setidaknya satu pengguna lain saat ini masuk ke database.

Untuk melihat semua sesi aktif untuk Amazon Redshift, ketik kueri berikut:

```
select *
from stv_sessions;
```

Hasil berikut menunjukkan empat sesi aktif yang berjalan di Amazon Redshift:

```
       starttime         | process |user_name                   | db_name
      | timeout_sec
-------------------------+---------+----------------------------
+---------------------------+-------------
  2018-08-06 08:44:07.50 |   13779 | IAMA:aws_admin:admin_grp   | dev
      | 0
  2008-08-06 08:54:20.50 |   19829 | dwuser                     | dev
      | 120
  2008-08-06 08:56:34.50 |   20279 | dwuser                     | dev
      | 120
  2008-08-06 08:55:00.50 |   19996 | dwuser                     | tickit
      | 0
(3 rows)
```

Nama pengguna yang diawali dengan IAMA menunjukkan bahwa pengguna masuk menggunakan sistem masuk tunggal federasi. Untuk informasi selengkapnya, lihat Menggunakan otentikasi IAM untuk menghasilkan kredenSIAL pengguna database.

STV_SLICE

Gunakan tabel STV_SLICES untuk melihat pemetaan irisan saat ini ke node.

Informasi dalam STV_SLICES digunakan terutama untuk tujuan investigasi.

STV_SLICES dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
simpul	integer	Node cluster tempat irisan berada.
mengiris	integer	Irisan simpul.
Slice lokal	integer	Informasi ini hanya untuk penggunaan internal.

Nama kolom	Jenis data	Deskripsi
jenis	karakter (1)	Informasi ini hanya untuk penggunaan internal.

Contoh kueri

Untuk melihat node cluster mana yang mengelola irisan mana, ketik kueri berikut:

```
select node, slice from stv_slices;
```

Kueri ini mengembalikan output sampel berikut:

```
 node | slice
------+-------
    0 |     2
    0 |     3
    0 |     1
    0 |     0
(4 rows)
```

STV_STARTUP_RECOVERY_STATE

Merekam keadaan tabel yang terkunci sementara selama operasi restart cluster. Amazon Redshift menempatkan kunci sementara pada tabel saat sedang diproses untuk menyelesaikan transaksi basi setelah restart cluster.

STV_STARTUP_RECOVERY_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
db_id	integer	ID Basis Data.

Nama kolom	Jenis data	Deskripsi
tabel_id	integer	ID tabel.
table_name	karakter (137)	Nama tabel.

Kueri Sampel

Untuk memantau tabel mana yang terkunci sementara, jalankan kueri berikut setelah restart cluster.

```
select * from STV_STARTUP_RECOVERY_STATE;

  db_id | tbl_id | table_name
--------+--------+------------
 100044 | 100058 | lineorder
 100044 | 100068 | part
 100044 | 100072 | customer
 100044 | 100192 | supplier
(4 rows)
```

STV_TBL_PERM

Tabel STV_TBL_PERM berisi informasi tentang tabel permanen di Amazon Redshift, termasuk tabel sementara yang dibuat oleh pengguna untuk sesi saat ini. STV_TBL_PERM berisi informasi untuk semua tabel di semua database.

Tabel ini berbeda dariSTV_TBL_TRANS, yang berisi informasi tentang tabel database transien yang dibuat sistem selama pemrosesan kueri.

STV_TBL_PERM hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Potongan simpul dialokasikan ke tabel.

Nama kolom	Jenis data	Deskripsi
id	integer	ID tabel.
name	karakter (72)	Nama tabel.
baris	bigint	Jumlah baris data dalam irisan.
baris yang diurutkan	bigint	Jumlah baris dalam irisan yang sudah diurutkan pada disk. Jika nomor ini tidak cocok dengan nomor ROWS, bersihkan tabel untuk menggunakan baris.
pekerja sementara	integer	Apakah tabel tersebut adalah tabel sementara atau tidak. 0 = salah; 1 = benar.
db_id	integer	ID database tempat tabel dibuat.
insert_pristine	integer	Untuk penggunaan internal.
delete_pristine	integer	Untuk penggunaan internal.
pencadangan	integer	Nilai yang menunjukkan apakah tabel disertakan dalam snapshot cluster. 0 = tidak; 1 = ya. Untuk informasi selengkapnya, lihat BACKUP parameter untuk perintah CREATE TABLE.
dist_style	integer	Gaya distribusi tabel tempat irisan tersebut dimiliki. Untuk informasi tentang nilai-nilai, lihatMelihat gaya distribusi. Untuk informasi tentang gaya distribusi, lihatGaya distribusi.
hitung_blok	integer	Jumlah blok yang digunakan oleh irisan. Nilainya adalah -1 ketika jumlah blok tidak dapat dihitung.

Kueri Sampel

Kueri berikut mengembalikan daftar ID tabel dan nama yang berbeda:

```
select distinct id, name
from stv_tbl_perm order by name;

   id   |          name
--------+------------------------
 100571 | category
 100575 | date
 100580 | event
 100596 | listing
 100003 | padb_config_harvest
 100612 | sales
...
```

Tabel sistem lain menggunakan ID tabel, jadi mengetahui ID tabel mana yang sesuai dengan tabel tertentu bisa sangat berguna. Dalam contoh ini, SELECT DISTINCT digunakan untuk menghapus duplikat (tabel didistribusikan di beberapa irisan).

Untuk menentukan jumlah blok yang digunakan oleh setiap kolom dalam tabel VENUE, ketik kueri berikut:

```
select col, count(*)
from stv_blocklist, stv_tbl_perm
where stv_blocklist.tbl = stv_tbl_perm.id
and stv_blocklist.slice = stv_tbl_perm.slice
and stv_tbl_perm.name = 'venue'
group by col
order by col;

 col | count
-----+-------
   0 |     8
   1 |     8
   2 |     8
   3 |     8
   4 |     8
   5 |     8
   6 |     8
   7 |     8
(8 rows)
```

Catatan penggunaan

Kolom ROWS menyertakan jumlah baris yang dihapus yang belum disedot (atau telah disedot tetapi dengan opsi SORT ONLY). Oleh karena itu, SUM kolom ROWS di tabel STV_TBL_PERM mungkin tidak cocok dengan hasil COUNT (*) saat Anda menanyakan tabel tertentu secara langsung. Misalnya, jika 2 baris dihapus dari VENUE, hasil COUNT (*) adalah 200 tetapi hasil SUM (ROWS) masih 202:

```
delete from venue
where venueid in (1,2);

select count(*) from venue;
count
-------
200
(1 row)

select trim(name) tablename, sum(rows)
from stv_tbl_perm where name='venue' group by name;

tablename | sum
-----------+-----
venue     | 202
(1 row)
```

Untuk menyinkronkan data di STV_TBL_PERM, jalankan vakum penuh tabel VENUE.

```
vacuum venue;

select trim(name) tablename, sum(rows)
from stv_tbl_perm
where name='venue'
group by name;

tablename | sum
-----------+-----
venue     | 200
(1 row)
```

STV_TBL_TRANS

Gunakan tabel STV_TBL_TRANS untuk mengetahui informasi tentang tabel database transien yang saat ini ada dalam memori.

Tabel transien biasanya kumpulan baris sementara yang digunakan sebagai hasil perantara saat kueri berjalan. STV_TBL_TRANS berbeda dari karena STV_TBL_PERM STV_TBL_PERM berisi informasi tentang tabel database permanen.

STV_TBL_TRANS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Potongan simpul dialokasikan ke tabel.
id	integer	ID tabel.
baris	bigint	Jumlah baris data dalam tabel.
ukuran	bigint	Jumlah byte yang dialokasikan ke tabel.
Query_id	bigint	ID kueri.
ref_cnt	integer	Jumlah referensi.
dari_dita ngguhkan	integer	Apakah tabel ini dibuat selama kueri yang sekarang ditangguhkan.
prep_swap	integer	Apakah tabel transien ini siap untuk ditukar ke disk jika diperlukan. (Swap hanya akan terjadi dalam situasi di mana memori rendah.)

Kueri Sampel

Untuk melihat informasi tabel sementara untuk kueri dengan ID kueri 90, ketik perintah berikut:

```
select slice, id, rows, size, query_id, ref_cnt
```

```
from stv_tbl_trans
where query_id = 90;
```

Kueri ini mengembalikan informasi tabel transien untuk query 90, seperti yang ditunjukkan pada output sampel berikut:

```
slice | id | rows | size | query_ | ref_ | from_     | prep_
      |    |      |      | id     | cnt  | suspended | swap
------+----+------+------+--------+------+-----------+-------
 1013 | 95 |    0 |    0 |     90 |    4 |         0 |   0
    7 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   10 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   17 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   14 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    3 | 96 |    0 |    0 |     90 |    4 |         0 |   0
 1013 | 99 |    0 |    0 |     90 |    4 |         0 |   0
    9 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    5 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   19 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    2 | 96 |    0 |    0 |     90 |    4 |         0 |   0
 1013 | 98 |    0 |    0 |     90 |    4 |         0 |   0
   13 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    1 | 96 |    0 |    0 |     90 |    4 |         0 |   0
 1013 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    6 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   11 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   15 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   18 | 96 |    0 |    0 |     90 |    4 |         0 |   0
```

Dalam contoh ini, Anda dapat melihat bahwa data kueri melibatkan tabel 95, 96, dan 98. Karena nol byte dialokasikan ke tabel ini, kueri ini dapat berjalan di memori.

STV_WLM_CLASSICATION_CONFIG

Berisi aturan klasifikasi saat ini untuk WLM.

STV_WLM_CLASSIFICATION_CONFIG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
id	integer	ID kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.
ketentuan	karakter (128)	Kondisi kueri.
aksi_seq	integer	Dicadangkan untuk penggunaan sistem.
tindakan	karakter (64)	Dicadangkan untuk penggunaan sistem.
kelas_tindakan_layanan_layanan	integer	Kelas layanan tempat tindakan berlangsung.

Contoh kueri

```
select * from STV_WLM_CLASSIFICATION_CONFIG;

id | condition                                    | action_seq | action |
 action_service_class
---+----------------------------------------------+------------+--------
+---------------------
 1 | (system user) and (query group: health)     |          0 | assign |
     1
 2 | (system user) and (query group: metrics)    |          0 | assign |
     2
 3 | (system user) and (query group: cmstats)    |          0 | assign |
     3
 4 | (system user)                               |          0 | assign |
     4
 5 | (super user) and (query group: superuser)   |          0 | assign |
     5
 6 | (query group: querygroup1)                  |          0 | assign |
     6
 7 | (user group: usergroup1)                    |          0 | assign |
     6
 8 | (user group: usergroup2)                    |          0 | assign |
     7
```

```
 9 | (query group: querygroup3)                  |          0 | assign |
     8
10 | (query group: querygroup4)                  |          0 | assign |
     9
11 | (user group: usergroup4)                    |          0 | assign |
     9
12 | (query group: querygroup*)                  |          0 | assign |
    10
13 | (user group: usergroup*)                    |          0 | assign |
    10
14 | (querytype: any)                            |          0 | assign |
    11
(4 rows)
```

STV_WLM_QMR_CONFIG

Merekam konfigurasi untuk aturan pemantauan kueri WLM (QMR). Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

STV_WLM_QMR_CONFIG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kelas_layanan	integer	ID untuk antrian kueri WLM (kelas layanan). Antrian kueri didefinisikan dalam konfigurasi WLM. Aturan dapat ditentukan hanya untuk antrian yang ditentukan pengguna. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.
nama aturan	karakter (256)	Nama aturan pemantauan kueri.
tindakan	karakter (256)	Tindakan aturan. Nilai yang mungkin adalah `log`, `hop`, `abort`, dan `change_query_priority` .
nama_metrik	karakter (256)	Nama metrik.
operator_metrik	karakter (256)	Operator metrik. Nilai yang mungkin adalah >, =, <.
nilai_metrik	double	Nilai ambang untuk metrik tertentu yang memicu tindakan.

Nama kolom	Jenis data	Deskripsi
nilai_tindakan	karakter (256)	Jika `action` ya`change_query_priority` , maka nilai yang mungkin adalah`highest`,`high`,`normal`,`low`, dan`lowest`. Jika `action` ya `loghop`,, `abort` atau nilainya kosong.

Contoh kueri

Untuk melihat definisi aturan QMR untuk semua kelas layanan yang lebih besar dari 5 (yang mencakup antrian yang ditentukan pengguna), jalankan kueri berikut. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

```
Select *
from stv_wlm_qmr_config
where service_class > 5
order by service_class;
```

STV_WLM_QUERY_QUEUE_STATE

Merekam status antrian kueri saat ini untuk kelas layanan.

STV_WLM_QUERY_QUEUE_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kelas_layanan	integer	ID untuk kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

Nama kolom	Jenis data	Deskripsi
posisi	integer	Posisi kueri dalam antrian. Kueri dengan **`position`** nilai terkecil berjalan berikutnya.
tugas	integer	ID digunakan untuk melacak kueri melalui pengelola beban kerja. Dapat dikaitkan dengan beberapa ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
kueri	integer	ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
Slot_count	integer	Jumlah slot kueri WLM.
waktu mulai	timestamp	Waktu kueri memasuki antrian.
antrian_waktu	bigint	Jumlah mikrodetik yang kueri berada dalam antrian.

Contoh kueri

Kueri berikut menunjukkan kueri dalam antrian untuk kelas layanan yang lebih besar dari 4.

```
select * from stv_wlm_query_queue_state
where service_class > 4
order by service_class;
```

Kueri ini mengembalikan output sampel berikut.

```
 service_class | position | task | query | slot_count |        start_time          |
 queue_time
---------------+----------+------+-------+------------+----------------------------
+------------
             5 |        0 |  455 |   476 |          5 | 2010-10-06 13:18:24.065838 |
 20937257
```

```
             6 |        1 |  456 |   478 |         5 | 2010-10-06 13:18:26.652906 |
 18350191
(2 rows)
```

STV_WLM_QUERY_STATE

Merekam keadaan kueri saat ini yang dilacak oleh WLM.

STV_WLM_QUERY_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_HISTORY. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
xid	integer	ID transaksi dari query atau subquery.
tugas	integer	ID digunakan untuk melacak kueri melalui pengelola beban kerja. Dapat dikaitkan dengan beberapa ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
kueri	integer	ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
kelas_layanan	integer	ID untuk kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.
Slot_count	integer	Jumlah slot kueri WLM.
wlm_start_time	timestamp	Waktu kueri memasuki antrian tabel sistem atau antrian kueri pendek.

Nama kolom	Jenis data	Deskripsi
status	karakter (16)	Status kueri atau subkueri saat ini. Nilai yang mungkin adalah sebagai berikut: • `Classified` — Query telah ditetapkan ke kelas layanan. • `Completed` — Query selesai berjalan. Kueri berhasil berjalan atau dibatalkan. Untuk keadaan akhir, periksa hasilnyaSTL_QUERY. • `Dequeued`- Penggunaan internal saja. • `Evicted`— Kueri telah diusir dari kelas layanan untuk restart. • `Evicting`— Kueri sedang diusir dari kelas layanan untuk restart. • `Initialized` - Penggunaan internal saja. • `Invalid`- Penggunaan internal saja. • `Queued`— Kueri dikirim ke antrian kueri karena tidak ada slot yang tersedia untuk menjalankannya. • `QueuedWaiting` — Kueri sedang menunggu dalam antrian kueri. • `Rejected`- Penggunaan internal saja. • `Returning` — Query mengembalikan hasil ke klien. • `Running`— Query sedang berjalan. • `TaskAssigned` - Penggunaan internal saja.
antrian_waktu	bigint	Jumlah mikrodetik yang telah dihabiskan kueri dalam antrian.
waktu eksekusi	bigint	Jumlah mikrodetik yang telah dijalankan kueri.

Nama kolom	Jenis data	Deskripsi
kueri_prioritas	arang (20)	Prioritas kueri. Nilai yang mungkin adalah n/a`lowest`,`low`,`normal`,`high`,, dan`highest`, yang n/a berarti bahwa prioritas kueri tidak didukung.

Contoh kueri

Kueri berikut menampilkan semua kueri yang sedang dijalankan di kelas layanan lebih besar dari 4. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

```
select xid, query, trim(state) as state, queue_time, exec_time
from stv_wlm_query_state
where service_class > 4;
```

Kueri ini mengembalikan output sampel berikut:

```
xid    | query | state   | queue_time | exec_time
-------+-------+---------+------------+-----------
100813 | 25942 | Running |          0 |    1369029
100074 | 25775 | Running |          0 | 2221589242
```

STV_WLM_QUERY_TASKING_STATE

Berisi status tugas kueri kelas layanan saat ini.

STV_WLM_QUERY_TASK_STATE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kelas_layanan	integer	ID untuk kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

Nama kolom	Jenis data	Deskripsi
tugas	integer	ID digunakan untuk melacak kueri melalui manajer beban kerja. Dapat dikaitkan dengan beberapa ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
kueri	integer	ID kueri. Jika kueri dimulai ulang, kueri akan diberi ID kueri baru tetapi bukan ID tugas baru.
Slot_count	integer	Jumlah slot kueri WLM.
waktu mulai	timestamp	Waktu kueri mulai dieksekusi.
waktu eksekusi	bigint	Jumlah mikrodetik yang telah dijalankan kueri.

Contoh kueri

Kueri berikut menampilkan status kueri saat ini di kelas layanan yang lebih besar dari 4. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

```
select * from stv_wlm_query_task_state
where service_class > 4;
```

Kueri ini mengembalikan output sampel berikut:

```
service_class | task | query |         start_time         | exec_time
--------------+------+-------+----------------------------+-----------
    5         |  466 |   491 | 2010-10-06 13:29:23.063787 | 357618748
(1 row)
```

STV_WLM_SERVICE_CLASS_CONFIG

Merekam konfigurasi kelas layanan untuk WLM.

STV_WLM_SERVICE_CLASS_CONFIG hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kelas_layanan	integer	ID untuk kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.
antrian_strategi	karakter (32)	Dicadangkan untuk penggunaan sistem.
nama_kueri_tugas	integer	Tingkat konkurensi aktual saat ini dari kelas layanan. Jika `num_query_tasks` dan `target_num_query_tasks` berbeda, transisi WLM dinamis sedang dalam proses. Nilai `-1` menunjukkan bahwa Auto WLM dikonfigurasi.
target_num_kueri_tugas	integer	Level konkurensi ditetapkan oleh perubahan konfigurasi WLM terbaru.
dapat diusir	karakter (8)	Dicadangkan untuk penggunaan sistem.
ambang pengusiran	bigint	Dicadangkan untuk penggunaan sistem.
Query_Working_Mem	integer	Jumlah aktual memori kerja saat ini, dalam MB per slot, per node, ditetapkan ke kelas layanan. Jika `query_working_mem` dan `target_query_working_mem` berbeda, transisi WLM dinamis sedang dalam proses. Nilai `-1` menunjukkan bahwa Auto WLM dikonfigurasi.
target_query_working_mem	integer	Jumlah memori kerja, dalam MB per slot, per node, ditetapkan oleh perubahan konfigurasi WLM terbaru.
min_step_mem	integer	Dicadangkan untuk penggunaan sistem.
name	karakter (64)	Nama kelas layanan.
waktu_eksekusi maksimal	bigint	Jumlah milidetik yang dapat dijalankan kueri sebelum dihentikan.

Nama kolom	Jenis data	Deskripsi
user_group_wild_card	Boolean	JikaTRUE, antrian WLM memperlakukan tanda bintang (*) sebagai karakter wildcard dalam string grup pengguna dalam konfigurasi WLM.
Query_group_wild_card	Boolean	JikaTRUE, antrian WLM memperlakukan tanda bintang (*) sebagai karakter wildcard dalam string grup kueri dalam konfigurasi WLM.
penskalaan_concurrency_scale	karakter (20)	Menjelaskan apakah penskalaan konkurensi adalah on atauoff.
Query_Prioritas	karakter (20)	Nilai prioritas kueri.
user_role_wild_card	Boolean	JikaTRUE, antrian WLM memperlakukan tanda bintang (*) sebagai karakter wildcard dalam string pengguna pengguna dalam konfigurasi WLM.

Contoh kueri

Kelas layanan pertama yang ditentukan pengguna adalah kelas layanan 6, yang diberi nama Service class #1. Kueri berikut menampilkan konfigurasi saat ini untuk kelas layanan yang lebih besar dari 4. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

```
select rtrim(name) as name,
num_query_tasks as slots,
query_working_mem as mem,
max_execution_time as max_time,
user_group_wild_card as user_wildcard,
query_group_wild_card as query_wildcard
from stv_wlm_service_class_config
where service_class > 4;

name                         | slots | mem | max_time | user_wildcard | query_wildcard
-----------------------------+-------+-----+----------+---------------+---------------
Service class for super user |     1 | 535 |        0 | false         | false
Queue 1                      |     5 | 125 |        0 | false         | false
Queue 2                      |     5 | 125 |        0 | false         | false
```

```
Queue 3                    |     5 | 125 |        0 | false          | false
Queue 4                    |     5 | 627 |        0 | false          | false
Queue 5                    |     5 | 125 |        0 | true           | true
Default queue              |     5 | 125 |        0 | false          | false
```

Kueri berikut menunjukkan status transisi WLM dinamis. Sementara transisi sedang dalam proses, `num_query_tasks` dan `target_query_working_mem` diperbarui sampai mereka sama dengan nilai target. Untuk informasi selengkapnya, lihat Properti konfigurasi dinamis dan statis WLM.

```
select rtrim(name) as name,
num_query_tasks as slots,
target_num_query_tasks as target_slots,
query_working_mem as memory,
target_query_working_mem as target_memory
from stv_wlm_service_class_config
where num_query_tasks > target_num_query_tasks
or query_working_mem > target_query_working_mem
and service_class > 5;

 name             | slots | target_slots | memory | target_mem
------------------+-------+--------------+--------+------------
 Queue 3          |     5 |           15 |    125 |       375
 Queue 5          |    10 |            5 |    250 |       125
 (2 rows)
```

STV_WLM_SERVICE_CLASS_STATE

Berisi status kelas layanan saat ini.

STV_WLM_SERVICE_CLASS_STATE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kelas_layanan	integer	ID untuk kelas layanan. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.
nom_queued_query	integer	Jumlah kueri saat ini dalam antrian.

Nama kolom	Jenis data	Deskripsi
nom_eksekusing_query	integer	Jumlah query yang sedang dijalankan.
nom_serviced_query	integer	Jumlah kueri yang pernah ada di kelas layanan.
nom_query yang dieksekusi	integer	Jumlah kueri yang telah berjalan sejak Amazon Redshift dimulai ulang.
nama_kueri yang diusir	integer	Jumlah kueri yang telah diusir sejak Amazon Redshift dimulai ulang. Beberapa alasan kueri yang diusir termasuk batas waktu WLM, tindakan hop QMR, dan kueri gagal pada cluster penskalaan konkurensi.
nom_concurrency_scaling_query	integer	Jumlah kueri yang dijalankan pada cluster penskalaan konkurensi sejak Amazon Redshift dimulai ulang.

Contoh kueri

Kueri berikut menampilkan status untuk kelas layanan yang lebih besar dari 5. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

```
select service_class, num_executing_queries,
num_executed_queries
from stv_wlm_service_class_state
where service_class > 5
order by service_class;
```

```
 service_class | num_executing_queries | num_executed_queries
---------------+-----------------------+----------------------
             6 |                     1 |                  222
             7 |                     0 |                  135
             8 |                     1 |                   39
(3 rows)
```

STV_XRESTORE_ALTER_QUEUE_STATE

Gunakan STV_XRESTORE_ALTER_QUEUE_STATE untuk memantau kemajuan migrasi setiap tabel selama perubahan ukuran klasik. Ini secara khusus berlaku ketika tipe node target adalah RG atau RA3. Untuk informasi selengkapnya tentang mengubah ukuran klasik ke node RG atau RA3, buka perubahan ukuran klasik. https://docs.aws.amazon.com/redshift/latest/mgmt/managing-cluster-operations.html#classic-resize-faster

STV_XRESTORE_ALTER_QUEUE_STATE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_RESTORE_STATE. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang memulai perubahan ukuran.
db_id	integer	ID database.
skema	karakter (128)	Nama skema.
table_name	karakter (128)	Nama tabel.
tbl	integer	ID tabel.
status	karakter (64)	Status kemajuan migrasi tabel. Nilai yang mungkin adalah sebagai berikut. • `Waiting`: Menunggu redistribusi dimulai • `Applying`: Saat ini mendistribusikan kembali • `Finished`: Selesai mendistribusikan ulang

Nama kolom	Jenis data	Deskripsi
task_type	integer	Jenis redistribusi untuk tabel. Nilai yang mungkin adalah sebagai berikut. • 1: KUNCI • 2: GENAP Untuk informasi selengkapnya tentang gaya distribusi, lihatGaya distribusi.

Contoh kueri

Kueri berikut menunjukkan jumlah tabel dalam database yang menunggu untuk diubah ukurannya, sedang diubah ukurannya, dan selesai mengubah ukuran.

```
select db_id, status, count(*)
from stv_xrestore_alter_queue_state
group by 1,2 order by 3 desc

db_id  |   status    | count
-------+-------------+------
694325 | Waiting     |   323
694325 | Finished    |    60
694325 | Applying    |     1
```

Tampilan SVCS untuk cluster penskalaan utama dan konkurensi

Tampilan sistem SVCS dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tabel dengan awalan STL kecuali bahwa tabel STL menyediakan informasi hanya untuk kueri yang dijalankan pada cluster utama.

Topik

- SVCS_ALERT_EVENT_LOG
- SVCS_KOMPILASI
- SVCS_CONCURRENCY_SCALING_PEMAKAIAN

- SVCS_JELASKAN
- SVCS_PLAN_INFO
- SVCS_QUERY_RINGKASAN
- SVCS_S3LIST
- SVCS_S3LOG
- SVCS_S3PARTITION_RINGKASAN
- SVCS_S3QUERY_RINGKASAN
- SVCS_STREAM_SEGS
- SVCS_UNLOAD_LOG

SVCS_ALERT_EVENT_LOG

Merekam peringatan saat pengoptimal kueri mengidentifikasi kondisi yang mungkin menunjukkan masalah kinerja. Tampilan ini berasal dari tabel sistem STL_ALERT_EVENT_LOG tetapi tidak menampilkan tingkat irisan untuk kueri yang dijalankan pada cluster penskalaan konkurensi. Gunakan tabel SVCS_ALERT_EVENT_LOG untuk mengidentifikasi peluang untuk meningkatkan kinerja kueri.

Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Untuk informasi selengkapnya, lihat Pemrosesan kueri.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tabel dengan awalan STL kecuali bahwa tabel STL menyediakan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_ALERT_EVENT_LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Langkah kueri yang berjalan.
pid	integer	ID proses yang terkait dengan pernyataan dan irisan. Kueri yang sama mungkin memiliki beberapa PID jika berjalan pada beberapa irisan.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
kejadian	karakter (1024)	Deskripsi peristiwa peringatan.
solusi	karakter (1024)	Solusi yang disarankan.
waktu_acara	timestamp	Waktu dalam UTC saat kueri dimulai. Total waktu termasuk antrian dan eksekusi. dengan 6 digit presisi untuk detik pecahan. Sebagai contoh: **`2009-06-12 11:29:19.131358`** .

Catatan penggunaan

Anda dapat menggunakan SVCS_ALERT_EVENT_LOG untuk mengidentifikasi masalah potensial dalam kueri Anda, lalu ikuti praktik untuk mengoptimalkan desain database Anda dan menulis ulang Penyetelan kinerja kueri kueri Anda. SVCS_ALERT_EVENT_LOG mencatat peringatan berikut:

- Statistik hilang

Statistik hilang. Jalankan ANALISIS berikut pemuatan data atau pembaruan signifikan dan gunakan STATUPDATE dengan operasi COPY. Untuk informasi selengkapnya, lihat Praktik terbaik Amazon Redshift untuk merancang kueri.

- Loop bersarang

  Loop bersarang biasanya merupakan produk Cartesian. Evaluasi kueri Anda untuk memastikan bahwa semua tabel yang berpartisipasi digabungkan secara efisien.

- Filter yang sangat selektif

  Rasio baris yang dikembalikan ke baris yang dipindai kurang dari 0,05. Baris yang dipindai adalah nilai `rows_pre_user_filter` dan baris yang dikembalikan adalah nilai baris dalam tabel STL_SCAN sistem. Menunjukkan bahwa kueri memindai jumlah baris yang luar biasa besar untuk menentukan kumpulan hasil. Hal ini dapat disebabkan oleh kunci pengurutan yang hilang atau salah. Untuk informasi selengkapnya, lihat Urutkan kunci.

- Baris hantu yang berlebihan

  Pemindaian melewatkan sejumlah besar baris yang ditandai sebagai dihapus tetapi tidak disedot, atau baris yang telah dimasukkan tetapi tidak dilakukan. Untuk informasi selengkapnya, lihat Meja penyedot debu.

- Distribusi besar

  Lebih dari 1.000.000 baris didistribusikan kembali untuk gabungan hash atau agregasi. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

- Siaran besar

  Lebih dari 1.000.000 baris disiarkan untuk hash join. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

- Eksekusi serial

  Gaya redistribusi DS_DIST_ALL_INNER ditunjukkan dalam rencana kueri, yang memaksa eksekusi serial karena seluruh tabel bagian dalam didistribusikan kembali ke satu node. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

Kueri Sampel

Kueri berikut menunjukkan peristiwa peringatan untuk empat kueri.

```
SELECT query, substring(event,0,25) as event,
substring(solution,0,25) as solution,
trim(event_time) as event_time from svcs_alert_event_log order by query;

 query |             event             |          solution            |     event_time

-------+-------------------------------+-------------------------------
+---------------------
  6567 | Missing query planner statist | Run the ANALYZE command       | 2014-01-03
 18:20:58
  7450 | Scanned a large number of del | Run the VACUUM command to rec| 2014-01-03
 21:19:31
  8406 | Nested Loop Join in the query | Review the join predicates to| 2014-01-04
 00:34:22
 29512 | Very selective query filter:r | Review the choice of sort key| 2014-01-06
 22:00:00

(4 rows)
```

SVCS_KOMPILASI

Merekam waktu dan lokasi kompilasi untuk setiap segmen kueri, termasuk kueri yang dijalankan pada cluster penskalaan serta kueri yang dijalankan pada cluster utama.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tampilan dengan awalan SVL kecuali bahwa tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_COMPILE terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang SCL_COMPILE, lihat. SVL_COMPILE

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
pid	integer	ID proses yang terkait dengan pernyataan.
kueri	integer	ID kueri. Anda dapat menggunakan ID ini untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
segmen	integer	Segmen query yang akan dikompilasi.
lokus	integer	Lokasi di mana segmen berjalan, **1** jika pada node komputasi dan **2** jika pada node pemimpin.
waktu mulai	timestamp	Waktu dalam Waktu Terkoordinasi Universal (UTC) dimana kompilasi dimulai.
waktu akhir	timestamp	Waktu dalam UTC saat kompilasi berakhir.
mengompilasikan	integer	Nilai yaitu **0** jika kompilasi digunakan kembali dan **1** jika segmen dikompilasi.

Kueri Sampel

Dalam contoh ini, query 35878 dan 35879 menjalankan pernyataan SQL yang sama. Kolom kompilasi untuk query 35878 menunjukkan 1 untuk empat segmen kueri, yang menunjukkan bahwa segmen dikompilasi. Query 35879 ditampilkan `0` di kolom kompilasi untuk setiap segmen, menunjukkan bahwa segmen tidak perlu dikompilasi lagi.

```
select userid, xid,  pid, query, segment, locus,
datediff(ms, starttime, endtime) as duration, compile
from svcs_compile
where query = 35878 or query = 35879
order by query, segment;
```

```
 userid |  xid   |  pid  | query | segment | locus | duration | compile
--------+--------+-------+-------+---------+-------+----------+---------
    100 | 112780 | 23028 | 35878 |       0 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       1 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       2 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       3 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       4 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       5 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       6 |     1 |     1380 |       1
    100 | 112780 | 23028 | 35878 |       7 |     1 |     1085 |       1
    100 | 112780 | 23028 | 35878 |       8 |     1 |     1197 |       1
    100 | 112780 | 23028 | 35878 |       9 |     2 |      905 |       1
    100 | 112782 | 23028 | 35879 |       0 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       1 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       2 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       3 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       4 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       5 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       6 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       7 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       8 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       9 |     2 |        0 |       0
(20 rows)
```

SVCS_CONCURRENCY_SCALING_PEMAKAIAN

Merekam periode penggunaan untuk penskalaan konkurensi. Setiap periode penggunaan adalah durasi berturut-turut di mana cluster penskalaan konkurensi individu secara aktif memproses kueri.

SVCS_CONCURRENCY_SCALING_USAGE Tabel ini terlihat oleh pengguna super. Superuser database dapat memilih untuk membukanya untuk semua pengguna.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
waktu mulai	stempel waktu tanpa zona waktu	Ketika periode penggunaan dimulai.

Nama kolom	Jenis data	Deskripsi
akhir_waktu	stempel waktu tanpa zona waktu	Ketika periode penggunaan berakhir.
pertanyaan	bigint	Jumlah kueri yang dijalankan selama periode penggunaan ini.
penggunaan_dalam_detik	numerik (27,0)	Total detik dalam periode penggunaan ini.

Kueri Sampel

Untuk melihat durasi penggunaan dalam detik untuk periode tertentu, masukkan kueri berikut:

```
select * from svcs_concurrency_scaling_usage order by start_time;

start_time | end_time | queries | usage_in_seconds
---------------------------+---------------------------+---------+-----------------
2019-02-14 18:43:53.01063 | 2019-02-14 19:16:49.781649 | 48 | 1977
```

SVCS_JELASKAN

Menampilkan rencana EXPLAIN untuk kueri yang telah dikirimkan untuk dieksekusi.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tabel dengan awalan STL kecuali bahwa tabel STL menyediakan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_EXPLAIN terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
nodeid	integer	Pengidentifikasi simpul rencana, di mana node memetakan ke satu atau lebih langkah dalam pelaksanaan kueri.
parental	integer	Rencanakan pengenal simpul untuk node induk. Node induk memiliki sejumlah node anak. Misalnya, gabungan gabungan adalah induk dari pemindaian pada tabel yang digabungkan.
plannode	karakter (400)	Teks simpul dari output EXPLAIN. Node rencana yang mengacu pada eksekusi pada node komputasi diawali dengan **XN** dalam output EXPLAIN.
info	karakter (400)	Informasi kualifikasi dan filter untuk simpul rencana. Misalnya, kondisi bergabung dan batasan klausa WHERE disertakan dalam kolom ini.

Kueri Sampel

Pertimbangkan output EXPLAIN berikut untuk kueri gabungan agregat:

```
explain select avg(datediff(day, listtime, saletime)) as avgwait
from sales, listing where sales.listid = listing.listid;
                                QUERY PLAN

------------------------------------------------------------------------------
 XN Aggregate  (cost=6350.30..6350.31 rows=1 width=16)
  ->  XN Hash Join DS_DIST_NONE  (cost=47.08..6340.89 rows=3766 width=16)
```

```
        Hash Cond: ("outer".listid = "inner".listid)
        -> XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=12)
        -> XN Hash  (cost=37.66..37.66 rows=3766 width=12)
              -> XN Seq Scan on sales  (cost=0.00..37.66 rows=3766 width=12)
(6 rows)
```

Jika Anda menjalankan kueri ini dan ID kuerinya adalah 10, Anda dapat menggunakan tabel SVCS_EXPLAIN untuk melihat jenis informasi yang sama dengan yang dikembalikan perintah EXPLAIN:

```
select query,nodeid,parentid,substring(plannode from 1 for 30),
substring(info from 1 for 20) from svcs_explain
where query=10 order by 1,2;

query| nodeid |parentid|           substring            |    substring
-----+--------+--------+--------------------------------+------------------
10   |      1 |      0 |XN Aggregate  (cost=6717.61..6  |
10   |      2 |      1 |  -> XN Merge Join DS_DIST_NO| Merge Cond:("outer"
10   |      3 |      2 |       -> XN Seq Scan on lis |
10   |      4 |      2 |       -> XN Seq Scan on sal |
(4 rows)
```

Pertimbangkan kueri berikut:

```
select event.eventid, sum(pricepaid)
from event, sales
where event.eventid=sales.eventid
group by event.eventid order by 2 desc;

eventid |   sum
--------+----------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
...
```

Jika ID kueri ini adalah 15, kueri tabel sistem berikut mengembalikan node rencana yang dilakukan. Dalam hal ini, urutan node dibalik untuk menunjukkan urutan eksekusi yang sebenarnya:

```
select query,nodeid,parentid,substring(plannode from 1 for 56)
```

```
from svcs_explain where query=15 order by 1, 2 desc;

query|nodeid|parentid|                              substring
-----+------+--------+-------------------------------------------------------
15   |    8 |      7 |                                   -> XN Seq Scan on eve
15   |    7 |      5 |                              -> XN Hash(cost=87.98..87.9
15   |    6 |      5 |                              -> XN Seq Scan on sales(cos
15   |    5 |      4 |                         -> XN Hash Join DS_DIST_OUTER(cos
15   |    4 |      3 |                    -> XN HashAggregate(cost=862286577.07..
15   |    3 |      2 |           -> XN Sort(cost=1000862287175.47..10008622871
15   |    2 |      1 |  -> XN Network(cost=1000862287175.47..1000862287197.
15   |    1 |      0 |XN Merge(cost=1000862287175.47..1000862287197.46 rows=87
(8 rows)
```

Kueri berikut mengambil ID kueri untuk setiap paket kueri yang berisi fungsi jendela:

```
select query, trim(plannode) from svcs_explain
where plannode like '%Window%';

query|                                      btrim
-----+-------------------------------------------------------------------------
26   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
27   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
(2 rows)
```

SVCS_PLAN_INFO

Gunakan tabel SVCS_PLAN_INFO untuk melihat output EXPLAIN untuk kueri dalam hal sekumpulan baris. Ini adalah cara alternatif untuk melihat rencana kueri.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tabel dengan awalan STL kecuali bahwa tabel STL menyediakan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_PLAN_INFO terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
nodeid	integer	Pengidentifikasi simpul rencana, di mana node memetakan ke satu atau lebih langkah dalam pelaksanaan kueri.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.
langkah	integer	Nomor yang mengidentifikasi langkah kueri.
lokus	integer	Lokasi di mana langkah berjalan. 0 jika pada node komputasi dan 1 jika pada node pemimpin.
plannode	integer	Nilai yang disebutkan dari simpul rencana. Lihat tabel berikut untuk enum untuk plannode. (Kolom PLANNODE di SVCS_JELASKAN berisi teks simpul rencana.)
biaya awal	double precision	Perkiraan biaya relatif pengembalian baris pertama untuk langkah ini.
total biaya	double precision	Perkiraan biaya relatif untuk melaksanakan langkah.
baris	bigint	Perkiraan jumlah baris yang akan diproduksi secara bertahap.
byte	bigint	Perkiraan jumlah byte yang akan dihasilkan oleh langkah.

Kueri Sampel

Contoh berikut membandingkan rencana kueri untuk kueri SELECT sederhana yang dikembalikan dengan menggunakan perintah EXPLAIN dan dengan menanyakan tabel SVCS_PLAN_INFO.

```
explain select * from category;
QUERY PLAN
```

```
------------------------------------------------------------
XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
(1 row)

select * from category;
catid | catgroup | catname | catdesc
-------+----------+-----------+--------------------------------------------
1 | Sports | MLB | Major League Baseball
3 | Sports | NFL | National Football League
5 | Sports | MLS | Major League Soccer
...

select * from svcs_plan_info where query=256;

query | nodeid | segment | step | locus | plannode | startupcost | totalcost
| rows | bytes
-------+--------+---------+------+-------+----------+-------------+-----------+------
+-------
256 | 1 | 0 | 1 | 0 | 104 | 0 | 0.11 | 11 | 539
256 | 1 | 0 | 0 | 0 | 104 | 0 | 0.11 | 11 | 539
(2 rows)
```

Dalam contoh ini, PLANNODE 104 mengacu pada pemindaian berurutan dari tabel CATEGORY.

```
select distinct eventname from event order by 1;

eventname
----------------------------------------------------------------------------
.38 Special
3 Doors Down
70s Soul Jam
A Bronx Tale
...

explain select distinct eventname from event order by 1;

QUERY PLAN
------------------------------------------------------------------------------------------
XN Merge (cost=1000000000136.38..1000000000137.82 rows=576 width=17)
Merge Key: eventname
-> XN Network (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Send to leader
```

```
-> XN Sort (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Sort Key: eventname
-> XN Unique (cost=0.00..109.98 rows=576 width=17)
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=17)
(8 rows)

select * from svcs_plan_info where query=240 order by nodeid desc;

query | nodeid | segment | step | locus | plannode | startupcost |
totalcost | rows | bytes
-------+--------+---------+------+-------+----------+-----------------
+------------------+------+--------
240 | 5 | 0 | 0 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 5 | 0 | 1 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 4 | 0 | 2 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 0 | 3 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 0 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 1 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 3 | 1 | 2 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 3 | 2 | 0 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 2 | 2 | 1 | 0 | 123 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 1 | 3 | 0 | 0 | 122 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
(10 rows)
```

SVCS_QUERY_RINGKASAN

Gunakan tampilan SVCS_QUERY_SUMMARY untuk menemukan informasi umum tentang pelaksanaan kueri.

Perhatikan bahwa informasi dalam SVCS_QUERY_SUMMARY dikumpulkan dari semua node.

> **Note**
>
> Tampilan SVCS_QUERY_SUMMARY hanya berisi informasi tentang kueri yang diselesaikan oleh Amazon Redshift, bukan perintah utilitas dan DDL lainnya. Untuk daftar lengkap dan informasi tentang semua pernyataan yang diselesaikan oleh Amazon Redshift, termasuk perintah DDL dan utilitas, Anda dapat menanyakan tampilan SVL_STATEMENTTEXT. Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tampilan dengan awalan SVL

kecuali bahwa tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_QUERY_SUMMARY dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan pemantauan SYSSYS_QUERY_DETAIL. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk kueri Anda.

Untuk informasi tentang SVL_QUERY_SUMMARY, lihat. SVL_QUERY_SUMMARY

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
stm	integer	Stream: Satu set segmen bersamaan dalam kueri. Kueri memiliki satu atau lebih aliran.
seg	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen query dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses.
langkah	integer	Langkah kueri yang berjalan.
waktu maksimal	bigint	Jumlah waktu maksimum untuk langkah berjalan (dalam mikrodetik).
avgtime	bigint	Waktu rata-rata untuk langkah berjalan (dalam mikrodetik).
baris	bigint	Jumlah baris data yang terlibat dalam langkah kueri.

Nama kolom	Jenis data	Deskripsi
bytes	bigint	Jumlah byte data yang terlibat dalam langkah query.
rate_baris	double precision	Tingkat eksekusi kueri per baris.
rate_byte	double precision	Tingkat eksekusi kueri per byte.
label	text	Label langkah, yang terdiri dari nama langkah kueri dan, bila berlaku, ID tabel dan nama tabel (misalnya, scan tbl=100448 name =user). Three-digit ID tabel biasanya mengacu pada pemindaia n tabel transien. Ketika Anda melihat `tbl=0`, biasanya mengacu pada pemindaian nilai konstan.
berbasis-disk	karakter (1)	Apakah langkah kueri ini dilakukan sebagai operasi berbasis disk pada node mana pun di cluster: true (**t**) atau false (**f**). Hanya langkah-langkah tertentu, seperti langkah hash, sort, dan agregat, yang dapat masuk ke disk. Banyak jenis langkah selalu dijalankan dalam memori.
workmem	bigint	Jumlah memori kerja (dalam byte) yang ditetapkan untuk langkah kueri.
is_rrscan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian terbatas rentang digunakan pada langkah tersebut. Defaultnya adalah false (**f**).
is_tertun da_scan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian tertunda digunakan pada langkah tersebut. Defaultnya adalah false (**f**).
baris pra_filter	bigint	Untuk pemindaian tabel permanen, jumlah total baris yang dipancarkan sebelum memfilter baris ditandai untuk dihapus (baris hantu).

Kueri Sampel

Melihat informasi pemrosesan untuk langkah kueri

Kueri berikut menunjukkan informasi pemrosesan dasar untuk setiap langkah kueri 87:

```
select query, stm, seg, step, rows, bytes
from svcs_query_summary
where query = 87
order by query, seg, step;
```

Kueri ini mengambil informasi pemrosesan tentang kueri 87, seperti yang ditunjukkan dalam output sampel berikut:

```
 query | stm | seg | step |  rows  |  bytes
-------+-----+-----+------+--------+---------
87     |   0 |   0 |    0 |     90 |    1890
87     |   0 |   0 |    2 |     90 |     360
87     |   0 |   1 |    0 |     90 |     360
87     |   0 |   1 |    2 |     90 |    1440
87     |   1 |   2 |    0 | 210494 | 4209880
87     |   1 |   2 |    3 |  89500 |       0
87     |   1 |   2 |    6 |      4 |      96
87     |   2 |   3 |    0 |      4 |      96
87     |   2 |   3 |    1 |      4 |      96
87     |   2 |   4 |    0 |      4 |      96
87     |   2 |   4 |    1 |      1 |      24
87     |   3 |   5 |    0 |      1 |      24
87     |   3 |   5 |    4 |      0 |       0
(13 rows)
```

Menentukan apakah langkah-langkah kueri tumpah ke disk

Kueri berikut menunjukkan apakah salah satu langkah untuk kueri dengan ID kueri 1025 (lihat SVL_QLOG tampilan untuk mempelajari cara mendapatkan ID kueri untuk kueri) tumpah ke disk atau jika kueri berjalan sepenuhnya dalam memori:

```
select query, step, rows, workmem, label, is_diskbased
from svcs_query_summary
where query = 1025
order by workmem desc;
```

Kueri ini mengembalikan output sampel berikut:

```
query| step|  rows  |  workmem   |  label        | is_diskbased
-----+-----+--------+------------+---------------+--------------
```

```
1025 |  0  |16000000|  141557760 |scan tbl=9     | f
1025 |  2  |16000000|  135266304 |hash tbl=142   | t
1025 |  0  |16000000|  128974848 |scan tbl=116536| f
1025 |  2  |16000000|  122683392 |dist           | f
(4 rows)
```

Dengan memindai nilai untuk IS_DISKBASED, Anda dapat melihat langkah kueri mana yang masuk ke disk. Untuk query 1025, langkah hash berjalan pada disk. Langkah-langkah mungkin berjalan pada disk termasuk langkah hash, aggr, dan pengurutan. Untuk melihat hanya langkah-langkah query berbasis disk, tambahkan **`and is_diskbased = 't'`** klausa ke pernyataan SQL dalam contoh di atas.

SVCS_S3LIST

Gunakan tampilan SVCS_S3LIST untuk mendapatkan detail tentang kueri Amazon Redshift Spectrum di tingkat segmen. Satu segmen dapat melakukan satu pemindaian tabel eksternal. Tampilan ini berasal dari tampilan sistem SVL_S3LIST tetapi tidak menampilkan tingkat irisan untuk kueri yang dijalankan pada cluster penskalaan konkurensi.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tampilan dengan awalan SVL kecuali bahwa tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_S3LIST terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang SVL_S3LIST, lihat. SVL_S3LIST

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri.

Nama kolom	Jenis data	Deskripsi
segmen	integer	Nomor segmen. Sebuah query terdiri dari beberapa segmen.
simpul	integer	Nomor simpul.
waktu acara	timestamp	Waktu dalam UTC saat peristiwa dicatat.
bucket	karakter (256)	Nama bucket Amazon S3.
prefix	karakter (256)	Awalan lokasi bucket Amazon S3.
rekursif	harimau (1)	Apakah ada pemindaian rekursif untuk subfolder.
file_diambil	integer	Jumlah file yang terdaftar.
maksimum_file_size	bigint	Ukuran file maksimum di antara file yang terdaftar.
avg_file_size	double precision	Ukuran file rata-rata di antara file yang terdaftar.
pemisahan yang dihasilkan	integer	Jumlah pemisahan file.
avg_split_length	double precision	Panjang rata-rata file terbagi dalam byte.
durasi	bigint	Durasi daftar file, dalam mikrodetik.

Contoh kueri

Contoh berikut menanyakan SVCS_S3LIST untuk kueri terakhir yang dilakukan.

```
select *
from svcs_s3list
```

```
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3LOG

Gunakan tampilan SVCS_S3LOG untuk mendapatkan detail pemecahan masalah tentang kueri data lake di tingkat segmen. Satu segmen dapat melakukan satu pemindaian tabel eksternal. Tampilan ini berasal dari tampilan sistem SVL_S3LOG tetapi tidak menampilkan tingkat irisan untuk kueri yang dijalankan pada cluster penskalaan konkurensi.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tampilan dengan awalan SVL kecuali bahwa tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_S3LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang SVL_S3LOG, lihat. SVL_S3LOG

Kolom tabel

Nama kolom	Jenis data	Deskripsi
pid	integer	ID proses.
kueri	integer	ID kueri.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
langkah	integer	Langkah query yang berjalan.
simpul	integer	Nomor simpul.

Nama kolom	Jenis data	Deskripsi
waktu acara	timestamp	Waktu dalam UTC saat peristiwa dicatat.
pesan	karakter (512)	Pesan untuk entri log.

Contoh kueri

Contoh berikut menanyakan SVCS_S3LOG untuk kueri terakhir yang dijalankan.

```
select *
from svcs_s3log
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3PARTITION_RINGKASAN

Gunakan tampilan SVCS_S3PARTITION_SUMMARY untuk mendapatkan ringkasan pemrosesan partisi kueri Redshift Spectrum di tingkat segmen. Satu segmen dapat melakukan satu pemindaian tabel eksternal.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tampilan dengan awalan SVL kecuali bahwa tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_S3PARTITION_SUMMARY dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang SVL_S3PARTITION, lihat. SVL_S3PARTISI

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Anda dapat menggunakan nilai ini untuk menggabungkan berbagai tabel dan tampilan sistem lainnya.
segmen	integer	Nomor segmen. Sebuah query terdiri dari beberapa segmen.
penetapan	harimau (1)	Jenis penetapan partisi di seluruh node.
min_waktu mulai	timestamp	Waktu dalam UTC saat pemrosesan partisi dimulai.
maksimum_ waktu akhir	timestamp	Waktu dalam UTC saat pemrosesan partisi selesai.
durasi minimum	bigint	Waktu pemrosesan partisi minimum yang digunakan oleh node untuk kueri ini (dalam mikrodetik).
durasi maksimum	bigint	Waktu pemrosesan partisi maksimum yang digunakan oleh node untuk kueri ini (dalam mikrodetik).
avg_durasi	bigint	Waktu pemrosesan partisi rata-rata yang digunakan oleh node untuk kueri ini (dalam mikrodetik).
total_partisi	integer	Jumlah total partisi dalam tabel eksternal.
partisi_b erkualifi kasi	integer	Jumlah total partisi yang memenuhi syarat.
partisi_m in_yang ditugaskan	integer	Jumlah minimum partisi yang ditetapkan pada satu node.

Nama kolom	Jenis data	Deskripsi
partisi_maksimum yang ditugaskan	integer	Jumlah maksimum partisi yang ditetapkan pada satu node.
partisi_avg_yang ditugaskan	bigint	Jumlah rata-rata partisi yang ditetapkan pada satu node.

Contoh kueri

Contoh berikut mendapatkan detail pemindaian partisi untuk kueri terakhir yang dilakukan.

```
select query, segment, assignment, min_starttime, max_endtime, min_duration,
 avg_duration
from svcs_s3partition_summary
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3QUERY_RINGKASAN

Gunakan tampilan SVCS_S3QUERY_SUMMARY untuk mendapatkan ringkasan dari semua kueri data lake yang telah dijalankan pada sistem. Satu segmen dapat melakukan satu pemindaian tabel eksternal.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tampilan dengan awalan SVL kecuali bahwa tampilan SVL memberikan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_S3QUERY_SUMMARY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang SVL_S3QUERY, lihat. SVL_S3QUERY

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang menghasilkan entri yang diberikan.
kueri	integer	ID kueri. Anda dapat menggunakan nilai ini untuk menggabungkan berbagai tabel dan tampilan sistem lainnya.
xid	bigint	ID transaksi.
pid	integer	ID proses.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
langkah	integer	Langkah query yang berjalan.
waktu mulai	timestamp	Waktu dalam UTC kueri Redshift Spectrum di segmen ini mulai berjalan. Satu segmen dapat memiliki satu pemindaian tabel eksternal.
waktu akhir	timestamp	Waktu dalam UTC saat kueri Redshift Spectrum di segmen ini selesai. Satu segmen dapat memiliki satu pemindaian tabel eksternal.
berlalu	integer	Lamanya waktu yang dibutuhkan kueri Redshift Spectrum di segmen ini untuk dijalankan (dalam mikrodetik).
dibatalkan	integer	Jika kueri dihentikan oleh sistem atau dibatalkan oleh pengguna, kolom ini berisi**1**. Jika kueri berjalan hingga selesai, kolom ini berisi**0**.

Nama kolom	Jenis data	Deskripsi
nama_tabel_eksternal	karakter (136)	Format internal nama nama eksternal tabel untuk pemindaian tabel eksternal.
file_format	karakter (16)	Format file dari data tabel eksternal.
dipartisi_dipartisi	harimau (1)	Jika true (**t**), nilai kolom ini menunjukkan bahwa tabel eksternal dipartisi.
is_rrscan	harimau (1)	Jika true (**t**), nilai kolom ini menunjukkan bahwa pemindaian terbatas rentang telah diterapkan.
is_bersarang	Whisky (1)	Jika true (**t**), nilai kolom ini menunjukkan bahwa tipe data kolom bersarang diakses.
s3_scanned_baris	bigint	Jumlah baris yang dipindai dari Amazon S3 dan dikirim ke lapisan Redshift Spectrum. Pada cluster yang disediakan RG, ini adalah jumlah total baris yang dipindai langsung dari Amazon S3 oleh pembaca asli cluster di semua irisan, sebelum pengosongan filter.
s3_scanned_byte	bigint	Jumlah byte yang dipindai dari Amazon S3 dan dikirim ke lapisan Redshift Spectrum, berdasarkan data terkompresi. Pada cluster yang disediakan RG, ini adalah jumlah total byte yang dipindai langsung dari Amazon S3 oleh pembaca asli cluster di semua irisan, sebelum pengosongan filter.
s3query_dikembalikan_baris	bigint	Jumlah baris yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Pada cluster yang disediakan RG, ini adalah jumlah total baris yang dikembalikan oleh pembaca asli cluster di semua irisan, setelah penembakan filter.

Nama kolom	Jenis data	Deskripsi
s3query_dikembalikan_byte	bigint	Jumlah byte yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Sejumlah besar data yang dikembalikan ke Amazon Redshift dapat memengaruhi kinerja sistem. Pada cluster yang disediakan RG, ini adalah jumlah total byte yang dikembalikan oleh pembaca asli cluster di semua irisan, setelah penembakan filter.
file	integer	Jumlah file yang diproses untuk query danau data ini. Sejumlah kecil file membatasi manfaat pemrosesan paralel.
file_maks	integer	Jumlah maksimum file yang diproses pada satu irisan.
file_avg	integer	Jumlah rata-rata file yang diproses pada satu irisan.
memisahkan	bigint	Jumlah split yang diproses untuk segmen ini. Jumlah split yang diproses pada irisan ini. Dengan file data terbagi besar, misalnya, file data yang lebih besar dari sekitar 512 MB, Redshift Spectrum mencoba membagi file menjadi beberapa permintaan S3 untuk pemrosesan paralel.
splits_max	integer	Jumlah maksimum pemisahan yang diproses pada irisan ini.
splits_avg	bigint	Jumlah rata-rata pemisahan yang diproses pada irisan ini.
total_split_ukuran	bigint	Ukuran total semua split diproses.
maksimum_split_ukuran	bigint	Ukuran split maksimum diproses, dalam byte.
avg_split_ukuran	bigint	Ukuran split rata-rata diproses, dalam byte.

Nama kolom	Jenis data	Deskripsi
total_retry	bigint	Jumlah total percobaan ulang untuk kueri Redshift Spectrum di segmen ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1. Untuk percobaan ulang klien Amazon S3 di RG, lihat. STL_S3CLIENT
max_retries	integer	Jumlah maksimum percobaan ulang untuk satu file yang diproses individual. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1. Untuk percobaan ulang klien Amazon S3 di RG, lihat. STL_S3CLIENT
durasi permohonan maksimal	bigint	Durasi maksimum permintaan file individu (dalam mikrodetik). Kueri yang berjalan lama mungkin menunjukkan hambatan. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
avg_request_durasi	bigint	Durasi rata-rata permintaan file (dalam mikrodetik). Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
max_request_paralelisme	integer	Jumlah maksimum permintaan paralel pada satu irisan untuk kueri Redshift Spectrum ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
avg_request_paralelisme	double precision	Jumlah rata-rata permintaan paralel pada satu irisan untuk kueri Redshift Spectrum ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
total_perlambatan_hitun_	bigint	Jumlah total permintaan Amazon S3 dengan kesalahan perlambatan yang terjadi selama pemindaian tabel eksternal. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.

Nama kolom	Jenis data	Deskripsi
penghitungan_perlambatan maksimum	integer	Jumlah maksimum permintaan Amazon S3 dengan kesalahan perlambatan yang terjadi selama pemindaian tabel eksternal pada satu irisan. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.

Contoh kueri

Contoh berikut mendapatkan detail langkah pemindaian untuk menjalankan kueri terakhir.

```
select query, segment, elapsed, s3_scanned_rows, s3_scanned_bytes,
 s3query_returned_rows, s3query_returned_bytes, files
from svcs_s3query_summary
where query = pg_last_query_id()
order by query,segment;
```

```
query | segment | elapsed | s3_scanned_rows | s3_scanned_bytes | s3query_returned_rows
 | s3query_returned_bytes | files
------+---------+---------+-----------------+------------------+----------------------
+------------------------+------
 4587 |       2 |   67811 |               0 |                0 |                    0
 |                      0 |     0
 4587 |       2 |  591568 |          172462 |         11260097 |                 8513
 |                 170260 |     1
 4587 |       2 |  216849 |               0 |                0 |                    0
 |                      0 |     0
 4587 |       2 |  216671 |               0 |                0 |                    0
 |                      0 |     0
```

SVCS_STREAM_SEGS

Mencantumkan hubungan antara aliran dan segmen bersamaan.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tabel dengan awalan STL kecuali

bahwa tabel STL menyediakan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_STREAM_SEGS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Kolom query dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
aliran	integer	Kumpulan segmen bersamaan dari kueri.
segmen	integer	Nomor yang mengidentifikasi segmen kueri.

Kueri Sampel

Untuk melihat hubungan antara aliran dan segmen bersamaan untuk kueri terbaru, ketik kueri berikut:

```
select *
from svcs_stream_segs
where query = pg_last_query_id();

 query | stream | segment
-------+--------+---------
    10 |      1 |       2
    10 |      0 |       0
    10 |      2 |       4
    10 |      1 |       3
    10 |      0 |       1
(5 rows)
```

SVCS_UNLOAD_LOG

Gunakan SVCS_UNLOAD_LOG untuk mendapatkan detail operasi UNLOAD.

SVCS_UNLOAD_LOG merekam satu baris untuk setiap file yang dibuat oleh pernyataan UNLOAD. Misalnya, jika UNLOAD membuat 12 file, SVCS_UNLOAD_LOG berisi 12 baris yang sesuai. Tampilan ini berasal dari tabel sistem STL_UNLOAD_LOG tetapi tidak menampilkan tingkat irisan untuk kueri yang dijalankan pada cluster penskalaan konkurensi.

> **Note**
>
> Tampilan sistem dengan awalan SVCS memberikan rincian tentang kueri pada cluster penskalaan utama dan konkurensi. Tampilan mirip dengan tabel dengan awalan STL kecuali bahwa tabel STL menyediakan informasi hanya untuk kueri yang dijalankan pada cluster utama.

SVCS_UNLOAD_LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
Userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri.
pid	integer	ID proses yang terkait dengan pernyataan query.
path	karakter (1280)	Jalur objek Amazon S3 lengkap untuk file.
waktu mulai	timestamp	Waktu mulai untuk operasi UNLOAD.
akhir_waktu	timestamp	Waktu akhir untuk operasi UNLOAD.
hitung_garis	bigint	Jumlah baris (baris) yang dibongkar ke file.

Nama kolom	Jenis data	Deskripsi
ukuran transfer	bigint	Jumlah byte yang ditransfer.
file_format	karakter (10)	Format file yang tidak dimuat.

Contoh kueri

Untuk mendapatkan daftar file yang ditulis ke Amazon S3 dengan perintah UNLOAD, Anda dapat memanggil operasi daftar Amazon S3 setelah UNLOAD selesai; namun, tergantung pada seberapa cepat Anda mengeluarkan panggilan, daftar tersebut mungkin tidak lengkap karena operasi daftar Amazon S3 akhirnya konsisten. Untuk mendapatkan daftar lengkap dan otoritatif segera, kueri SVCS_UNLOAD_LOG.

Kueri berikut mengembalikan nama jalur untuk file yang dibuat oleh UNLOAD untuk kueri terakhir yang diselesaikan:

```
select query, substring(path,0,40) as path
from svcs_unload_log
where query = pg_last_query_id()
order by path;
```

Perintah ini mengembalikan output sampel berikut:

```
 query |             path
 ------+-----------------------------------------
  2320 | s3://amzn-s3-demo-bucket/venue0000_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0001_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0002_part_00
  2320 | s3://amzn-s3-demo-bucket/venue0003_part_00
(4 rows)
```

Tampilan SVL untuk cluster utama

Tampilan SVL adalah tampilan sistem di Amazon Redshift yang berisi referensi ke tabel STL dan log untuk informasi lebih rinci.

Tampilan ini memberikan akses yang lebih cepat dan lebih mudah ke data yang sering ditanyakan yang ditemukan di tabel tersebut.

> **Note**
>
> Tampilan SVL_QUERY_SUMMARY hanya berisi informasi tentang kueri yang dijalankan oleh Amazon Redshift, bukan utilitas dan perintah DDL lainnya. Untuk daftar lengkap dan informasi tentang semua pernyataan yang dijalankan oleh Amazon Redshift, termasuk perintah DDL dan utilitas, Anda dapat menanyakan tampilan SVL_STATEMENTTEXT.

Topik

- SVL_AUTO_WORKER_ACTION
- SVL_COMPILE
- SVL_DATASHARE_CHANGE_LOG
- SVL_DATASHARE_CROSS_REGION_USAGE
- SVL_DATASHARE_USAGE_CONSUMER
- SVL_DATASHARE_USAGE_PRODUCER
- SVL_FEDERATED_QUERY
- SVL_MULTI_STATEMENT_VIOLATIONS
- SVL_MV_REFRESH_STATUS
- SVL_QERROR
- SVL_QLOG
- SVL_QUERY_METRICS
- SVL_QUERY_METRICS_SUMMARY
- SVL_QUERY_QUEUE_INFO
- SVL_QUERY_REPORT
- SVL_QUERY_SUMMARY
- SVL_RESTORE_ALTER_TABLE_PROGRESS
- SVL_S3LIST
- SVL_S3LOG
- SVL_S3PARTISI
- SVL_S3PARTITION_SUMMARY

- SVL_S3QUERY
- SVL_S3QUERY_SUMMARY
- SVL_S3MENCOBA LAGI
- SVL_SPATIAL_MENYEDERHANAKAN
- SVL_SPECTRUM_SCAN_ERROR
- SVL_STATEMENTTEXT
- SVL_STORED_PROC_CALL
- SVL_STORED_PROC_MESSAGES
- SVL_TERMINATE
- SVL_UDF_LOG
- SVL_USER_INFO
- SVL_VACUUM_PERCENTAGE

SVL_AUTO_WORKER_ACTION

Merekam tindakan otomatis yang diambil oleh Amazon Redshift pada tabel yang ditentukan untuk pengoptimalan otomatis.

SVL_AUTO_WORKER_ACTION hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
table_id	integer	Pengidentifikasi tabel.
jenis	karakter (32)	Jenis rekomendasi. Nilai yang mungkin adalah distkey dan sortkey.
status	karakter (128)	Status penyelesaian rekomendasi. Nilai yang mungkin adalah Mulai, Lengkap, Dilewati, Batalkan, Pos Pemeriksaan, dan Gagal.
waktu acara	timestamp	Stempel waktu kolom. `status`

Nama kolom	Jenis data	Deskripsi
urutan	integer	Nomor urut dari nilai terpotong. `previous_state` Ketika satu `previous_state` berisi lebih dari 200 karakter, baris tambahan dicatat untuk nilai itu. Urutan adalah 0 adalah baris pertama, 1 adalah yang kedua, dan seterusnya.
previous_state	karakter (200)	Gaya distribusi sebelumnya dan kunci pengurutan tabel sebelum menerapkan rekomendasi. Nilai terpotong menjadi kenaikan 200 karakter.

Beberapa contoh nilai `status` kolom adalah sebagai berikut:

- Skipped:Table tidak ditemukan.
- Skipped:Recommendation kosong.
- Skipped:Apply rekomendasi sortkey dinonaktifkan.
- Skipped:Retry melebihi batas maksimum untuk sebuah tabel.
- Skipped:Table kolom telah berubah.
- Abort:This Tabel tidak AUTO.
- Abort:This tabel baru saja dikonversi.
- Abort:This tabel melebihi ambang batas ukuran tabel.
- Abort:This tabel sudah menjadi gaya yang direkomendasikan.
- Pos pemeriksaan: progres*21.9963*%.

Kueri Sampel

Dalam contoh berikut, baris dalam hasil menunjukkan tindakan yang diambil oleh Amazon Redshift.

```
select table_id, type, status, eventtime, sequence, previous_state
from SVL_AUTO_WORKER_ACTION;
```

```
 table_id |  type   |                        status                        |
 eventtime          | sequence | previous_state
```

```
----------+---------+-----------------------------------------------------
+---------------------------+----------+----------------
   118082 | sortkey | Start                                                | 2020-08-22
 19:42:20.727049 | 0        |
   118078 | sortkey | Start                                                | 2020-08-22
 19:43:54.728819 | 0        |
   118082 | sortkey | Start                                                | 2020-08-22
 19:42:52.690264 | 0        |
   118072 | sortkey | Start                                                | 2020-08-22
 19:44:14.793572 | 0        |
   118082 | sortkey | Failed                                               | 2020-08-22
 19:42:20.728917 | 0        |
   118078 | sortkey | Complete                                             | 2020-08-22
 19:43:54.792705 | 0        | SORTKEY: None;
   118086 | sortkey | Complete                                             | 2020-08-22
 19:42:00.72635  | 0        | SORTKEY: None;
   118082 | sortkey | Complete                                             | 2020-08-22
 19:43:34.728144 | 0        | SORTKEY: None;
   118072 | sortkey | Skipped:Retry exceeds the maximum limit for a table. | 2020-08-22
 19:44:46.706155 | 0        |
   118086 | sortkey | Start                                                | 2020-08-22
 19:42:00.685255 | 0        |
   118082 | sortkey | Start                                                | 2020-08-22
 19:43:34.69531  | 0        |
   118072 | sortkey | Start                                                | 2020-08-22
 19:44:46.703331 | 0        |
   118082 | sortkey | Checkpoint: progress 14.755079%                      | 2020-08-22
 19:42:52.692828 | 0        |
   118072 | sortkey | Failed                                               | 2020-08-22
 19:44:14.796071 | 0        |
   116723 | sortkey | Abort:This table is not AUTO.                        | 2020-10-28
 05:12:58.479233 | 0        |
   110203 | distkey | Abort:This table is not AUTO.                        | 2020-10-28
 05:45:54.67259  | 0        |
```

SVL_COMPILE

Rekaman mengkompilasi waktu dan lokasi untuk setiap segmen kueri kueri.

SVL_COMMPILE dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> SVL_COMMPILE hanya berisi kueri yang dijalankan pada kluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses menjelaskan rencana kueri yang dijalankan di kedua kluster utama, kluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_QUERY_HISTORY Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Untuk informasi tentang SVCS_COMMPILE, lihat. SVCS_KOMPILASI

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
pid	integer	ID proses yang terkait dengan pernyataan.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
segmen	integer	Segmen query yang akan dikompilasi.
lokus	integer	Lokasi di mana segmen berjalan. **1**jika pada node komputasi dan **2** jika pada node pemimpin.
waktu mulai	timestamp	Waktu di UTC kompilasi dimulai.
akhir waktu	timestamp	Waktu di UTC kompilasi berakhir.
mengompilasikan	integer	**0**jika kompilasi digunakan kembali, **1** jika segmen dikompilasi.

Kueri Sampel

Dalam contoh ini, kueri 35878 dan 35879 menjalankan pernyataan SQL yang sama. Kolom kompilasi untuk kueri 35878 ditampilkan 1 untuk empat segmen kueri, yang menunjukkan bahwa segmen dikompilasi. Kueri 35879 ditampilkan `0` di kolom kompilasi untuk setiap segmen, menunjukkan bahwa segmen tidak perlu dikompilasi lagi.

```
select userid, xid,  pid, query, segment, locus,
datediff(ms, starttime, endtime) as duration, compile
from svl_compile
where query = 35878 or query = 35879
order by query, segment;

 userid |  xid   |  pid  | query | segment | locus | duration | compile
--------+--------+-------+-------+---------+-------+----------+---------
    100 | 112780 | 23028 | 35878 |       0 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       1 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       2 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       3 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       4 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       5 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       6 |     1 |     1380 |       1
    100 | 112780 | 23028 | 35878 |       7 |     1 |     1085 |       1
    100 | 112780 | 23028 | 35878 |       8 |     1 |     1197 |       1
    100 | 112780 | 23028 | 35878 |       9 |     2 |      905 |       1
    100 | 112782 | 23028 | 35879 |       0 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       1 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       2 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       3 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       4 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       5 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       6 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       7 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       8 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       9 |     2 |        0 |       0
(20 rows)
```

SVL_DATASHARE_CHANGE_LOG

Merekam tampilan konsolidasi untuk melacak perubahan pada datashares pada kluster produsen dan konsumen.

SVL_DATASHARE_CHANGE_LOG terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_DATASHARE_CHANGE_LOG pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang mengambil tindakan.
nama pengguna	varchar (128)	Nama pengguna yang mengambil tindakan.
pid	integer	ID proses.
xid	bigint	ID transaksi.
share_id	integer	ID datashare terpengaruh.
share_name	varchar (128)	Nama datashare.
source_database_id	integer	ID database tempat datashare berada.
source_database_name	varchar (128)	Nama database tempat datashare berada.
consumer_database_id	integer	ID database yang diimpor dari datashare.

Nama kolom	Jenis data	Deskripsi
consumer_database_name	varchar (128)	Nama database yang diimpor dari datashare.
arn	varchar (192)	ARN dari sumber daya yang mendukung database yang diimpor.
rekor waktu	timestamp	Stempel waktu aksi.
tindakan	varchar (128)	Aksi sedang dijalankan. Nilai yang mungkin adalah CREATE DATASHARE, DROP DATASHARE, GRANT ALTER, REVOKE ALTER, GRANT SHARE, REVOKE SHARE, ALTER ADD, ALTER REMOVE, ALTER SET, GRANT USE, REVOKE USE, CREATE DATABASE, GRANT atau REVOKE USE pada database bersama, DROP SHARED DATABASE, ALTER SHARED DATABASE.
status	integer	Status tindakan. Nilai yang mungkin adalah SUCCESS dan ERROR-ERROR CODE.
share_object_type	varchar(64)	Jenis objek database yang ditambahkan atau dihapus dari datashare. Nilai yang mungkin adalah skema, tabel, kolom, fungsi, dan tampilan. Ini adalah bidang untuk cluster produser.
share_object_id	integer	ID objek database yang ditambahkan atau dihapus dari datashare. Ini adalah bidang untuk cluster produser.
share_object_name	varchar (128)	Nama objek database yang ditambahkan atau dihapus dari datashare. Ini adalah bidang untuk cluster produser.
target_user_type	varchar(16)	Jenis pengguna atau grup yang diberikan hak istimewa. Ini adalah bidang untuk cluster produsen dan konsumen.
target_userid	integer	ID pengguna atau grup yang diberikan hak istimewa. Ini adalah bidang untuk cluster produsen dan konsumen.

Nama kolom	Jenis data	Deskripsi
target_username	varchar(128)	Nama pengguna atau grup yang diberi hak istimewa. Ini adalah bidang untuk cluster produsen dan konsumen.
consumer_account	varchar(16)	ID akun konsumen data. Ini adalah bidang untuk cluster produser.
consumer_namespace	varchar(64)	Namespace dari akun konsumen data. Ini adalah bidang untuk cluster produser.
producer_account	varchar(16)	ID akun akun produsen tempat datashare milik. Ini adalah bidang untuk cluster konsumen.
producer_namespace	varchar(64)	Namespace akun produk yang dimiliki datashare. Ini adalah bidang untuk cluster konsumen.
atribut_nama	varchar(64)	Nama atribut datashare atau database bersama.
atribut_nilai	varchar(128)	Nilai atribut datashare atau database bersama.
pesan	varchar(512)	Pesan kesalahan saat tindakan gagal.

Kueri Sampel

Contoh berikut menunjukkan tampilan SVL_DATASHARE_CHANGE_LOG.

```
SELECT DISTINCT action
FROM svl_datashare_change_log
WHERE share_object_name LIKE 'tickit%';

         action
 ----------------------
  "ALTER DATASHARE ADD"
```

SVL_DATASHARE_CROSS_REGION_USAGE

Gunakan tampilan SVL_DATASHARE_CROSS_REGION_USAGE untuk mendapatkan ringkasan penggunaan data lintas wilayah yang ditransfer yang disebabkan oleh kueri pembagian data lintas wilayah. SVL_DATASHARE_CROSS_REGION_USAGE menggabungkan detail di tingkat segmen.

SVL_DATASHARE_CROSS_REGION_USAGE dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_DATASHARE_CROSS_REGION_PENGGUNAAN pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Gunakan nilai ini untuk bergabung dengan tabel dan tampilan sistem lainnya.
segmen	bigint	Jumlah segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
start_time	Waktu	Waktu di UTC transfer data dimulai.
waktu_akhir	Waktu	Waktu di UTC transfer data berakhir.
transfer_data	bigint	Jumlah byte data yang ditransfer dari Wilayah produsen ke Wilayah konsumen.
source_region	arang (25)	Wilayah produsen tempat kueri mentransfer data dari.
rekor waktu	timestamp	Waktu ketika tindakan direkam.

Kueri Sampel

Contoh berikut menunjukkan tampilan SVL_DATASHARE_CROSS_REGION_USAGE.

```
SELECT query, segment, transferred_data, source_region
from svl_datashare_cross_region_usage
where query = pg_last_query_id()
order by query,segment;

  query | segment | transferred_data | source_region
--------+---------+------------------+---------------
 200048 |       2 |          4194304 |    us-west-1
 200048 |       2 |          4194304 |    us-east-2
```

SVL_DATASHARE_USAGE_CONSUMER

Mencatat aktivitas dan penggunaan datashares. Pandangan ini hanya relevan pada cluster konsumen.

SVL_DATASHARE_USAGE_CONSUMER terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_DATASHARE_PENGGUNAAN_KONSUMEN pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang mengeluarkan permintaan.
pid	integer	ID dari proses pemimpin yang menjalankan kueri.
xid	bigint	Konteks transaksi saat ini.
request_id	varchar (50)	ID unik dari panggilan API yang diminta.

Nama kolom	Jenis data	Deskripsi
request_type	varchar (25)	Jenis permintaan yang dibuat untuk cluster produsen.
transaksi_uid	varchar (50)	ID unik transaksi.
rekor waktu	timestamp	Waktu ketika tindakan direkam.
status	integer	Status panggilan API yang diminta.
kesalahan	varchar (512)	Pesan untuk kesalahan.

Kueri Sampel

Contoh berikut menunjukkan tampilan SVL_DATASHARE_USAGE_CONSUMER.

```
SELECT request_type, status, trim(error) AS error
FROM svl_datashare_usage_consumer

  request_type  | status | error
----------------+--------+--------
 "GET RELATION" |   0    |
```

SVL_DATASHARE_USAGE_PRODUCER

Mencatat aktivitas dan penggunaan datashares. Pandangan ini hanya relevan pada cluster produser.

SVL_DATASHARE_USAGE_PRODUCER terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_DATASHARE_PENGGUNAAN_PRODUSEN pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
share_id	integer	Objek ID (OID) dari datashare.
share_name	varchar (128)	Nama datashare.
request_id	varchar (50)	ID unik dari panggilan API yang diminta.
request_type	varchar (25)	Jenis permintaan yang dibuat untuk cluster produsen.
object_type	varchar(64)	Jenis objek yang dibagikan dari datashare. Nilai yang mungkin adalah skema, tabel, kolom, fungsi, dan tampilan.
object_oid	integer	ID objek yang dibagikan dari datashare.
object_name	varchar (128)	Nama objek yang dibagikan dari datashare.
consumer_account	varchar(16)	Akun akun konsumen tempat datashare dibagikan.
consumer_namespace	varchar(64)	Namespace akun konsumen tempat datashare dibagikan.
consumer_transaction_uid	varchar (50)	ID transaksi unik dari pernyataan pada cluster konsumen.
rekor waktu	timestamp	Waktu ketika tindakan direkam.
status	integer	Status datashare.
kesalahan	varchar (512)	Pesan untuk kesalahan.

Nama kolom	Jenis data	Deskripsi
consumer_region	arang (64)	Wilayah tempat cluster konsumen berada.

Kueri Sampel

Contoh berikut menunjukkan tampilan SVL_DATASHARE_USAGE_PRODUCER.

```
SELECT DISTINCT request_type
FROM svl_datashare_usage_producer
WHERE object_name LIKE 'tickit%';

   request_type
 ------------------
   "GET RELATION"
```

SVL_FEDERATED_QUERY

Gunakan tampilan SVL_FEDERATED_QUERY untuk melihat informasi tentang panggilan kueri federasi.

SVL_FEDERATED_QUERY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_EXTERNAL_QUERY_DETAIL pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menjalankan kueri.

Nama kolom	Jenis data	Deskripsi
xid	bigint	ID transaksi.
pid	integer	ID dari proses pemimpin yang menjalankan kueri.
kueri	integer	ID kueri panggilan federasi.
sourcetype	karakter (32)	Jenis sumber panggilan federasi, misalnya “PG”.
rekor waktu	timestamp	Waktu ketika kueri dikirim untuk federasi. UTC digunakan.
querytext	karakter (4000)	String kueri dikirim ke mesin PostgreSQL jarak jauh untuk dieksekusi.
num_rows	bigint	Jumlah baris yang dikembali kan oleh kueri federasi.
num_bytes	bigint	Jumlah byte yang dikembali kan oleh kueri federasi.
durasi	bigint	Waktu (mikrodetik) dihabiska n untuk mengambil baris dari panggilan kursor. Ini adalah waktu yang dihabiskan untuk menjalankan kueri federasi, serta, mendapatkan hasil kembali.

Kueri Sampel

Untuk menampilkan informasi tentang panggilan kueri federasi, jalankan kueri berikut.

```
select query, trim(sourcetype) as type, recordtime, trim(querytext) as "PG Subquery"
 from svl_federated_query where query = 4292;

 query | type |         recordtime         |                          pg subquery
-------+------+----------------------------
+-------------------------------------------------------------
  4292 | PG   | 2020-03-27 04:29:58.485126 | SELECT "level" FROM functional.employees
 WHERE ("level" >= 6)
(1 row)
```

SVL_MULTI_STATEMENT_VIOLATIONS

Gunakan tampilan SVL_MULTI_STATEMENT_VIOLATIONS untuk mendapatkan catatan lengkap dari semua perintah SQL yang dijalankan pada sistem yang melanggar pembatasan blok transaksi.

Pelanggaran terjadi ketika Anda menjalankan salah satu perintah SQL berikut yang dibatasi Amazon Redshift di dalam blok transaksi atau permintaan multi-pernyataan:

- BUAT BASIS DATA
- DROP DATABASE
- UBAH TABEL DITAMBAHKAN
- CREATE EXTERNAL TABLE
- JATUHKAN TABEL EKSTERNAL
- GANTI NAMA TABEL EKSTERNAL
- MENGUBAH TABEL EKSTERNAL
- BUAT TABLESPACE
- JATUHKAN TABLESPACE
- BUAT PERPUSTAKAAN
- JATUHKAN PERPUSTAKAAN
- MEMBANGUN KEMBALI
- INDEXCAT
- MENGINDEKS ULANG BASIS DATA
- VAKUM
- HIBAH

- MENYALIN

> **Note**
>
> Jika ada entri dalam tampilan ini, maka ubah aplikasi dan skrip SQL yang sesuai. Sebaiknya ubah kode aplikasi Anda untuk memindahkan penggunaan perintah SQL terbatas ini di luar blok transaksi. Jika Anda membutuhkan bantuan lebih lanjut, hubungi AWS Support.

SVL_MULTI_STATEMENT_VIOLATIONS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_HISTORY pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menyebabkan pelanggaran.
basis data	karakter (32)	Nama database yang terhubung dengan pengguna.
cmdname	karakter (20)	Nama perintah yang tidak dapat berjalan di dalam blok transaksi atau permintaan multi-pernyataan. Misalnya, BUAT DATABASE, JATUHKAN DATABASE, UBAH TABEL TAMBAHKAN, BUAT TABEL EKSTERNAL, JATUHKAN TABEL EKSTERNAL, GANTI NAMA TABEL EKSTERNAL, UBAH TABEL EKSTERNAL, BUAT PERPUSTAKAAN, DROP LIBRARY, REBUILDCAT, INDEXCAT, REINDEX DATABASE, VACUUM, GRANT pada sumber daya eksternal, CLUSTER, COPY, CREATE TABLESPACE, dan DROP TABLESPACE.

Nama kolom	Jenis data	Deskripsi
xid	bigint	ID transaksi yang terkait dengan pernyataan tersebut.
pid	integer	ID proses untuk pernyataan tersebut.
label	karakter (320)	Entah nama file yang digunakan untuk menjalankan kueri atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, bidang ini kosong.
waktu mulai	timestamp	Waktu yang tepat ketika pernyataan mulai dijalankan, dengan 6 digit presisi untuk detik pecahan, misalnya: **2009-06-12 11:29:19.131358**
akhir waktu	timestamp	Waktu yang tepat ketika pernyataan selesai dieksekusi, dengan 6 digit presisi untuk detik pecahan, misalnya: **2009-06-12 11:29:19.193640**
urutan	integer	Ketika satu pernyataan berisi lebih dari 200 karakter, baris tambahan dicatat untuk pernyataan itu. Urutan 0 adalah baris pertama, 1 adalah yang kedua, dan seterusnya.
jenis	varchar (10)	Jenis pernyataan SQL:**QUERY**,**DDL**, atau**UTILITY**.
text	karakter (200)	Teks SQL, dalam peningkatan 200 karakter. Bidang ini mungkin berisi karakter khusus seperti garis miring terbalik (\\) dan baris baru (). \n

Contoh kueri

Query berikut mengembalikan beberapa pernyataan yang memiliki pelanggaran.

```
select * from svl_multi_statement_violations order by starttime asc;

userid | database | cmdname | xid | pid | label | starttime | endtime | sequence | type
 | text
```

```
================================================================================================
1 | dev | CREATE DATABASE | 1034 | 5729 |label1 | ******** | ******* | 0 | DDL |
 create table c(b int);
1 | dev | CREATE DATABASE | 1034 | 5729 |label1 | ******** | ******* | 0 | UTILITY |
 create database b;
1 | dev | CREATE DATABASE | 1034 | 5729 |label1 | ******** | ******* | 0 | UTILITY |
 COMMIT
...
```

SVL_MV_REFRESH_STATUS

Tampilan SVL_MV_REFRESH_STATUS berisi baris untuk aktivitas penyegaran tampilan terwujud.

Untuk informasi lebih lanjut tentang tampilan terwujud, lihatTampilan yang terwujud di Amazon Redshift.

SVL_MV_REFRESH_STATUS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_MV_REFRESH RIWAYAT pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
db_nama	arang (128)	Database yang berisi tampilan terwujud.
userid	bigint	ID pengguna yang melakukan penyegaran.
schema_name	arang (128)	Skema pandangan terwujud.
mv_nama	arang (128)	Nama tampilan yang terwujud.
xid	bigint	ID transaksi penyegaran.
waktu mulai	timestamp	Waktu mulai penyegaran.

Nama kolom	Jenis data	Deskripsi
akhir waktu	timestamp	Waktu akhir penyegaran.

Nama kolom	Jenis data	Deskripsi
status	text	Status penyegaran. Contoh nilai meliputi yang berikut: • Segarkan MV yang berhasil diperbarui secara bertahap Jika ini adalah tampilan terwujud untuk streaming, pesan mungkin memiliki kualifikasi tambahan mengenai jumlah catatan. Sumber daya yang dimaksud meliputi: • Stream tidak mengembalikan data baru — Tidak ada catatan yang diambil. • Semua catatan yang diterima dari aliran dilewati — Catatan diambil, tetapi karena kesalahan semua dilewati. • Beberapa catatan aliran dilewati — Catatan diambil, tetapi karena kesalahan beberapa dilewati. Jika tidak ada kualifikasi, maka setidaknya satu catatan diambil dan semua catatan tersedia dalam tampilan terwujud. Ada satu kemungkin an kualifikasi yang tersisa: • Aliran mungkin berisi lebih banyak data — Penyegaran berakhir sebelum Amazon Redshift menentukan bahwa tidak ada catatan lebih lanjut untuk dikonsumsi. Streaming bisa up to date, tetapi belum dikonfirmasi oleh Amazon Redshift. • Refresh berhasil dihitung ulang MV dari awal • Segarkan MV yang diperbarui sebagian secara bertahap hingga transaksi aktif • MV sudah diperbarui

Nama kolom	Jenis data	Deskripsi
		• <schema>Penyegaran otomatis kaskade dilewati karena tampilan terwujud. <name>tidak disegarkan. • <schema>Penyegaran kaskade dilewati karena tampilan terwujud. <name>tidak disegarkan. • <schema>Penyegaran otomatis kaskade gagal karena tampilan terwujud. <name>tidak disegarkan. <schema>Penyegaran kaskade gagal karena tampilan terwujud. <name>tidak disegarkan. • Refresh gagal. Kolom tabel dasar diganti namanya • Refresh gagal. Tipe kolom tabel dasar diubah • Refresh gagal. Tabel dasar diganti namanya • Penyegaran gagal karena kesalahan internal • Refresh gagal. Kolom tabel dasar dijatuhkan • Refresh gagal. Skema MV diganti namanya • Refresh gagal. MV tidak ditemukan • Penyegaran otomatis dibatalkan karena beban kerja pengguna yang berlebihan • Refresh gagal. Pelanggaran isolasi yang dapat diserialisasi
refresh_type	arang (32)	Definisi jenis refresh. Contoh nilai termasuk Manual dan Auto.
consumer_ account	arang (12)	ID AWS akun cluster konsumen yang memulai penyegaran. Bidang ini diisi saat penyegaran dimulai dari kluster konsumen dalam pengaturan berbagi data.

Nama kolom	Jenis data	Deskripsi
consumer_ region	arang (32)	AWS Wilayah cluster konsumen yang memulai penyegaran. Bidang ini diisi saat penyegaran dimulai dari kluster konsumen dalam pengaturan berbagi data.
consumer_ namespace	arang (36)	Pengenal namespace dari cluster konsumen yang memulai penyegaran. Bidang ini diisi saat penyegaran dimulai dari kluster konsumen dalam pengaturan berbagi data.

Contoh kueri

Untuk melihat status penyegaran tampilan terwujud, jalankan kueri berikut.

```
select * from svl_mv_refresh_status;
```

Query ini mengembalikan output sampel berikut:

```
 db_name | userid |  schema   |  name   |  xid  |         starttime          |
      endtime           |                         status
                          |  refresh_type | consumer_account | consumer_region |
 consumer_namespace
---------+--------+-----------+---------+-------+----------------------------
+----------------------------
+-------------------------------------------------------------------------------
+--------------+-----------------+----------------
+-----------------------------------
 dev     |    169 | mv_schema | mv_test |  6640 | 2020-02-14 02:26:53.497935 |
 2020-02-14 02:26:53.556156 | Refresh successfully recomputed MV from scratch
                           |  Manual       |                 |                |
 dev     |    166 | mv_schema | mv_test |  6517 | 2020-02-14 02:26:39.287438 |
 2020-02-14 02:26:39.349539 | Refresh successfully updated MV incrementally
                           |  Auto         |                 |                |
 ext_db  |    162 | producer_schema | producer_mv |  6388 | 2020-02-14 02:26:27.863426
 | 2020-02-14 02:26:27.918307 | Refresh successfully updated MV incrementally
                             |  Manual       | 0123456789      | us-east-1      |
 623d8ff2-4391-4381-83d7-177caa6767af
```

```
dev     |    161 | mv_schema | mv_test |  6323 | 2020-02-14 02:26:20.020717 |
2020-02-14 02:26:20.080002 | Refresh successfully updated MV incrementally
                           |  Auto        |                 |                |
dev     |    161 | mv_schema | mv_test |  6301 | 2020-02-14 02:26:05.796146 |
2020-02-14 02:26:07.853986 | Refresh successfully recomputed MV from scratch
                           |  Manual      |                 |                |
dev     |    153 | mv_schema | mv_test |  6024 | 2020-02-14 02:25:18.762335 |
2020-02-14 02:25:20.043462 | MV was already updated
                           |  Manual      |                 |                |
dev     |    143 | mv_schema | mv_test |  5557 | 2020-02-14 02:24:23.100601 |
2020-02-14 02:24:23.100633 | MV was already updated
                           |  Manual
dev     |    141 | mv_schema | mv_test |  5447 | 2020-02-14 02:23:54.102837 |
2020-02-14 02:24:00.310166 | Refresh successfully updated MV incrementally
                           |  Auto
dev     |      1 | mv_schema | mv_test |  5329 | 2020-02-14 02:22:26.328481 |
2020-02-14 02:22:28.369217 | Refresh successfully recomputed MV from scratch
                           |  Auto
dev     |    138 | mv_schema | mv_test |  5290 | 2020-02-14 02:21:56.885093 |
2020-02-14 02:21:56.885098 | Refresh failed. MV was not found
                           |  Manual
dev     |    138 | mv_schema | mv_test |  5100 | 2020-02-14 02:20:33.123445 |
2020-02-14 02:20:33.123445 | Cascade refresh failed because materialized view
mv_schema.child was not refreshed. |  Manual
dev     |    138 | mv_schema | child   |  5100 | 2020-02-14 02:20:33.123445 |
2020-02-14 02:20:33.123445 | Refresh failed due to an internal error.
                           |  Manual
dev     |    138 | mv_schema | mv_test |  5099 | 2020-02-14 02:10:23.492344 |
2020-02-14 02:10:23.492344 | Cascade refresh skipped because materialized view
mv_schema.child was not refreshed.|  Manual
dev     |    138 | mv_schema | child   |  5099 | 2020-02-14 02:10:23.492344 |
2020-02-14 02:10:23.492344 | Refresh failed due to an internal error.
                           |  Manual
```

SVL_QERROR

Tampilan SVL_QERROR tidak digunakan lagi.

SVL_QLOG

Tampilan SVL_QLOG berisi log dari semua kueri yang dijalankan terhadap database.

Amazon Redshift membuat tampilan SVL_QLOG sebagai subset informasi yang dapat dibaca dari tabel. STL_QUERY Gunakan tabel ini untuk menemukan ID kueri untuk kueri yang baru saja dijalankan atau untuk melihat berapa lama waktu yang dibutuhkan kueri untuk diselesaikan.

SVL_QLOG dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_HISTORY pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Anda dapat menggunakan ID ini untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
xid	bigint	ID Transaksi.
pid	integer	ID proses yang terkait dengan kueri.
waktu mulai	timestamp	Waktu yang tepat ketika pernyataan mulai dijalanka n, dengan enam digit presisi untuk detik pecahan — misalnya: **2009-06-12 11:29:19.131358**
akhir waktu	timestamp	Waktu yang tepat ketika pernyataan selesai dieksekus i, dengan enam digit presisi untuk detik pecahan — misalnya: **2009-06-12 11:29:19.193640**
berlalu	bigint	Lama waktu yang dibutuhkan kueri untuk dijalankan (dalam mikrodetik).
digugurkan	integer	Jika kueri dihentikan oleh sistem atau dibatalkan oleh pengguna, kolom ini berisi**1**. Jika kueri berjalan hingga

Nama kolom	Jenis data	Deskripsi
		selesai, kolom ini berisi**0**. Kueri yang dibatalkan untuk tujuan manajemen beban kerja dan selanjutnya dimulai ulang juga memiliki nilai di kolom ini. **1**
label	karakter (320)	Entah nama file yang digunakan untuk menjalankan kueri, atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, nilai bidang ini adalah. `default`
substring	karakter (60)	Teks kueri terpotong.
source_query	integer	Jika kueri menggunakan hasil caching, ID kueri dari kueri yang merupakan sumber hasil cache. Jika hasil caching tidak digunakan, nilai bidang ini adalah`NULL`.
concurrency_scaling_status_txt	text	Deskripsi apakah kueri berjalan di cluster utama atau cluster penskalaan konkurensi.
dari_sp_call	integer	Jika kueri dipanggil dari prosedur tersimpan, ID kueri panggilan prosedur. Jika kueri tidak dijalankan sebagai bagian dari prosedur tersimpan, bidang ini adalah`NULL`.

Kueri Sampel

Contoh berikut mengembalikan ID query, waktu eksekusi, dan teks query terpotong untuk lima query database terbaru yang dijalankan oleh pengguna dengan. `userid = 100`

```
select query, pid, elapsed, substring from svl_qlog
where userid = 100
order by starttime desc
limit 5;

 query  |  pid  | elapsed  |          substring
```

```
--------+-------+----------+-----------------------------------------------
 187752 | 18921 | 18465685 | select query, elapsed, substring from svl_...
 204168 |  5117 |    59603 | insert into testtable values (100);
 187561 | 17046 |  1003052 | select * from pg_table_def where tablename...
 187549 | 17046 |  1108584 | select * from STV_WLM_SERVICE_CLASS_CONFIG
 187468 | 17046 |  5670661 | select * from pg_table_def where schemaname...
(5 rows)
```

Contoh berikut mengembalikan nama script SQL (LABEL kolom) dan waktu berlalu untuk query yang dibatalkan (): **`aborted=1`**

```
select query, elapsed, trim(label) querylabel
from svl_qlog where aborted=1;

 query | elapsed  |       querylabel
-------+----------+-------------------------
    16 |  6935292 | alltickittablesjoin.sql
(1 row)
```

SVL_QUERY_METRICS

Tampilan SVL_QUERY_METRICS menunjukkan metrik untuk kueri yang diselesaikan. Pandangan ini berasal dari tabel STL_QUERY_METRICS sistem. Gunakan nilai dalam tampilan ini sebagai bantuan untuk menentukan nilai ambang batas untuk mendefinisikan aturan pemantauan kueri. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.

SVL_QUERY_METRICS dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_DETAIL pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menjalankan kueri yang menghasilkan entri.
kueri	integer	ID kueri. Kolom kueri dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
service_class	integer	ID untuk antrian kueri WLM (kelas layanan). Antrian kueri didefinisikan dalam konfigurasi WLM. Metrik dilaporkan hanya untuk antrian yang ditentukan pengguna. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.
dimension	varchar (24)	Dimensi di mana metrik dilaporkan. Nilai yang mungkin adalah kueri, segmen, dan langkah.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen kueri dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses. Jika nilai segmen adalah 0, nilai segmen metrik digulung ke tingkat kueri.
langkah	integer	ID untuk jenis langkah yang dilakukan. Deskripsi untuk jenis langkah ditampilkan di `step_label` kolom.
step_label	varchar (30)	Jenis langkah yang dilakukan.
query_cpu_time	bigint	Waktu CPU yang digunakan oleh kueri, dalam hitungan detik. Waktu CPU berbeda dari waktu proses kueri.
query_blocks_read	bigint	Jumlah blok 1 MB dibaca oleh kueri.
query_execution_time	bigint	Waktu eksekusi yang telah berlalu untuk kueri, dalam hitungan detik. Waktu eksekusi tidak termasuk waktu yang dihabiskan menunggu dalam antrian. Lihat `query_queue_time` untuk waktu antrian.

Nama kolom	Jenis data	Deskripsi
query_cpu _usage_percent	bigint	Persentase kapasitas CPU yang digunakan oleh query.
query_tem p_blocks_to_disk	bigint	Jumlah ruang disk yang digunakan oleh kueri untuk menulis hasil antara, dalam MB.
segment_e xecution_time	bigint	Waktu eksekusi berlalu untuk satu segmen, dalam hitungan detik.
cpu_miring	numerik (38,2)	Rasio penggunaan CPU maksimum untuk setiap irisan terhadap penggunaan CPU rata-rata untuk semua irisan. Metrik ini didefinisikan pada tingkat segmen.
io_miring	numerik (38,2)	Rasio blok maksimum read (I/O) untuk setiap irisan dengan blok rata-rata dibaca untuk semua irisan.
scan_row_count	bigint	Jumlah baris dalam langkah pemindaian. Jumlah baris adalah jumlah total baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna.
join_row_count	bigint	Jumlah baris yang diproses dalam langkah gabungan.
nested_lo op_join_r ow_count	bigint	Jumlah baris dalam loop bersarang bergabung.
kembali_r ow_count	bigint	Jumlah baris yang dikembalikan oleh kueri.
spectrum_ scan_row_count	bigint	Jumlah baris yang dipindai oleh Amazon Redshift Spectrum di Amazon S3.
spectrum_ scan_size_mb	bigint	Jumlah data, dalam MB, dipindai oleh Amazon Redshift Spectrum di Amazon S3.

Nama kolom	Jenis data	Deskripsi
query_queue_time	bigint	Jumlah waktu dalam hitungan detik kueri antri.

SVL_QUERY_METRICS_SUMMARY

Tampilan SVL_QUERY_METRICS_SUMMARY menunjukkan nilai maksimum metrik untuk kueri yang diselesaikan. Pandangan ini berasal dari tabel STL_QUERY_METRICS sistem. Gunakan nilai dalam tampilan ini sebagai bantuan untuk menentukan nilai ambang batas untuk mendefinisikan aturan pemantauan kueri. Untuk informasi selengkapnya tentang aturan dan metrik untuk pemantauan kueri untuk Amazon Redshift, lihat. Aturan pemantauan kueri WLM

SVL_QUERY_METRICS_SUMMARY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_DETAIL pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menjalankan kueri yang menghasilkan entri.
kueri	integer	ID kueri. Kolom kueri dapat digunakan untuk bergabung dengan tabel dan tampilan sistem lainnya.
service_class	integer	ID untuk antrian kueri WLM (kelas layanan). Antrian kueri didefinisikan dalam konfigurasi WLM. Metrik dilaporkan hanya untuk antrian yang ditentukan pengguna. Untuk daftar ID kelas layanan, lihatID kelas layanan WLM.

Nama kolom	Jenis data	Deskripsi
query_cpu_time	bigint	Waktu CPU yang digunakan oleh kueri, dalam hitungan detik. Waktu CPU berbeda dari waktu proses kueri.
query_blocks_read	bigint	Jumlah blok 1 MB dibaca oleh kueri.
query_execution_time	bigint	Waktu eksekusi yang telah berlalu untuk kueri, dalam hitungan detik. Waktu eksekusi tidak termasuk waktu yang dihabiskan menunggu dalam antrian.
query_cpu_usage_percent	numerik (38,2)	Persentase kapasitas CPU yang digunakan oleh query.
query_temp_blocks_to_disk	bigint	Jumlah ruang disk yang digunakan oleh kueri untuk menulis hasil antara, dalam MB.
segment_execution_time	bigint	Waktu eksekusi berlalu untuk satu segmen, dalam hitungan detik.
cpu_miring	numerik (38,2)	Rasio penggunaan CPU maksimum untuk setiap irisan terhadap penggunaan CPU rata-rata untuk semua irisan. Metrik ini didefinisikan pada tingkat segmen.
io_miring	numerik (38,2)	Rasio blok maksimum read (I/O) untuk setiap irisan dengan blok rata-rata dibaca untuk semua irisan.
scan_row_count	bigint	Jumlah baris dalam langkah pemindaian. Jumlah baris adalah jumlah total baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna.
join_row_count	bigint	Jumlah baris yang diproses dalam langkah gabungan.
nested_loop_join_row_count	bigint	Jumlah baris dalam loop bersarang bergabung.

Nama kolom	Jenis data	Deskripsi
kembali_row_count	bigint	Jumlah baris yang dikembalikan oleh kueri.
spectrum_scan_row_count	bigint	Jumlah baris yang dipindai oleh Amazon Redshift Spectrum di Amazon S3.
spectrum_scan_size_mb	bigint	Jumlah data, dalam MB, dipindai oleh Amazon Redshift Spectrum di Amazon S3.
query_queue_time	bigint	Jumlah waktu dalam hitungan detik kueri antri.

SVL_QUERY_QUEUE_INFO

Merangkum detail untuk kueri yang menghabiskan waktu dalam antrian kueri manajemen beban kerja (WLM) atau antrian komit.

Tampilan SVL_QUERY_QUEUE_INFO memfilter kueri yang dilakukan oleh sistem dan hanya menampilkan kueri yang dilakukan oleh pengguna.

Tampilan SVL_QUERY_QUEUE_INFO merangkum informasi dari tabel,, dan sistem. STL_QUERY STL_WLM_QUERY STL_COMM_STATS

SVL_QUERY_QUEUE_INFO hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
basis data	text	Nama database yang terhubung dengan pengguna saat kueri dikeluarkan.
kueri	integer	ID kueri.
xid	bigint	ID Transaksi.

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menghasilkan kueri.
querytxt	text	100 karakter pertama dari teks kueri.
antrian_start_time	timestamp	Waktu di UTC ketika kueri memasuki antrian WLM.
exec_start_time	timestamp	Waktu di UTC saat eksekusi kueri dimulai.
service_class	integer	ID untuk kelas layanan. Kelas layanan didefinisikan dalam file konfigurasi WLM.
slot	integer	Jumlah slot kueri WLM.
antrian_berlalu	bigint	Waktu yang kueri dihabiskan menunggu dalam antrian WLM (dalam hitungan detik).
exec_elapsed	bigint	Waktu yang dihabiskan untuk mengeksekusi kueri (dalam hitungan detik).
wlm_total_elapsed	bigint	Waktu yang dihabiskan kueri dalam antrian WLM (queue_elapsed), ditambah waktu yang dihabiskan untuk mengeksekusi kueri (exec_elapsed).
commit_queue_lapsed	bigint	Waktu yang kueri dihabiskan menunggu dalam antrian komit (dalam hitungan detik).
commit_exec_time	bigint	Waktu yang kueri dihabiskan dalam operasi komit (dalam detik).
service_class_name	karakter (64)	Nama kelas layanan.

Kueri Sampel

Contoh berikut menunjukkan waktu yang kueri dihabiskan dalam antrian WLM.

```
select query, service_class, queue_elapsed, exec_elapsed, wlm_total_elapsed
from svl_query_queue_info
where wlm_total_elapsed > 0;

  query  | service_class | queue_elapsed | exec_elapsed | wlm_total_elapsed
---------+---------------+---------------+--------------+-------------------
 2742669 |             6 |             2 |          916 |               918
 2742668 |             6 |             4 |          197 |               201
(2 rows)
```

SVL_QUERY_REPORT

Amazon Redshift membuat tampilan SVL_QUERY_REPORT dari UNION sejumlah tabel sistem Amazon Redshift STL untuk memberikan informasi tentang langkah-langkah kueri yang diselesaikan.

Tampilan ini memecah informasi tentang kueri yang diselesaikan berdasarkan irisan dan langkah demi langkah, yang dapat membantu mengatasi masalah node dan irisan di cluster Amazon Redshift.

SVL_QUERY_REPORT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_DETAIL pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.

Nama kolom	Jenis data	Deskripsi
mengiris	integer	Irisan data tempat langkahnya berjalan.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen kueri dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses.
langkah	integer	Langkah kueri yang selesai.
start_time	timestamp	Waktu yang tepat di UTC saat segmen mulai dieksekusi, dengan 6 digit presisi untuk detik pecahan. Misalnya: **2012-12-12 11:29:19.131358**
waktu_akhir	timestamp	Waktu yang tepat di UTC ketika segmen selesai dieksekusi, dengan 6 digit presisi untuk detik pecahan. Misalnya: **2012-12-12 11:29:19.131467**
berlalu_waktu	bigint	Waktu (dalam mikrodetik) yang dibutuhkan segmen untuk berjalan.
baris	bigint	Jumlah baris yang dihasilkan oleh langkah (per irisan). Angka ini mewakili jumlah baris untuk irisan yang dihasilkan dari eksekusi langkah, bukan jumlah baris yang diterima atau diproses oleh langkah. Dengan kata lain, ini adalah jumlah baris yang bertahan dari langkah dan diteruskan ke langkah berikutnya.
bytes	bigint	Jumlah byte yang dihasilkan oleh langkah (per irisan).
label	arang (256)	Label langkah, yang terdiri dari nama langkah kueri dan, bila berlaku, ID tabel dan nama tabel (misalnya,`scan tbl=100448 name =user`). Three-digit ID tabel biasanya mengacu pada pemindaian tabel transien. Ketika Anda melihat `tbl=0`, biasanya mengacu pada pemindaian nilai konstan.

Nama kolom	Jenis data	Deskripsi
is_diskbased	karakter (1)	Apakah langkah kueri ini dilakukan sebagai operasi berbasis disk: true (**t**) atau false (**f**). Hanya langkah-langkah tertentu, seperti hash, sortir, dan langkah agregat, yang dapat masuk ke disk. Banyak jenis langkah selalu dilakukan dalam memori.
workmem	bigint	Jumlah memori kerja (dalam byte) ditetapkan ke langkah query. Nilai ini adalah ambang query_working_mem yang dialokasikan untuk digunakan selama eksekusi, bukan jumlah memori yang sebenarnya digunakan
is_rrscan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian terbatas rentang digunakan pada langkah tersebut.
is_delayed_scan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian tertunda digunakan pada langkah.
baris_pre_filter	bigint	Untuk pemindaian tabel permanen, jumlah baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna.

Kueri Sampel

Kueri berikut menunjukkan kemiringan data dari baris yang dikembalikan untuk kueri dengan kueri ID 279. Gunakan kueri ini untuk menentukan apakah data database didistribusikan secara merata di atas irisan di cluster gudang data:

```
select query, segment, step, max(rows), min(rows),
case when sum(rows) > 0
then ((cast(max(rows) -min(rows) as float)*count(rows))/sum(rows))
else 0 end
from svl_query_report
where query = 279
group by query, segment, step
order by segment, step;
```

Kueri ini harus mengembalikan data yang mirip dengan output sampel berikut:

```
query | segment | step |   max    |   min    |         case
------+---------+------+----------+----------+----------------------
279 |       0 |    0 | 19721687 | 19721687 |                    0
279 |       0 |    1 | 19721687 | 19721687 |                    0
279 |       1 |    0 |   986085 |   986084 | 1.01411202804304e-06
279 |       1 |    1 |   986085 |   986084 | 1.01411202804304e-06
279 |       1 |    4 |   986085 |   986084 | 1.01411202804304e-06
279 |       2 |    0 |  1775517 |   788460 |     1.00098637606408
279 |       2 |    2 |  1775517 |   788460 |     1.00098637606408
279 |       3 |    0 |  1775517 |   788460 |     1.00098637606408
279 |       3 |    2 |  1775517 |   788460 |     1.00098637606408
279 |       3 |    3 |  1775517 |   788460 |     1.00098637606408
279 |       4 |    0 |  1775517 |   788460 |     1.00098637606408
279 |       4 |    1 |  1775517 |   788460 |     1.00098637606408
279 |       4 |    2 |        1 |        1 |                    0
279 |       5 |    0 |        1 |        1 |                    0
279 |       5 |    1 |        1 |        1 |                    0
279 |       6 |    0 |       20 |       20 |                    0
279 |       6 |    1 |        1 |        1 |                    0
279 |       7 |    0 |        1 |        1 |                    0
279 |       7 |    1 |        0 |        0 |                    0
(19 rows)
```

SVL_QUERY_SUMMARY

Gunakan tampilan SVL_QUERY_SUMMARY untuk menemukan informasi umum tentang eksekusi kueri.

Tampilan SVL_QUERY_SUMMARY berisi subset data dari tampilan SVL_QUERY_REPORT. Perhatikan bahwa informasi dalam SVL_QUERY_SUMMARY dikumpulkan dari semua node.

> **Note**
>
> Tampilan SVL_QUERY_SUMMARY hanya berisi informasi tentang kueri yang dilakukan oleh Amazon Redshift, bukan utilitas dan perintah DDL lainnya. Untuk daftar lengkap dan informasi tentang semua pernyataan yang dilakukan oleh Amazon Redshift, termasuk perintah DDL dan utilitas, Anda dapat menanyakan tampilan SVL_STATEMENTTEXT.

SVL_QUERY_SUMMARY dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_DETAIL pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Untuk informasi tentang SVCS_QUERY_SUMMARY, lihat. SVCS_QUERY_RINGKASAN

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang membuat entri.
kueri	integer	ID kueri. Dapat digunakan untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
stm	integer	Stream: Satu set segmen bersamaan dalam kueri. Kueri memiliki satu atau lebih aliran.
seg	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah. Segmen kueri dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses.
langkah	integer	Langkah kueri yang berjalan.
maxtime	bigint	Jumlah waktu maksimum untuk menjalankan langkah (dalam mikrodetik).
avgtime	bigint	Waktu rata-rata untuk langkah berjalan (dalam mikrodetik).
baris	bigint	Jumlah baris data yang terlibat dalam langkah kueri.
byte	bigint	Jumlah byte data yang terlibat dalam langkah kueri.

Nama kolom	Jenis data	Deskripsi
rate_row	double precision	Tingkat eksekusi kueri per baris.
rate_byte	double precision	Tingkat eksekusi kueri per byte.
label	text	Label langkah, yang terdiri dari nama langkah kueri dan, bila berlaku, ID tabel dan nama tabel (misalnya, pindai tbl=10044 8 name =user). Three-digit ID tabel biasanya mengacu pada pemindaian tabel transien. Ketika Anda melihat `tbl=0`, biasanya mengacu pada pemindaian nilai konstan.
is_diskbased	karakter (1)	Apakah langkah kueri ini dilakukan sebagai operasi berbasis disk pada node mana pun di cluster: true (**t**) atau false (**f**). Hanya langkah-langkah tertentu, seperti hash, sortir, dan langkah agregat, yang dapat masuk ke disk. Banyak jenis langkah selalu dilakukan dalam memori.
workmem	bigint	Jumlah memori kerja (dalam byte) ditetapkan ke langkah query.
is_rrscan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian terbatas rentang digunakan pada langkah tersebut. Default adalah false (**f**).
is_delayed_scan	karakter (1)	Jika true (**t**), menunjukkan bahwa pemindaian tertunda digunakan pada langkah. Default adalah false (**f**).
baris_pre_filter	bigint	Untuk pemindaian tabel permanen, jumlah baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu).

Kueri Sampel

Melihat informasi pemrosesan untuk langkah kueri

Kueri berikut menunjukkan informasi pemrosesan dasar untuk setiap langkah kueri 87:

```
select query, stm, seg, step, rows, bytes
```

```
from svl_query_summary
where query = 87
order by query, seg, step;
```

Query ini mengambil informasi pemrosesan tentang query 87, seperti yang ditunjukkan dalam contoh output berikut:

```
 query | stm | seg | step |  rows  |  bytes
-------+-----+-----+------+--------+---------
87     |   0 |   0 |    0 |     90 |    1890
87     |   0 |   0 |    2 |     90 |     360
87     |   0 |   1 |    0 |     90 |     360
87     |   0 |   1 |    2 |     90 |    1440
87     |   1 |   2 |    0 | 210494 | 4209880
87     |   1 |   2 |    3 |  89500 |       0
87     |   1 |   2 |    6 |      4 |      96
87     |   2 |   3 |    0 |      4 |      96
87     |   2 |   3 |    1 |      4 |      96
87     |   2 |   4 |    0 |      4 |      96
87     |   2 |   4 |    1 |      1 |      24
87     |   3 |   5 |    0 |      1 |      24
87     |   3 |   5 |    4 |      0 |       0
(13 rows)
```

Menentukan apakah langkah-langkah kueri tumpah ke disk

Kueri berikut menunjukkan apakah salah satu langkah untuk kueri dengan ID kueri 1025 (lihat SVL_QLOG tampilan untuk mempelajari cara mendapatkan ID kueri untuk kueri) tumpah ke disk atau jika kueri berjalan sepenuhnya dalam memori:

```
select query, step, rows, workmem, label, is_diskbased
from svl_query_summary
where query = 1025
order by workmem desc;
```

Query ini mengembalikan output sampel berikut:

```
query| step|  rows  |  workmem   |  label        | is_diskbased
-----+-----+--------+------------+---------------+--------------
1025 |  0  |16000000|  141557760 |scan tbl=9     | f
1025 |  2  |16000000|  135266304 |hash tbl=142   | t
```

```
1025 |  0  |16000000|  128974848 |scan tbl=116536| f
1025 |  2  |16000000|  122683392 |dist           | f
(4 rows)
```

Dengan memindai nilai untuk IS_DISKBASED, Anda dapat melihat langkah kueri mana yang masuk ke disk. Untuk kueri 1025, langkah hash berjalan pada disk. Langkah-langkah yang mungkin berjalan pada disk termasuk hash, aggr, dan langkah pengurutan. Untuk melihat hanya langkah-langkah query berbasis disk, tambahkan **`and is_diskbased = 't'`** klausa ke pernyataan SQL dalam contoh di atas.

SVL_RESTORE_ALTER_TABLE_PROGRESS

Gunakan SVL_RESTORE_ALTER_TABLE_PROGRESS untuk memantau kemajuan migrasi setiap tabel di cluster selama pengubahan ukuran klasik ke node RG atau RA3. Ini menangkap throughput historis migrasi data selama operasi pengubahan ukuran. Untuk informasi selengkapnya tentang pengubahan ukuran klasik ke node RG atau RA3, buka Classic resize.

SVL_RESTORE_ALTER_TABLE_PROGRESS hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_RESTORE_LOG pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

> **Note**
>
> Baris dengan kemajuan `100.00%` atau `ABORTED` dihapus setelah 7 hari. Baris untuk tabel yang dijatuhkan selama atau setelah pengubahan ukuran klasik masih dapat muncul di SVL_RESTORE_ALTER_TABLE_PROGRESS.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
tbl	integer	ID tabel.

Nama kolom	Jenis data	Deskripsi
kemajuan	arang (32)	Status kemajuan redistribusi tabel. Nilai yang mungkin adalah persentase dari `0.00%` ke `100.00%` dan pesan`ABORTED`. `ABORTED`berarti redistribusi dihentikan tanpa finishing, dengan alasan dijelaskan di `message` kolom.
pesan	arang (256)	Pesan yang terkait dengan kemajuan redistribusi tabel.

Contoh kueri

Query berikut mengembalikan query berjalan dan antrian.

```
select * from svl_restore_alter_table_progress;

tbl     | progress |                           message
--------+----------+-------------------------------------------------------------
105614  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105610  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105594  | 0.00%    | Table waiting for alter diststyle conversion.
105602  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105606  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105598  | 100.00%  | Restored to distkey successfully.
```

SVL_S3LIST

Gunakan tampilan SVL_S3LIST untuk mendapatkan detail tentang kueri data lake di tingkat segmen.

SVL_S3LIST dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> SVL_S3LIST hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses menjelaskan rencana kueri yang dijalankan di kedua kluster utama, kluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan

pemantauan SYS. SYS_EXTERNAL_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen.
simpul	integer	Nomor simpul.
mengiris	integer	Irisan data yang dilawan segmen tertentu.
waktu acara	timestamp	Waktu di UTC bahwa acara tersebut direkam.
bucket	text	Nama bucket Amazon S3.
prefix	text	Awalan lokasi bucket Amazon S3.
rekursif	arang (1)	Apakah ada pemindaian rekursif untuk subfolder.
retrieved_files	integer	Jumlah file yang terdaftar.
max_file_size	bigint	Ukuran file maksimum di antara file yang terdaftar.
avg_file_size	double precision	Ukuran file rata-rata di antara file yang terdaftar.
dihasilkan_split	integer	Jumlah pemisahan file.
avg_split_length	double precision	Rata-rata panjang file split dalam byte.

Nama kolom	Jenis data	Deskripsi
durasi	bigint	Durasi daftar file, dalam mikrodetik.

Contoh kueri

Contoh berikut menanyakan SVL_S3LIST untuk kueri terakhir yang akan dijalankan.

```
select *
from svl_s3list
where query = pg_last_query_id()
order by query,segment;
```

SVL_S3LOG

Gunakan tampilan SVL_S3LOG untuk mendapatkan detail tentang kueri data lake di segmen dan tingkat irisan node.

SVL_S3LOG dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> SVL_S3LOG hanya berisi kueri yang dijalankan pada kluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses menjelaskan rencana kueri yang dijalankan di kedua kluster utama, kluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_EXTERNAL_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
pid	integer	ID proses.
kueri	integer	ID kueri.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
langkah	integer	Langkah kueri yang berjalan.
simpul	integer	Nomor simpul.
mengiris	integer	Irisan data yang dilawan segmen tertentu.
waktu acara	timestamp	Waktu di UTC bahwa langkah mulai dijalankan.
pesan	text	Pesan untuk entri log.

Contoh kueri

Contoh berikut menanyakan SVL_S3LOG untuk kueri terakhir yang berjalan.

```
select *
from svl_s3log
where query = pg_last_query_id()
order by query,segment,slice;
```

SVL_S3PARTISI

Gunakan tampilan SVL_S3PARTITION untuk mendapatkan detail tentang partisi data lake di segmen dan tingkat irisan node.

SVL_S3PARTITION terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> SVL_S3PARTITION hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses menjelaskan rencana kueri yang dijalankan di kedua kluster utama, kluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_EXTERNAL_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
simpul	integer	Nomor simpul.
mengiris	integer	Irisan data yang dilawan segmen tertentu.
waktu mulai	stempel waktu tanpa zona waktu	Waktu di UTC pemangkasan partisi mulai dijalankan.
akhir waktu	stempel waktu tanpa zona waktu	Waktu di UTC pemangkasan partisi selesai.
durasi	bigint	Waktu berlalu (dalam mikrodetik).
total_partisi	integer	Jumlah total partisi.
qualified _partitions	integer	Jumlah partisi yang memenuhi syarat.
assigned_ partitions	integer	Jumlah partisi yang ditugaskan pada irisan.

Nama kolom	Jenis data	Deskripsi
penetapan	karakter	Jenis tugas.

Contoh kueri

Contoh berikut mendapatkan rincian partisi untuk query terakhir selesai.

```
SELECT query, segment,
       MIN(starttime) AS starttime,
       MAX(endtime) AS endtime,
       datediff(ms,MIN(starttime),MAX(endtime)) AS dur_ms,
       MAX(total_partitions) AS total_partitions,
       MAX(qualified_partitions) AS qualified_partitions,
       MAX(assignment) as assignment_type
FROM svl_s3partition
WHERE query=pg_last_query_id()
GROUP BY query, segment
```

```
query | segment |           starttime           |           endtime           | dur_ms|
 total_partitions | qualified_partitions | assignment_type
------+---------+-------------------------------+-----------------------------+-------
+------------------+----------------------+----------------
99232 |       0 | 2018-04-17 22:43:50.201515    | 2018-04-17 22:43:54.674595  |  4473 |
       2526       |        334           | p
```

SVL_S3PARTITION_SUMMARY

Gunakan tampilan SVL_S3PARTITION_SUMMARY untuk mendapatkan ringkasan pemrosesan partisi kueri Redshift Spectrum di tingkat segmen.

SVL_S3PARTITION_SUMMARY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Untuk informasi tentang SVCS_S3PARTITION, lihat. SVCS_S3PARTITION_RINGKASAN

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Anda dapat menggunakan nilai ini untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen.
penetapan	arang (1)	Jenis penugasan partisi di seluruh node.
min_start time	timestamp	Waktu di UTC bahwa pemrosesan partisi dimulai.
max_endti me	timestamp	Waktu di UTC bahwa pemrosesan partisi selesai.
min_durasi	bigint	Waktu pemrosesan partisi minimum yang digunakan oleh node untuk kueri ini (dalam mikrodetik).
maks_dura si	bigint	Waktu pemrosesan partisi maksimum yang digunakan oleh node untuk kueri ini (dalam mikrodetik).
avg_durat ion	bigint	Rata-rata waktu pemrosesan partisi yang digunakan oleh node untuk query ini (dalam mikrodetik).
total_partisi	integer	Jumlah total partisi dalam tabel eksternal.
qualified _partitions	integer	Jumlah total partisi yang memenuhi syarat.
min_assig ned_parti tions	integer	Jumlah minimum partisi yang ditetapkan pada satu node.
max_assig ned_parti tions	integer	Jumlah maksimum partisi yang ditetapkan pada satu node.

Nama kolom	Jenis data	Deskripsi
avg_assigned_partitions	bigint	Jumlah rata-rata partisi yang ditetapkan pada satu node.

Contoh kueri

Contoh berikut mendapatkan rincian pemindaian partisi untuk query terakhir selesai.

```
select query, segment, assignment, min_starttime, max_endtime, min_duration,
 avg_duration
from svl_s3partition_summary
where query = pg_last_query_id()
order by query,segment;
```

SVL_S3QUERY

Gunakan tampilan SVL_S3QUERY untuk mendapatkan detail tentang kueri data lake di segmen dan tingkat irisan node.

SVL_S3QUERY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Note**
>
> SVL_S3QUERY hanya berisi kueri yang dijalankan pada cluster utama yang disediakan. Itu tidak berisi kueri yang dijalankan pada cluster penskalaan konkurensi atau pada ruang nama tanpa server. Untuk mengakses menjelaskan rencana kueri yang dijalankan di kedua kluster utama, kluster penskalaan konkurensi, dan ruang nama tanpa server, sebaiknya gunakan tampilan pemantauan SYS. SYS_EXTERNAL_QUERY_DETAIL Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menghasilkan entri tertentu.
kueri	integer	ID kueri.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
langkah	integer	Langkah kueri yang berjalan.
simpul	integer	Nomor simpul.
mengiris	integer	Irisan data yang dilawan segmen tertentu.
waktu mulai	timestamp	Waktu di UTC kueri mulai dijalankan.
akhir waktu	timestamp	Waktu di UTC bahwa eksekusi kueri selesai
berlalu	integer	Waktu berlalu (dalam mikrodetik).
external_table_name	arang (136)	Format internal nama tabel eksternal untuk langkah pemindaian s3.
is_dipartisi	arang (1)	Jika true (`t`), nilai kolom ini menunjukkan bahwa tabel eksternal dipartisi.
is_rrscan	arang (1)	Jika true (`t`), nilai kolom ini menunjukkan bahwa pemindaian terbatas rentang diterapkan.
s3_scanned_rows	bigint	Jumlah baris yang dipindai dari Amazon S3 dan dikirim ke lapisan Redshift Spectrum. Pada kluster yang disediakan RG, ini adalah jumlah baris yang dibaca langsung dari Amazon S3 oleh pembaca asli cluster, sebelum pushdown filter.

Nama kolom	Jenis data	Deskripsi
s3_scanned_bytes	bigint	Jumlah byte yang dipindai dari Amazon S3 dan dikirim ke lapisan Redshift Spectrum. Pada cluster yang disediakan RG, ini adalah ukuran rentang pemindaian total dalam byte yang diproses oleh pembaca asli cluster.
s3query_returned_rows	bigint	Jumlah baris yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Pada cluster yang disediakan RG, ini adalah jumlah baris yang dihasilkan oleh pemindaian tabel eksternal setelah pushdown filter.
s3query_returned_bytes	bigint	Jumlah byte yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Pada cluster yang disediakan RG, ini adalah jumlah byte yang dihasilkan oleh pemindaian tabel eksternal setelah pushdown filter.
file	integer	Jumlah file yang diproses untuk langkah pemindaian S3 ini pada irisan ini.
membelah	int	Jumlah split diproses pada irisan ini. Dengan file data yang dapat dibagi besar, misalnya, file data yang lebih besar dari sekitar 512 MB, Redshift Spectrum mencoba membagi file menjadi beberapa permintaan S3 untuk pemrosesan paralel. Pada cluster yang disediakan RG, ini adalah jumlah rentang pemindaian yang dikonsumsi oleh pembaca asli cluster.
total_split_size	bigint	Ukuran total semua split diproses pada irisan ini, dalam byte. Pada cluster yang disediakan RG, ini adalah ukuran total dari semua rentang pemindaian yang dikonsumsi, dalam byte.
max_split_size	bigint	Ukuran split maksimum diproses untuk irisan ini, dalam byte. Pada cluster yang disediakan RG, ini adalah ukuran rentang pemindaian terbesar yang dikonsumsi, dalam byte.

Nama kolom	Jenis data	Deskripsi
total_retries	integer	Jumlah total percobaan ulang untuk file yang diproses. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1. Untuk percobaan ulang klien Amazon S3 di RG, lihat. STL_S3CLIENT
max_retries	integer	Jumlah maksimum percobaan ulang untuk file yang diproses individual. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1. Untuk percobaan ulang klien Amazon S3 di RG, lihat. STL_S3CLIENT
max_request_duration	integer	Durasi maksimum permintaan Redshift Spectrum individu (dalam mikrodetik). Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
avg_request_duration	double precision	Durasi rata-rata permintaan Redshift Spectrum (dalam mikrodetik). Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
max_request_parallelisme	integer	Jumlah maksimum Redshift Spectrum yang luar biasa pada irisan ini untuk langkah pemindaian S3 ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
avg_request_parallelisme	double precision	Jumlah rata-rata permintaan Spektrum Redshift paralel pada irisan ini untuk langkah pemindaian S3 ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.

Contoh kueri

Contoh berikut mendapatkan rincian langkah scan untuk query terakhir selesai.

```
select query, segment, slice, elapsed, s3_scanned_rows, s3_scanned_bytes,
 s3query_returned_rows, s3query_returned_bytes, files
from svl_s3query
```

```
where query = pg_last_query_id()
order by query,segment,slice;
```

```
query | segment | slice | elapsed | s3_scanned_rows | s3_scanned_bytes |
 s3query_returned_rows | s3query_returned_bytes | files
------+---------+-------+---------+-----------------+------------------
+----------------------+------------------------+------
 4587 |       2 |     0 |   67811 |               0 |                0 |
      0 |                      0 |     0
 4587 |       2 |     1 |  591568 |          172462 |         11260097 |
   8513 |                  170260 |     1
 4587 |       2 |     2 |  216849 |               0 |                0 |
      0 |                      0 |     0
 4587 |       2 |     3 |  216671 |               0 |                0 |
      0 |                      0 |     0
```

SVL_S3QUERY_SUMMARY

Gunakan tampilan SVL_S3QUERY_SUMMARY untuk mendapatkan ringkasan semua kueri data lake yang telah dijalankan di sistem. SVL_S3QUERY_SUMMARY mengumpulkan detail dari SVL_S3QUERY di tingkat segmen.

SVL_S3QUERY_SUMMARY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_EXTERNAL_QUERY_DETAIL pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Untuk SVCS_S3QUERY_SUMMARY, lihat. SVCS_S3QUERY_RINGKASAN

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menghasilkan entri yang diberikan.

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri. Anda dapat menggunakan nilai ini untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
xid	bigint	ID transaksi.
pid	integer	ID proses.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih langkah.
langkah	integer	Langkah kueri yang berjalan.
waktu mulai	timestamp	Waktu di UTC kueri mulai dijalankan.
akhir waktu	timestamp	Waktu di UTC bahwa kueri selesai.
berlalu	integer	Lamanya waktu yang dibutuhkan kueri untuk dijalankan (dalam mikrodetik).
digugurkan	integer	Jika kueri dihentikan oleh sistem atau dibatalkan oleh pengguna, kolom ini berisi**1**. Jika kueri berjalan hingga selesai, kolom ini berisi**0**.
external_table_name	arang (136)	Format internal nama nama eksternal tabel untuk pemindaian tabel eksternal.
file_format	karakter (16)	Format file dari data tabel eksternal.
is_dipartisi	arang (1)	Jika true (**t**), nilai kolom ini menunjukkan bahwa tabel eksternal dipartisi.
is_rrscan	arang (1)	Jika true (**t**), nilai kolom ini menunjukkan bahwa pemindaian terbatas rentang diterapkan.

Nama kolom	Jenis data	Deskripsi
is_bersarang	arang (1)	Jika true (`t`), nilai kolom ini menunjukkan bahwa tipe data kolom bersarang diakses.
s3_scanned_rows	bigint	Jumlah baris yang dipindai dari Amazon S3 dan dikirim ke lapisan Redshift Spectrum. Pada cluster yang disediakan RG, ini adalah jumlah total baris yang dipindai langsung dari Amazon S3 oleh pembaca asli cluster di semua irisan, sebelum filter pushdown.
s3_scanned_bytes	bigint	Jumlah byte yang dipindai dari Amazon S3 dan dikirim ke lapisan Redshift Spectrum, berdasarkan data terkompresi. Pada cluster yang disediakan RG, ini adalah jumlah total byte yang dipindai langsung dari Amazon S3 oleh pembaca asli cluster di semua irisan, sebelum filter pushdown.
s3query_returned_rows	bigint	Jumlah baris yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Pada cluster yang disediakan RG, ini adalah jumlah total baris yang dikembalikan oleh pembaca asli cluster di semua irisan, setelah filter pushdown.
s3query_returned_bytes	bigint	Jumlah byte yang dikembalikan dari lapisan Redshift Spectrum ke cluster. Sejumlah besar data yang dikembalikan ke Amazon Redshift dapat memengaruhi kinerja sistem. Pada cluster yang disediakan RG, ini adalah jumlah total byte yang dikembalikan oleh pembaca asli cluster di semua irisan, setelah filter pushdown.
file	integer	Jumlah file yang diproses untuk kueri data lake ini. Sejumlah kecil file membatasi manfaat pemrosesan paralel.
file_max	integer	Jumlah maksimum file yang diproses pada satu irisan.
files_avg	integer	Rata-rata jumlah file yang diproses pada satu irisan.

Nama kolom	Jenis data	Deskripsi
membelah	int	Jumlah split yang diproses untuk segmen ini. Jumlah split diproses pada irisan ini. Dengan file data yang dapat dibagi besar, misalnya, file data yang lebih besar dari sekitar 512 MB, Redshift Spectrum mencoba membagi file menjadi beberapa permintaan S3 untuk pemrosesan paralel.
splits_max	int	Jumlah maksimum split diproses pada irisan ini.
splits_avg	int	Jumlah rata-rata split diproses pada irisan ini.
total_split_size	bigint	Ukuran total semua split diproses.
max_split_size	bigint	Ukuran split maksimum diproses, dalam byte.
avg_split_size	bigint	Ukuran split rata-rata diproses, dalam byte.
total_retries	integer	Jumlah total percobaan ulang untuk satu file yang diproses individu. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1. Untuk percobaan ulang klien Amazon S3 di RG, lihat. `STL_S3CLIENT`
max_retries	integer	Jumlah maksimum percobaan ulang untuk file yang diproses. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1. Untuk percobaan ulang klien Amazon S3 di RG, lihat. `STL_S3CLIENT`
max_request_duration	integer	Durasi maksimum permintaan file individual (dalam mikrodetik). Kueri yang berjalan lama mungkin menunjukkan kemacetan. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.

Nama kolom	Jenis data	Deskripsi
avg_request_duration	double precision	Durasi rata-rata permintaan file (dalam mikrodetik). Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
max_request_parallelisme	integer	Jumlah maksimum permintaan paralel pada satu irisan untuk kueri Redshift Spectrum ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
avg_request_parallelisme	double precision	Jumlah rata-rata permintaan paralel pada satu irisan untuk kueri Redshift Spectrum ini. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
total_slowdown_count	bigint	Jumlah total permintaan Amazon S3 dengan kesalahan pelambatan yang terjadi selama pemindaian tabel eksternal. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.
max_slowdown_count	integer	Jumlah maksimum permintaan Amazon S3 dengan kesalahan pelambatan yang terjadi selama pemindaian tabel eksternal pada satu irisan. Pada cluster tipe node RG, kolom ini tidak digunakan lagi dan berisi -1.

Contoh kueri

Contoh berikut mendapatkan rincian langkah scan untuk query terakhir selesai.

```
select query, segment, elapsed, s3_scanned_rows, s3_scanned_bytes,
 s3query_returned_rows, s3query_returned_bytes, files
from svl_s3query_summary
where query = pg_last_query_id()
order by query,segment;
```

```
query | segment | elapsed | s3_scanned_rows | s3_scanned_bytes | s3query_returned_rows
 | s3query_returned_bytes | files
------+---------+---------+-----------------+------------------+----------------------
+------------------------+------
```

```
4587 |       2 |   67811 |               0 |               0 |                    0
|                    0 |     0
4587 |       2 |  591568 |          172462 |         11260097 |                 8513
|                170260 |     1
4587 |       2 |  216849 |               0 |               0 |                    0
|                    0 |     0
4587 |       2 |  216671 |               0 |               0 |                    0
|                    0 |     0
```

SVL_S3MENCOBA LAGI

Gunakan tampilan SVL_S3RETRIES untuk mendapatkan informasi tentang mengapa kueri data lake di Amazon S3 yang menggunakan Amazon Redshift Spectrum gagal.

SVL_S3RETRIES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

> **Cluster yang disediakan RG**
>
> `SVL_S3RETRIES`tidak diisi pada cluster yang disediakan RG. Tampilan ada dan tetap dapat dikueri, tetapi tidak mengembalikan baris untuk kueri yang berjalan pada kluster RG, karena RG menggunakan pembaca asli cluster untuk menanyakan Amazon S3 dan tidak menggunakan model coba lagi permintaan Spectrum.
> Untuk klien Amazon S3 coba lagi dan detail kesalahan pada kluster yang disediakan RG, gunakan dan. `STL_S3CLIENT` `STL_S3CLIENT_ERROR`

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri.
segmen	integer	Nomor segmen. Kueri terdiri dari beberapa segmen, dan setiap segmen terdiri dari satu atau lebih

Nama kolom	Jenis data	Deskripsi
		langkah. Segmen kueri dapat berjalan secara paralel. Setiap segmen berjalan dalam satu proses.
simpul	integer	Nomor simpul.
mengiris	integer	Irisan data yang dilawan segmen tertentu.
waktu acara	stempel waktu tanpa zona waktu	Waktu di UTC bahwa langkah mulai dijalanka n.
mencoba lagi	integer	Jumlah percobaan ulang untuk kueri.
successfu l_fetches	integer	Berapa kali data dikembalikan.
file_size	bigint	Ukuran file ini dalam byte.
lokasi	text	Lokasi meja.
pesan	text	Pesan kesalahan.

Contoh kueri

Contoh berikut mengambil data tentang kueri S3 gagal.

```
SELECT svl_s3retries.query, svl_s3retries.segment, svl_s3retries.node,
 svl_s3retries.slice, svl_s3retries.eventtime, svl_s3retries.retries,
svl_s3retries.successful_fetches, svl_s3retries.file_size,
 btrim((svl_s3retries."location")::text) AS "location",
 btrim((svl_s3retries.message)::text)
```

```
AS message FROM svl_s3retries;
```

SVL_SPATIAL_MENYEDERHANAKAN

Anda dapat menanyakan tampilan sistem SVL_SPATIAL_SIMPLIFY untuk mendapatkan informasi tentang objek geometri spasial yang disederhanakan menggunakan perintah COPY. Saat Anda menggunakan COPY pada shapefile, Anda dapat menentukan opsi SIMPLIFY, SIMPLIFY AUTO`tolerance`, dan SIMPLIFY AUTO ingestion. `max_tolerance` Hasil penyederhanaan dirangkum dalam tampilan sistem SVL_SPATIAL_SIMPLIFY.

Ketika SIMPLIFY AUTO `max_tolerance` diatur, tampilan ini berisi baris untuk setiap geometri yang melebihi ukuran maksimum. Ketika SIMPLIFY `tolerance` diatur, maka satu baris untuk seluruh operasi COPY disimpan. Baris ini mereferensikan ID kueri COPY dan toleransi penyederhanaan yang ditentukan.

SVL_SPATIAL_SIMPLIFY terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_SPATIAL_SIMPLIFIKASI pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	integer	ID kueri (perintah COPY) yang menghasilkan baris ini.
line_number	integer	Ketika `SIMPLIFY AUTO` opsi COPY ditentukan, nilai ini adalah nomor catatan catatan yang disederhanakan dalam shapefile.
maksimum_toleransi	double	Nilai toleransi jarak ditentukan dalam perintah COPY. Ini adalah nilai toleransi maksimum menggunakan `SIMPLIFY AUTO` opsi, atau nilai toleransi tetap menggunakan `SIMPLIFY` opsi.

Nama kolom	Jenis data	Deskripsi
initial_size	integer	Ukuran dalam byte dari nilai GEOMETRY data sebelum penyederhanaan.
disederhanakan	arang (1)	Ketika SIMPLIFY AUTO opsi COPY ditentukan, t jika geometri berhasil disederhanakan, atau f sebaliknya. Geometri mungkin tidak berhasil disederhanakan jika setelah penyederhanaan dengan toleransi maksimum yang diberikan ukurannya masih lebih besar dari ukuran geometri maksimum.
final_size	integer	Ketika SIMPLIFY AUTO opsi COPY ditentukan, ini adalah ukuran dalam byte geometri setelah penyederhanaan.
final_tolerance	double	

Contoh kueri

Query berikut mengembalikan daftar catatan yang COPY disederhanakan.

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
 final_tolerance
-------+-------------+-------------------+--------------+------------+------------
+----------------------
    20 |     1184704 |                -1 |      1513736 | t          |    1008808 |
 1.276386653895e-05
    20 |     1664115 |                -1 |      1233456 | t          |    1023584 |
 6.11707814796635e-06
```

SVL_SPECTRUM_SCAN_ERROR

Anda dapat menanyakan tampilan sistem SVL_SPECTRUM_SCAN_ERROR untuk mendapatkan informasi tentang kesalahan pemindaian Redshift Spectrum.

SVL_SPECTRUM_SCAN_ERROR terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_EXTERNAL_QUERY_ERROR pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Menampilkan contoh kesalahan yang dicatat. Defaultnya adalah 10 entri per kueri.

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang menghasilkan baris ini.
kueri	integer	ID kueri yang menghasilkan baris ini.
lokasi	karakter (128)	Lokasi data yang ditanyakan.
rowid	karakter (128)	Lokasi kesalahan dalam file. `rowid`Bagian-bagian dipisahkan dengan `:` (titik dua) dan bagian tambahan dapat ditambahkan di masa depan. `row_offset :row_group :row_id` Row_offset adalah offset (dalam byte) dari baris dalam file dan diatur ke untuk format file yang tidak didukung. `-1` Sebuah tabel dibagi menjadi row_groups, dan setiap grup memiliki baris dengan row_ids yang berbeda.
nama	karakter (128)	Nama kolom yang dikembalikan oleh kueri.
nilai_asli	karakter (128)	Nilai asli ditanyakan.
modified_value	karakter (128)	Nilai yang dimodifikasi dikembalikan berdasarkan opsi konfigurasi penanganan data yang ditentukan dalam kueri.

Nama kolom	Jenis data	Deskripsi
pemicu	karakter (128)	Opsi penanganan data yang ditentukan dalam kueri.
tindakan	karakter (128)	Tindakan yang terkait dengan opsi penanganan data yang ditentukan dalam kueri.
action_value	karakter (128)	Nilai parameter tindakan yang terkait dengan opsi penanganan data yang ditentukan dalam kueri.
error_code	integer	Kode hasil dari opsi penanganan data yang ditentukan dalam kueri.

Contoh kueri

Query berikut mengembalikan daftar baris untuk operasi penanganan data yang dilakukan.

```
SELECT * FROM svl_spectrum_scan_error;
```

Query mengembalikan hasil yang mirip dengan berikut ini.

```
  userid  query     location                                    rowid   colname
         original_value            modified_value       trigger          action
                 action_valueerror_code
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0     league_name
         Barclays Premier League    Barclays Premier Lea UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0     league_nspi
          34595                     32767                UNSPECIFIED
OVERFLOW_VALUE                                  199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:1     league_nspi
          34151                     32767                UNSPECIFIED
OVERFLOW_VALUE                                  199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2     league_name
         Barclays Premier League    Barclays Premier Lea UNSPECIFIED      TRUNCATE
                                    156
```

```
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2     league_nspi
          33223                      32767               UNSPECIFIED
OVERFLOW_VALUE                                  199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3     league_name
         Barclays Premier League    Barclays Premier Lea UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3     league_nspi
          32808                      32767               UNSPECIFIED
OVERFLOW_VALUE                                  199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:4     league_nspi
          32790                      32767               UNSPECIFIED
OVERFLOW_VALUE                                  199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:5     league_name
         Spanish Primera Division   Spanish Primera Divi UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:6     league_name
         Spanish Primera Division   Spanish Primera Divi UNSPECIFIED      TRUNCATE
                                    156
```

SVL_STATEMENTTEXT

Gunakan tampilan SVL_STATEMENTTEXT untuk mendapatkan catatan lengkap dari semua perintah SQL yang telah dijalankan pada sistem.

Tampilan SVL_STATEMENTTEXT berisi gabungan semua baris dalam,, dan tabel. STL_DDLTEXT STL_QUERYTEXT STL_UTILITYTEXT Tampilan ini juga mencakup gabungan ke tabel STL_QUERY.

SVL_STATEMENTTEXT dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_HISTORY pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang membuat entri.
xid	bigint	ID transaksi yang terkait dengan pernyataan.
pid	integer	ID proses untuk pernyataan tersebut.
label	karakter (320)	Entah nama file yang digunakan untuk menjalankan kueri atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, bidang ini kosong.
waktu mulai	timestamp	Waktu yang tepat ketika pernyataan mulai dijalankan, dengan 6 digit presisi untuk detik pecahan. Misalnya: **2009-06-12 11:29:19.131358**
akhir waktu	timestamp	Waktu yang tepat ketika pernyataan selesai dieksekusi, dengan 6 digit presisi untuk detik pecahan. Misalnya: **2009-06-12 11:29:19.193640**
urutan	integer	Ketika satu pernyataan berisi lebih dari 200 karakter, baris tambahan dicatat untuk pernyataan itu. Urutan 0 adalah baris pertama, 1 adalah yang kedua, dan seterusnya.
jenis	varchar (10)	Jenis pernyataan SQL:**QUERY**,**DDL**, atau**UTILITY**.
text	karakter (200)	Teks SQL, dalam peningkatan 200 karakter. Bidang ini mungkin berisi karakter khusus seperti garis miring terbalik (\\) dan baris baru (). \n

Contoh kueri

Query berikut mengembalikan pernyataan DDL yang dijalankan pada 16 Juni 2009:

```
select starttime, type, rtrim(text) from svl_statementtext
where starttime like '2009-06-16%' and type='DDL' order by starttime asc;

starttime                  | type |              rtrim
---------------------------|------|--------------------------------
2009-06-16 10:36:50.625097 | DDL  | create table ddltest(c1 int);
2009-06-16 15:02:16.006341 | DDL  | drop view alltickitjoin;
2009-06-16 15:02:23.65285  | DDL  | drop table sales;
2009-06-16 15:02:24.548928 | DDL  | drop table listing;
2009-06-16 15:02:25.536655 | DDL  | drop table event;
...
```

Merekonstruksi SQL yang tersimpan

Untuk merekonstruksi SQL yang disimpan di `text` kolom SVL_STATEMENTTEXT, jalankan pernyataan SELECT untuk membuat SQL dari 1 atau lebih bagian dalam kolom. `text` Sebelum menjalankan SQL yang direkonstruksi, ganti setiap (\n) karakter khusus dengan baris baru. Hasil dari pernyataan SELECT berikut adalah baris SQL direkonstruksi di lapangan. `query_statement`

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS query_statement
from SVL_STATEMENTTEXT where pid=pg_backend_pid();
```

Misalnya, query berikut memilih 3 kolom. Kueri itu sendiri lebih panjang dari 200 karakter dan disimpan di bagian-bagian dalam SVL_STATEMENTTEXT.

```
select
1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,
3 AS b01234567890123456789012345678901234567890123456789012345678901234
FROM stl_querytext;
```

Dalam contoh ini, kueri disimpan dalam 2 bagian (baris) di `text` kolom SVL_STATEMENTTEXT.

```
select sequence, text from SVL_STATEMENTTEXT where pid = pg_backend_pid() order by
 starttime, sequence;
```

```
 sequence |
                text
```

```
----------
+-----------------------------------------------------------------------------------------------
        0 | select\n1 AS
 a01234567890123456789012345678901234567890123456789012345678901234567890,\n2 AS
 b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b01234567890123456789012345678901234567890123456789012345678901234
        1 | \nFROM stl_querytext;
```

Untuk merekonstruksi SQL yang disimpan di STL_STATEMENTTEXT, jalankan SQL berikut.

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS text
from SVL_STATEMENTTEXT where pid=pg_backend_pid();
```

Untuk menggunakan SQL yang direkonstruksi yang dihasilkan di klien Anda, ganti karakter khusus (\n) apa pun dengan baris baru.

```


                    text


-----------------------------------------------------------------------------------------------
 select\n1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
\n2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b01234567890123456789012345678901234567890123456789012345678901234\nFROM stl_querytext;
```

SVL_STORED_PROC_CALL

Anda dapat menanyakan tampilan sistem SVL_STORED_PROC_CALL untuk mendapatkan informasi tentang panggilan prosedur tersimpan, termasuk waktu mulai, waktu akhir, dan apakah panggilan dibatalkan. Setiap panggilan prosedur yang disimpan menerima ID kueri.

SVL_STORED_PROC_CALL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_PROSEDUR_PANGGILAN pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang hak istimewanya digunakan untuk menjalankan pernyataan. Jika panggilan ini bersarang dalam prosedur tersimpan SECURITY DEFINER, maka ini adalah userid pemilik prosedur tersimpan tersebut.
session_userid	integer	ID pengguna yang membuat sesi dan merupakan pemanggil dari panggilan prosedur tersimpan tingkat atas.
kueri	integer	ID kueri panggilan prosedur.
label	karakter (320)	Entah nama file yang digunakan untuk menjalankan kueri atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, nilai bidang ini adalah default.
xid	bigint	ID transaksi.
pid	integer	ID proses. Biasanya, semua kueri dalam sesi dijalankan dalam proses yang sama, jadi nilai ini biasanya tetap konstan jika Anda menjalankan serangkaian kueri dalam sesi yang sama. Setelah peristiwa internal tertentu, Amazon Redshift mungkin memulai ulang sesi aktif dan menetapkan nilai pid baru. Untuk informasi selengkapnya, lihat STL_RESTARTED_SESSIONS.
basis data	karakter (32)	Nama database yang terhubung dengan pengguna saat kueri dikeluarkan.
querytxt	karakter (4000)	Teks sebenarnya dari permintaan panggilan prosedur.
waktu mulai	timestamp	Waktu di UTC kueri mulai berjalan, dengan enam digit presisi untuk detik pecahan, misalnya: **2009-06-12 11:29:19.131358.**

Nama kolom	Jenis data	Deskripsi
akhir waktu	timestamp	Waktu di UTC bahwa kueri selesai berjalan, dengan enam digit presisi untuk detik pecahan, misalnya: **2009-06-12 11:29:19.131358.**
digugurkan	integer	Jika prosedur yang disimpan dihentikan oleh sistem atau dibatalkan oleh pengguna, kolom ini berisi 1. Jika panggilan berjalan hingga selesai, kolom ini berisi 0.
dari_sp_call	integer	Jika panggilan prosedur dipanggil oleh panggilan prosedur lain, kolom ini berisi ID kueri panggilan luar. Jika tidak, bidangnya adalah NULL.

Contoh kueri

Kueri berikut mengembalikan waktu yang telah berlalu dalam urutan menurun dan status penyelesaian untuk panggilan prosedur tersimpan dalam satu hari terakhir.

```
select query, datediff(seconds, starttime, endtime) as elapsed_time, aborted,
 trim(querytxt) as call from svl_stored_proc_call where starttime >= getdate() -
 interval '1 day' order by 2 desc;

  query | elapsed_time | aborted |                                          call
--------+--------------+---------
+-------------------------------------------------------------------------------
   4166 |            7 |       0 | call search_batch_status(35,'succeeded');
   2433 |            3 |       0 | call test_batch (123456)
   1810 |            1 |       0 | call prod_benchmark (123456)
   1836 |            1 |       0 | call prod_testing (123456)
   1808 |            1 |       0 | call prod_portfolio ('N', 123456)
   1816 |            1 |       1 | call prod_portfolio ('Y', 123456)
```

SVL_STORED_PROC_MESSAGES

Anda dapat menanyakan tampilan sistem SVL_STORED_PROC_MESSAGES untuk mendapatkan informasi tentang pesan prosedur yang disimpan. Pesan yang diangkat dicatat bahkan jika panggilan prosedur yang disimpan dibatalkan. Setiap panggilan prosedur yang disimpan menerima ID kueri.

Untuk informasi selengkapnya tentang cara menyetel level minimum untuk pesan yang dicatat, lihat stored_proc_log_min_messages.

SVL_STORED_PROC_MESSAGES terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_PROSEDUR_PESAN-PESAN pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
userid	integer	ID pengguna yang hak istimewanya digunakan untuk menjalankan pernyataan. Jika panggilan ini bersarang dalam prosedur tersimpan SECURITY DEFINER, maka ini adalah userid pemilik prosedur tersimpan tersebut.
session_userid	integer	ID pengguna yang membuat sesi dan merupakan pemanggil dari panggilan prosedur tersimpan tingkat atas.
pid	integer	ID proses.
xid	bigint	ID transaksi dari permintaan panggilan prosedur.
kueri	integer	ID kueri panggilan prosedur.
rekor waktu	timestamp	Waktu di UTC bahwa pesan itu diangkat.
loglevel	integer	Nilai numerik dari tingkat log dari pesan yang diangkat. Nilai yang mungkin: 20 - untuk LOG 30 - untuk INFO 40 - untuk PEMBERITAHUAN 50 - untuk PERINGATAN 60 - untuk PENGECUALIAN

Nama kolom	Jenis data	Deskripsi
loglevel_text	karakter (10)	Tingkat log yang sesuai dengan nilai numerik di loglevel. Nilai yang mungkin: LOG, INFO, PEMBERITAHUAN, PERINGATAN, dan PENGECUALIAN.
pesan	karakter (1024)	Teks dari pesan yang diangkat.
linenum	integer	Nomor baris pernyataan yang diangkat.
querytext	karakter (500)	Teks sebenarnya dari permintaan panggilan prosedur.
label	karakter (320)	Entah nama file yang digunakan untuk menjalankan kueri atau label yang ditentukan dengan perintah SET QUERY_GROUP. Jika kueri tidak berbasis file atau parameter QUERY_GROUP tidak disetel, nilai bidang ini adalah default.
digugurkan	integer	Jika prosedur yang disimpan dihentikan oleh sistem atau dibatalkan oleh pengguna, kolom ini berisi 1. Jika panggilan berjalan hingga selesai, kolom ini berisi 0.
message_xid	bigint	ID transaksi dari pesan yang diangkat.

Contoh kueri

Pernyataan SQL berikut menunjukkan cara menggunakan SVL_STORED_PROC_MESSAGES untuk meninjau pesan yang dimunculkan.

```
-- Create and run a stored procedure
CREATE OR REPLACE PROCEDURE test_proc1(f1 int) AS
$$
BEGIN
    RAISE INFO 'Log Level: Input f1 is %',f1;
    RAISE NOTICE 'Notice Level: Input f1 is %',f1;
    EXECUTE 'select invalid';
    RAISE NOTICE 'Should not print this';
```

```
EXCEPTION WHEN OTHERS THEN
     raise exception 'EXCEPTION level: Exception Handling';
END;
$$ LANGUAGE plpgsql;

-- Call this stored procedure
CALL test_proc1(2);

-- Show raised messages with level higher than INFO
SELECT query, recordtime, loglevel, loglevel_text, trim(message) as message, aborted
 FROM svl_stored_proc_messages
  WHERE loglevel > 30 AND query = 193 ORDER BY recordtime;

 query |         recordtime         | loglevel | loglevel_text |               message
              | aborted
-------+----------------------------+----------+---------------
+-------------------------------------+---------
   193 | 2020-03-17 23:57:18.277196 |       40 | NOTICE        | Notice Level: Input f1
 is 2         |       1
   193 | 2020-03-17 23:57:18.277987 |       60 | EXCEPTION     | EXCEPTION level:
 Exception Handling |       1
(2 rows)

-- Show raised messages at EXCEPTION level
SELECT query, recordtime, loglevel, loglevel_text, trim(message) as message, aborted
 FROM svl_stored_proc_messages
  WHERE loglevel_text = 'EXCEPTION' AND query = 193 ORDER BY recordtime;

 query |         recordtime         | loglevel | loglevel_text |               message
              | aborted
-------+----------------------------+----------+---------------
+-------------------------------------+---------
   193 | 2020-03-17 23:57:18.277987 |       60 | EXCEPTION     | EXCEPTION level:
 Exception Handling |       1
```

Pernyataan SQL berikut menunjukkan cara menggunakan SVL_STORED_PROC_MESSAGES untuk meninjau pesan yang diangkat dengan opsi SET saat membuat prosedur tersimpan. Karena test_proc () memiliki tingkat log minimum PEMBERITAHUAN, hanya pesan tingkat PEMBERITAHUAN, PERINGATAN, dan PENGECUALIAN yang masuk SVL_STORED_PROC_MESSAGES.

```
-- Create a stored procedure with minimum log level of NOTICE
CREATE OR REPLACE PROCEDURE test_proc() AS
```

```
$$
BEGIN
    RAISE LOG 'Raise LOG messages';
    RAISE INFO 'Raise INFO messages';
    RAISE NOTICE 'Raise NOTICE messages';
    RAISE WARNING 'Raise WARNING messages';
    RAISE EXCEPTION 'Raise EXCEPTION messages';
    RAISE WARNING 'Raise WARNING messages again'; -- not reachable
END;
$$ LANGUAGE plpgsql SET stored_proc_log_min_messages = NOTICE;

-- Call this stored procedure
CALL test_proc();

-- Show the raised messages
SELECT query, recordtime, loglevel_text, trim(message) as message, aborted FROM
 svl_stored_proc_messages
  WHERE query = 149 ORDER BY recordtime;

 query |         recordtime         | loglevel_text |          message         |
 aborted
-------+----------------------------+---------------+--------------------------
+---------
   149 | 2020-03-16 21:51:54.847627 | NOTICE        | Raise NOTICE messages    |
 1
   149 | 2020-03-16 21:51:54.84766  | WARNING       | Raise WARNING messages   |
 1
   149 | 2020-03-16 21:51:54.847668 | EXCEPTION     | Raise EXCEPTION messages |
 1
(3 rows)
```

SVL_TERMINATE

Mencatat waktu ketika pengguna membatalkan atau mengakhiri proses.

PILIH PG_TERMINATE_BACKEND (pid), SELECT PG_CANCEL_BACKEND (pid), dan CANCEL pid membuat entri log di SVL_TERMINATE.

SVL_TERMINATE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_QUERY_HISTORY pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan

dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
pid	integer	ID proses dari proses yang dibatalkan atau dihentikan.
waktu acara	timestamp	Waktu ketika proses dibatalkan atau dihentikan.
userid	integer	ID pengguna pengguna yang menjalankan perintah.
jenis	string	Jenis penghentian. Bisa jadi CANCEL atau TERMINATE.

Perintah berikut menunjukkan kueri terbaru yang dibatalkan.

```
select * from svl_terminate order by eventtime desc limit 1;
 pid  |         eventtime          | userid |  type
------+----------------------------+--------+--------
 8324 | 2020-03-24 09:42:07.298937 |      1 | CANCEL
(1 row)
```

SVL_UDF_LOG

Merekam kesalahan yang ditentukan sistem dan pesan peringatan yang dihasilkan selama eksekusi fungsi yang ditentukan pengguna (UDF).

SVL_UDF_LOG dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_UDF_LOG pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
kueri	bigint	ID kueri. Anda dapat menggunakan ID ini untuk bergabung dengan berbagai tabel dan tampilan sistem lainnya.
pesan	arang (4096)	Pesan yang dihasilkan oleh fungsi.
dibuat	timestamp	Waktu log dibuat.
traceback	arang (4096)	Jika tersedia, nilai ini menyediakan stack traceback untuk UDF. Untuk informasi selengkapnya, lihat traceback di Python Standard Library.
nama fungsi	karakter (256)	Nama UDF yang mengeksek usi.
simpul	integer	Node tempat pesan dihasilka n.
mengiris	integer	Potongan tempat pesan dihasilkan.
seq	integer	Urutan pesan pada irisan.

Kueri Sampel

Contoh berikut menunjukkan bagaimana UDF menangani kesalahan yang ditentukan sistem. Blok pertama menunjukkan definisi untuk fungsi UDF yang mengembalikan kebalikan dari argumen. Ketika Anda menjalankan fungsi dan memberikan argumen 0, seperti yang ditunjukkan blok kedua, fungsi mengembalikan kesalahan. Pernyataan ketiga membaca pesan kesalahan yang masuk SVL_UDF_LOG

```
-- Create a function to find the inverse of a number

CREATE OR REPLACE FUNCTION  f_udf_inv(a int)
  RETURNS float IMMUTABLE
AS $$
   return 1/a
$$ LANGUAGE plpythonu;

-- Run the function with a 0 argument to create an error
Select f_udf_inv(0) from sales;

-- Query SVL_UDF_LOG to view the message

Select query, created, message::varchar
from svl_udf_log;

 query |          created           | message
-------+----------------------------
+------------------------------------------------------------
  2211 | 2015-08-22 00:11:12.04819  | ZeroDivisionError: long division or modulo by
 zero\nNone
```

Contoh berikut menambahkan logging dan pesan peringatan ke UDF sehingga operasi pembagian dengan nol menghasilkan pesan peringatan alih-alih berhenti dengan pesan kesalahan.

```
-- Create a function to find the inverse of a number and log a warning

CREATE OR REPLACE FUNCTION f_udf_inv_log(a int)
  RETURNS float IMMUTABLE
 AS $$
  import logging
  logger = logging.getLogger() #get root logger
  if a==0:
    logger.warning('You attempted to divide by zero.\nReturning zero instead of error.
\n')
    return 0
  else:
     return 1/a
$$ LANGUAGE plpythonu;
```

Contoh berikut menjalankan fungsi, kemudian query SVL_UDF_LOG untuk melihat pesan.

```
-- Run the function with a 0 argument to trigger the warning
Select f_udf_inv_log(0) from sales;

-- Query SVL_UDF_LOG to view the message

Select query, created, message::varchar
from svl_udf_log;

query |          created           | message
------+----------------------------+---------------------------------
    0 | 2015-08-22 00:11:12.04819  | You attempted to divide by zero.
                                     Returning zero instead of error.
```

SVL_USER_INFO

Anda dapat mengambil data tentang pengguna database Amazon Redshift dengan tampilan SVL_USER_INFO.

SVL_USER_INFO hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama pengguna	text	Nama pengguna untuk peran tersebut.
usesisid	integer	ID pengguna untuk pengguna.
digunakan createdb	boolean	Nilai yang menunjukkan apakah pengguna memiliki izin untuk membuat database.
menggunak ansuper	boolean	Nilai yang menunjukkan apakah pengguna adalah superuser.
menggunak ancatupd	boolean	Nilai yang menunjukkan apakah pengguna dapat memperbarui katalog sistem.

Nama kolom	Jenis data	Deskripsi
useconnlimit	text	Jumlah koneksi yang dapat dibuka pengguna.
syslogaccess	text	Nilai yang menunjukkan apakah pengguna memiliki akses ke log sistem. Dua nilai yang mungkin adalah `RESTRICTED` danUNRESTRICTED . `RESTRICTED` berarti bahwa pengguna yang bukan pengguna super dapat melihat catatan mereka sendiri. `UNRESTRICTED` berarti bahwa pengguna yang bukan pengguna super dapat melihat semua catatan dalam tampilan sistem dan tabel tempat mereka memiliki `SELECT` hak istimewa.
last_ddl_ts	timestamp	Stempel waktu untuk bahasa definisi data terakhir (DDL) membuat pernyataan yang dijalankan oleh pengguna.
sessiontimeout	integer	Waktu maksimum dalam detik sesi tetap tidak aktif atau tidak aktif sebelum waktu habis. 0 menunjukkan bahwa tidak ada batas waktu yang ditetapkan. Untuk informasi tentang setelan batas waktu siaga atau tidak aktif klaster, lihat Kuota dan batas di Amazon Redshift di Panduan Manajemen Pergeseran Merah Amazon.
external_id	text	Pengidentifikasi unik pengguna di penyedia identitas pihak ketiga.

Kueri Sampel

Perintah berikut mengambil informasi pengguna dari SVL_USER_INFO.

```
SELECT * FROM SVL_USER_INFO;
```

SVL_VACUUM_PERCENTAGE

Tampilan SVL_VACUUM_PERSENTASE melaporkan persentase blok data yang dialokasikan ke tabel setelah melakukan vakum. Angka persentase ini menunjukkan berapa banyak ruang disk yang direklamasi. Lihat VAKUM perintah untuk informasi lebih lanjut tentang utilitas vakum.

SVL_VACUUM_PERSENTASE hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Beberapa atau semua data dalam tabel ini juga dapat ditemukan di tampilan SYS_VACUUM_HISTORY pemantauan SYS. Data dalam tampilan pemantauan SYS diformat agar lebih mudah digunakan dan dipahami. Kami menyarankan Anda menggunakan tampilan pemantauan SYS untuk pertanyaan Anda.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
xid	bigint	ID Transaksi untuk pernyataan vakum.
table_id	integer	ID tabel untuk tabel yang disedot debu.
persentase	bigint	Persentase blok data setelah vakum (relatif terhadap jumlah blok dalam tabel sebelum vakum dijalankan).

Contoh kueri

Kueri berikut menampilkan persentase untuk operasi tertentu pada tabel 100238:

```
select * from svl_vacuum_percentage
where table_id=100238 and xid=2200;

xid  | table_id | percentage
-----+----------+------------
1337 |   100238 |         60
(1 row)
```

Setelah operasi vakum ini, tabel berisi 60 persen dari blok asli.

Tabel katalog sistem

Topik

- PG_ATTRIBUTE_INFO

- PG_CLASS_INFO
- PG_DATABASE_INFO
- PG_DEFAULT_ACL
- PG_EXTERNAL_SCHEMA
- PG_LIBRARY
- PG_PROC_INFO
- PG_STATISTIC_INDICATOR
- PG_TABLE_DEF
- PG_USER_INFO
- Menanyakan tabel katalog

Katalog sistem menyimpan metadata skema, seperti informasi tentang tabel dan kolom. Tabel katalog sistem memiliki awalan PG.

Tabel katalog PostgreSQL standar dapat diakses oleh pengguna Amazon Redshift. Untuk informasi selengkapnya tentang katalog sistem PostgreSQL, lihat tabel sistem PostgreSQL

PG_ATTRIBUTE_INFO

PG_ATTRIBUTE_INFO adalah tampilan sistem Amazon Redshift yang dibangun di atas tabel katalog PostgreSQL PG_ATTRIBUTE dan tabel katalog internal PG_ATTRIBUTE_ACL. PG_ATTRIBUTE_INFO mencakup rincian tentang kolom tabel atau tampilan, termasuk daftar kontrol akses kolom, jika ada.

Kolom tabel

PG_ATTRIBUTE_INFO menunjukkan kolom berikut selain kolom di PG_ATTRIBUTE.

Nama kolom	Jenis data	Deskripsi
menyerang	aklitem []	Hak akses tingkat kolom, jika ada, yang telah diberikan secara khusus pada kolom ini.

PG_CLASS_INFO

PG_CLASS_INFO adalah tampilan sistem Amazon Redshift yang dibangun di atas tabel katalog PostgreSQL PG_CLASS dan PG_CLASS_EXTENDED. PG_CLASS_INFO mencakup rincian tentang waktu pembuatan tabel dan gaya distribusi saat ini. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.

PG_CLASS_INFO dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

PG_CLASS_INFO menunjukkan kolom berikut selain kolom di PG_CLASS. `oid`Kolom di PG_CLASS dipanggil `reloid` dalam tabel PG_CLASS_INFO.

Nama kolom	Jenis data	Deskripsi
relkreasi waktu	timestamp	Waktu di UTC bahwa tabel dibuat.
releffect ivediststyle	integer	Gaya distribusi tabel atau, jika tabel menggunakan distribusi otomatis, gaya distribusi saat ini ditetapkan oleh Amazon Redshift.

Kolom RELEFFECTIVEDISTSTYLE di PG_CLASS_INFO menunjukkan gaya distribusi saat ini untuk tabel. Jika tabel menggunakan distribusi otomatis, RELEFFECTIVEDISTSTYLE adalah 10, 11, atau 12, yang menunjukkan apakah gaya distribusi efektif adalah AUTO (ALL), AUTO (EVEN), atau AUTO (KEY). Jika tabel menggunakan distribusi otomatis, gaya distribusi mungkin awalnya menampilkan AUTO (ALL), lalu ubah ke AUTO (EVEN) ketika tabel tumbuh atau AUTO (KEY) jika kolom ditemukan berguna sebagai kunci distribusi.

Tabel berikut memberikan gaya distribusi untuk setiap nilai dalam kolom RELEFFECTIVEDISTSTYLE:

RELEFFECTIVEDISTSTYLE	Gaya distribusi saat ini
0	PUN

RELEFFECTIVEDISTSTYLE	Gaya distribusi saat ini
1	KUNCI
8	SEMUA
10	OTOMATIS (SEMUA)
11	OTOMATIS (DATAR)
12	OTOMATIS (KUNCI)

Contoh

Query berikut mengembalikan gaya distribusi tabel saat ini dalam katalog.

```
select reloid as tableid,trim(nspname) as schemaname,trim(relname) as
 tablename,reldiststyle,releffectivediststyle,
CASE WHEN "reldiststyle" = 0 THEN 'EVEN'::text
     WHEN "reldiststyle" = 1 THEN 'KEY'::text
     WHEN "reldiststyle" = 8 THEN 'ALL'::text
     WHEN "releffectivediststyle" = 10 THEN 'AUTO(ALL)'::text
     WHEN "releffectivediststyle" = 11 THEN 'AUTO(EVEN)'::text
     WHEN "releffectivediststyle" = 12 THEN 'AUTO(KEY)'::text ELSE '<<UNKNOWN>>'::text
 END as diststyle,relcreationtime
from pg_class_info a left join pg_namespace b on a.relnamespace=b.oid;
```

```
 tableid | schemaname | tablename | reldiststyle | releffectivediststyle | diststyle  |
      relcreationtime
---------+------------+-----------+--------------+-----------------------+------------
+----------------------------
 3638033 | public     | customer  |            0 |                     0 | EVEN       |
 2019-06-13 15:02:50.666718
 3638037 | public     | sales     |            1 |                     1 | KEY        |
 2019-06-13 15:03:29.595007
 3638035 | public     | lineitem  |            8 |                     8 | ALL        |
 2019-06-13 15:03:01.378538
 3638039 | public     | product   |            9 |                    10 | AUTO(ALL)  |
 2019-06-13 15:03:42.691611
```

```
 3638041 | public     | shipping  |           9 |                    11 | AUTO(EVEN) |
 2019-06-13 15:03:53.69192
 3638043 | public     | support   |           9 |                    12 | AUTO(KEY)  |
 2019-06-13 15:03:59.120695
(6 rows)
```

PG_DATABASE_INFO

PG_DATABASE_INFO adalah tampilan sistem Amazon Redshift yang memperluas tabel katalog PostgreSQL PG_DATABASE.

PG_DATABASE_INFO dapat dilihat oleh semua pengguna.

Kolom tabel

PG_DATABASE_INFO berisi kolom berikut selain kolom di PG_DATABASE. `oid`Kolom di PG_DATABASE disebut dalam tabel `datid` PG_DATABASE_INFO. Untuk informasi selengkapnya, lihat Dokumentasi PostgreSQL.

Nama kolom	Jenis data	Deskripsi
datid	oid	Object identifier (OID) yang digunakan secara internal oleh tabel sistem.
datconnlimit	text	Jumlah maksimum koneksi bersamaan yang dapat dibuat ke database ini. Nilai -1 berarti tidak ada batas.

PG_DEFAULT_ACL

Menyimpan informasi tentang hak akses default. Untuk informasi selengkapnya tentang hak akses default, lihatUBAH HAK ISTIMEWA DEFAULT.

PG_DEFAULT_ACL terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
defacluser	integer	ID pengguna yang menerapkan hak istimewa yang terdaftar.
defaclnamespace	oid	ID objek skema di mana hak istimewa default diterapkan. Nilai default adalah 0 jika tidak ada skema yang ditentukan.
defaclobjtype	karakter	Jenis objek yang hak istimewa default diterapkan. Nilai yang valid adalah sebagai berikut: • r — relasi (tabel atau tampilan) • f - fungsi • p — prosedur tersimpan
defaclacl	aklitem []	String yang mendefinisikan hak istimewa default untuk pengguna tertentu atau kelompok pengguna dan tipe objek. Jika hak istimewa diberikan kepada pengguna, string dalam bentuk berikut: *{ username=privilegestring/grantor }* username Nama pengguna yang hak istimewanya diberikan. Jika nama pengguna dihilangkan, hak istimewa diberikan kepada PUBLIC. Jika hak istimewa diberikan kepada grup pengguna, string dalam bentuk berikut: *{ "group groupname=privilegestring/grantor" }* privilegestring

Nama kolom	Jenis data	Deskripsi
		Sebuah string yang menentukan hak istimewa yang diberikan. Nilai yang valid adalah: • a— INSERT (tambahkan) • r— PILIH (baca) • w— PEMBARUAN (tulis) • d— HAPUS • R— ATURAN • x— Memberikan hak istimewa untuk membuat kendala kunci asing (REFERENSI). • t— PEMICU • X— MENGEKSEKUSI • U— PENGGUNAAN • C— BUAT • T— BUAT TEMP • D— JATUHKAN • P— MEMOTONG • A— MENGUBAH • *— Menunjukkan bahwa pengguna yang menerima hak istimewa sebelumnya pada gilirannya dapat memberikan hak istimewa yang sama kepada orang lain (DENGAN OPSI PEMBERIAN). String yang memegang semua karakter kode hak istimewa, dalam urutan berdasarkan posisi bitmask, terlihat seperti “ArwdRxtxuctDPA”. pemberi Nama pengguna yang memberikan hak istimewa.

Nama kolom	Jenis data	Deskripsi
		Contoh berikut menunjukkan bahwa pengguna admin memberikan semua hak istimewa, termasuk DENGAN OPSI PEMBERIAN, kepada penggunadbuser. `dbuser=r*a*w*d*x*X*/admin`

Contoh

Query berikut mengembalikan semua hak istimewa default didefinisikan untuk database.

```
select pg_get_userbyid(d.defacluser) as user,
n.nspname as schema,
case d.defaclobjtype when 'r' then 'tables' when 'f' then 'functions' end
as object_type,
array_to_string(d.defaclacl, ' + ')  as default_privileges
from pg_catalog.pg_default_acl d
left join pg_catalog.pg_namespace n on n.oid = d.defaclnamespace;

 user  | schema | object_type |              default_privileges
-------+--------+-------------+-------------------------------------------------------
 admin | tickit | tables      | user1=r/admin + "group group1=a/admin" + user2=w/admin
```

Hasil dalam contoh sebelumnya menunjukkan bahwa untuk semua tabel baru yang dibuat oleh pengguna admin dalam tickit skema, admin memberikan hak SELECT untuk, INSERT privileges touser1, dan UPDATE privileges untukgroup1. user2

PG_EXTERNAL_SCHEMA

Menyimpan informasi tentang skema eksternal.

PG_EXTERNAL_SCHEMA terlihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat metadata yang dapat mereka akses. Untuk informasi selengkapnya, lihat BUAT SKEMA EKSTERNAL.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
esoid	oid	ID skema eksternal.
eskind	integer	Jenis skema eksternal.
esdbname	text	Nama database eksternal.
esoptions	text	Opsi skema eksternal.

Contoh

Contoh berikut menunjukkan rincian untuk skema eksternal.

```
select esoid, nspname as schemaname, nspowner, esdbname as external_db, esoptions
from pg_namespace a,pg_external_schema b where a.oid=b.esoid;

esoid  | schemaname      | nspowner | external_db | esoptions

-------+-----------------+----------+-------------
+-------------------------------------------------------------
100134 | spectrum_schema |      100 | spectrum_db |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
100135 | spectrum        |      100 | spectrumdb  |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
100149 | external        |      100 | external_db |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

PG_LIBRARY

Menyimpan informasi tentang pustaka yang ditentukan pengguna.

PG_LIBRARY dapat dilihat oleh semua pengguna. Pengguna super dapat melihat semua baris; pengguna biasa hanya dapat melihat data mereka sendiri. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama	name	Nama perpustakaan.
bahasa_oid	oid	Dicadangkan untuk penggunaan sistem.
file_store_id	integer	Dicadangkan untuk penggunaan sistem.
owner	integer	User ID dari pemilik perpustakaan.

Contoh

Contoh berikut mengembalikan informasi untuk pustaka yang diinstal pengguna.

```
select * from pg_library;

name       | language_oid | file_store_id | owner
-----------+--------------+---------------+------
f_urlparse |       108254 |          2000 |   100
```

PG_PROC_INFO

PG_PROC_INFO adalah tampilan sistem Amazon Redshift yang dibangun di atas tabel katalog PostgreSQL PG_PROC dan tabel katalog internal PG_PROC_EXTENDED. PG_PROC_INFO mencakup rincian tentang prosedur dan fungsi yang disimpan, termasuk informasi yang terkait dengan argumen keluaran, jika ada.

Kolom tabel

PG_PROC_INFO menunjukkan kolom berikut selain kolom di PG_PROC. `oid`Kolom di PG_PROC disebut `prooid` dalam tabel PG_PROC_INFO.

Nama kolom	Jenis data	Deskripsi
prooid	oid	ID objek dari fungsi atau prosedur yang disimpan.
prokind	“arang”	Nilai yang menunjukkan jenis fungsi atau prosedur yang disimpan. Nilai ini adalah 'f' untuk fungsi reguler, 'p' untuk prosedur tersimpan, dan 'a' untuk fungsi agregat.
proargmode	“arang" []	Array dengan mode argumen prosedur, dikodekan sebagai 'i' untuk argumen IN, 'o' untuk argumen OUT, dan 'b' untuk argumen INOUT. Jika semua argumen adalah argumen IN, bidang ini adalah NULL. Subskrip sesuai dengan posisi dalam larik proallargtypes.
proallargtipe	oid []	Sebuah array dengan tipe data dari argumen prosedur. Array ini mencakup semua jenis argumen (termasuk argumen OUT dan INOUT). Namun, jika semua argumen adalah argumen IN, bidang ini adalah NULL. Berlangganan berbasis satu. Sebaliknya, proargtypes berlangganan dari nol.

Proargnames bidang di PG_PROC_INFO berisi nama semua jenis argumen (termasuk OUT dan INOUT), jika ada.

PG_STATISTIC_INDICATOR

Menyimpan informasi tentang jumlah baris yang dimasukkan atau dihapus sejak ANALISIS terakhir. Tabel PG_STATISTIC_INDICATOR diperbarui sering mengikuti operasi DML. sehingga statistik adalah perkiraan.

PG_STATISTIC_INDICATOR hanya terlihat oleh pengguna super. Untuk informasi selengkapnya, lihat Visibilitas data dalam tabel dan tampilan sistem.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
stairelid	oid	ID Tabel
stairows	float	Total jumlah baris dalam tabel.
staiin	float	Jumlah baris yang disisipkan sejak ANALYSIS terakhir.
staidel	float	Jumlah baris yang dihapus atau diperbarui sejak ANALISIS terakhir.

Contoh

Contoh berikut mengembalikan informasi untuk perubahan tabel sejak ANALYSIS terakhir.

```
select * from pg_statistic_indicator;

stairelid | stairows | staiins | staidels
----------+----------+---------+---------
   108271 |       11 |       0 |        0
   108275 |      365 |       0 |        0
   108278 |     8798 |       0 |        0
   108280 |    91865 |       0 |   100632
   108267 |    89981 |   49990 |     9999
   108269 |      808 |     606 |      374
   108282 |   152220 |   76110 |   248566
```

PG_TABLE_DEF

Menyimpan informasi tentang kolom tabel.

PG_TABLE_DEF hanya mengembalikan informasi tentang tabel yang terlihat oleh pengguna. Jika PG_TABLE_DEF tidak mengembalikan hasil yang diharapkan, verifikasi bahwa search_path parameter disetel dengan benar untuk menyertakan skema yang relevan.

Anda dapat menggunakan SVV_TABLE_INFO untuk melihat informasi yang lebih komprehensif tentang tabel, termasuk kemiringan distribusi data, kemiringan distribusi kunci, ukuran tabel, dan statistik.

Kolom tabel

Nama kolom	Jenis data	Deskripsi
nama skema	name	Nama skema.
tablename	name	Nama tabel.
kolom	name	Nama kolom.
jenis	text	Jenis data kolom.
encoding	karakter (32)	Pengkodean kolom.
distkey	boolean	Benar jika kolom ini adalah kunci distribusi untuk tabel.
sortkey	integer	Urutan kolom di tombol sortir. Jika tabel menggunakan kunci sortir majemuk, maka semua kolom yang merupakan bagian dari kunci sortir memiliki nilai positif yang menunjukkan posisi kolom dalam kunci sortir. Jika tabel menggunakan kunci sortir interleaved, maka setiap kolom yang merupakan bagian dari kunci sortir memiliki nilai yang bergantian positif atau negatif, di mana nilai absolut menunjukkan posisi kolom dalam kunci penguruta n. Jika 0, kolom bukan bagian dari kunci sortir.
notnull	boolean	Benar jika kolom memiliki kendala NOT NULL.

Contoh

Contoh berikut menunjukkan kolom kunci sortir senyawa untuk tabel LINEORDER_COMPOUND.

```
select "column", type, encoding, distkey, sortkey, "notnull"
```

```
from pg_table_def
where tablename = 'lineorder_compound'
and sortkey <> 0;

column       | type    | encoding | distkey | sortkey | notnull
-------------+---------+----------+---------+---------+--------
lo_orderkey  | integer | delta32k | false   |       1 | true
lo_custkey   | integer | none     | false   |       2 | true
lo_partkey   | integer | none     | true    |       3 | true
lo_suppkey   | integer | delta32k | false   |       4 | true
lo_orderdate | integer | delta    | false   |       5 | true
(5 rows)
```

Contoh berikut menunjukkan kolom kunci sortir yang disisipkan untuk tabel LINEORDER_INTERLEAVED.

```
select "column", type, encoding, distkey, sortkey, "notnull"
from pg_table_def
where tablename = 'lineorder_interleaved'
and sortkey <> 0;

column       | type    | encoding | distkey | sortkey | notnull
-------------+---------+----------+---------+---------+--------
lo_orderkey  | integer | delta32k | false   |      -1 | true
lo_custkey   | integer | none     | false   |       2 | true
lo_partkey   | integer | none     | true    |      -3 | true
lo_suppkey   | integer | delta32k | false   |       4 | true
lo_orderdate | integer | delta    | false   |      -5 | true
(5 rows)
```

PG_TABLE_DEF hanya akan mengembalikan informasi untuk tabel dalam skema yang disertakan dalam jalur pencarian. Untuk informasi selengkapnya, lihat search_path.

Misalnya, Anda membuat skema baru dan tabel baru, lalu kueri PG_TABLE_DEF.

```
create schema demo;
create table demo.demotable (one int);
select * from pg_table_def where tablename = 'demotable';

schemaname|tablename|column| type | encoding | distkey | sortkey | notnull
----------+---------+------+------+----------+---------+---------+--------
```

Query tidak mengembalikan baris untuk tabel baru. Periksa pengaturan untuk`search_path`.

```
show search_path;

  search_path
---------------
 $user, public
(1 row)
```

Tambahkan demo skema ke jalur pencarian dan jalankan kueri lagi.

```
set search_path to '$user', 'public', 'demo';

select * from pg_table_def where tablename = 'demotable';

schemaname| tablename |column|  type   | encoding |distkey|sortkey| notnull
----------+-----------+------+---------+----------+-------+-------+--------
demo      | demotable | one  | integer | none     | f     |     0 | f
(1 row)
```

PG_USER_INFO

PG_USER_INFO adalah tampilan sistem Amazon Redshift yang menampilkan informasi pengguna, seperti ID pengguna dan waktu kedaluwarsa kata sandi.

Hanya pengguna super yang dapat melihat PG_USER_INFO.

Kolom tabel

PG_USER_INFO berisi kolom berikut. Untuk informasi selengkapnya, lihat Dokumentasi PostgreSQL.

Nama kolom	Jenis data	Deskripsi
nama pengguna	name	Nama pengguna.
usesisid	integer	ID pengguna.
digunakan createdb	boolean	Benar jika pengguna dapat membuat database.

Nama kolom	Jenis data	Deskripsi
menggunakansuper	boolean	Benar jika pengguna adalah superuser.
menggunakancatupd	boolean	Benar jika pengguna dapat memperbarui katalog sistem.
passwd	text	Kata sandi.
valuntil	abstime	Tanggal dan waktu kedaluwarsa kata sandi.
useconfig	teks []	Default sesi untuk variabel run-time.
useconnlimit	text	Jumlah koneksi yang dapat dibuka pengguna.

Menanyakan tabel katalog

Topik

- Contoh kueri katalog

Secara umum, Anda dapat menggabungkan tabel katalog dan tampilan (relasi yang namanya dimulai dengan**PG_**) ke tabel dan tampilan Amazon Redshift.

Tabel katalog menggunakan sejumlah tipe data yang tidak didukung Amazon Redshift. Tipe data berikut didukung saat kueri menggabungkan tabel katalog ke tabel Amazon Redshift:

- bool
- “arang”
- mengapung4
- int2
- int4
- int8
- name

- oid
- text
- varchar

Jika Anda menulis kueri gabungan yang secara eksplisit atau implisit mereferensikan kolom yang memiliki tipe data yang tidak didukung, kueri akan mengembalikan kesalahan. Fungsi SQL yang digunakan dalam beberapa tabel katalog juga tidak didukung, kecuali yang digunakan oleh tabel PG_SETTINGS dan PG_LOCKS.

Misalnya, tabel PG_STATS tidak dapat ditanyakan dalam gabungan dengan tabel Amazon Redshift karena fungsi yang tidak didukung.

Tabel dan tampilan katalog berikut memberikan informasi berguna yang dapat digabungkan dengan informasi di tabel Amazon Redshift. Beberapa tabel ini hanya mengizinkan akses sebagian karena tipe data dan pembatasan fungsi. Saat Anda menanyakan tabel yang dapat diakses sebagian, pilih atau referensikan kolomnya dengan hati-hati.

Tabel berikut sepenuhnya dapat diakses dan tidak mengandung jenis atau fungsi yang tidak didukung:

- pg_atribut
- pg_cast
- pg_tergantung
- pg_description
- pg_locks
- pg_opclass

Tabel berikut dapat diakses sebagian dan berisi beberapa jenis, fungsi, dan kolom teks terpotong yang tidak didukung. Nilai dalam kolom teks terpotong menjadi nilai varchar (256).

- pg_class
- pg_kendala
- pg_database
- pg_group
- pg_language

- pg_namespace
- pg_operator
- pg_proc
- pg_settings
- pg_statistik
- pg_tables
- pg_type
- pg_user
- pg_views

Tabel katalog yang tidak tercantum di sini tidak dapat diakses atau tidak mungkin berguna bagi administrator Amazon Redshift. Namun, Anda dapat menanyakan tabel katalog apa pun atau melihat secara terbuka jika kueri Anda tidak melibatkan gabungan ke tabel Amazon Redshift.

Anda dapat menggunakan kolom OID di tabel katalog Postgres sebagai kolom gabungan. Misalnya, kondisi gabungan `pg_database.oid = stv_tbl_perm.db_id` cocok dengan ID objek database internal untuk setiap baris PG_DATABASE dengan kolom DB_ID yang terlihat di tabel STV_TBL_PERM. Kolom OID adalah kunci primer internal yang tidak terlihat ketika Anda memilih dari tabel. Tampilan katalog tidak memiliki kolom OID.

Beberapa fungsi Amazon Redshift harus berjalan hanya pada node komputasi. Jika kueri mereferensikan tabel yang dibuat pengguna, SQL berjalan pada node komputasi.

Kueri yang hanya mereferensikan tabel katalog (tabel dengan awalan PG, seperti PG_TABLE_DEF) atau yang tidak mereferensikan tabel apa pun, berjalan secara eksklusif pada node pemimpin.

Jika kueri yang menggunakan fungsi compute-node tidak mereferensikan tabel yang ditentukan pengguna atau tabel sistem Amazon Redshift mengembalikan kesalahan berikut.

```
[Amazon](500310) Invalid operation: One or more of the used functions must be applied
 on at least one user created table.
```

Contoh kueri katalog

Kueri berikut menunjukkan beberapa cara di mana Anda dapat menanyakan tabel katalog untuk mendapatkan informasi berguna tentang database Amazon Redshift.

Lihat ID tabel, database, skema, dan nama tabel

Definisi tampilan berikut menggabungkan tabel sistem STV_TBL_PERM dengan tabel katalog sistem PG_CLASS, PG_NAMESPACE, dan PG_DATABASE untuk mengembalikan ID tabel, nama database, nama skema, dan nama tabel.

```
create view tables_vw as
select distinct(stv_tbl_perm.id) table_id
,trim(pg_database.datname)   db_name
,trim(pg_namespace.nspname)   schema_name
,trim(pg_class.relname)   table_name
from stv_tbl_perm
join pg_class on pg_class.oid = stv_tbl_perm.id
join pg_namespace on pg_namespace.oid = pg_class.relnamespace
join pg_database on pg_database.oid = stv_tbl_perm.db_id;
```

Contoh berikut mengembalikan informasi untuk ID tabel 117855.

```
select * from tables_vw where table_id = 117855;
```

```
table_id | db_name   | schema_name | table_name
---------+-----------+-------------+-----------
  117855 |       dev | public      | customer
```

Buat daftar jumlah kolom per tabel Amazon Redshift

Kueri berikut bergabung dengan beberapa tabel katalog untuk mengetahui berapa banyak kolom yang berisi setiap tabel Amazon Redshift. Nama tabel Amazon Redshift disimpan di PG_TABLES dan STV_TBL_PERM; jika memungkinkan, gunakan PG_TABLES untuk mengembalikan nama tabel Amazon Redshift.

Kueri ini tidak melibatkan tabel Amazon Redshift apa pun.

```
select nspname, relname, max(attnum) as num_cols
from pg_attribute a, pg_namespace n, pg_class c
where n.oid = c.relnamespace and  a.attrelid = c.oid
and c.relname not like '%pkey'
and n.nspname not like 'pg%'
and n.nspname not like 'information%'
group by 1, 2
```

```
order by 1, 2;

nspname | relname  | num_cols
--------+----------+----------
public  | category |        4
public  | date     |        8
public  | event    |        6
public  | listing  |        8
public  | sales    |       10
public  | users    |       18
public  | venue    |        5
(7 rows)
```

Daftar skema dan tabel dalam database

Kueri berikut bergabung dengan STV_TBL_PERM ke beberapa tabel PG untuk mengembalikan daftar tabel dalam database TICKIT dan nama skema mereka (kolom NSPNAME). Query juga mengembalikan jumlah total baris di setiap tabel. (Kueri ini berguna ketika beberapa skema di sistem Anda memiliki nama tabel yang sama.)

```
select datname, nspname, relname, sum(rows) as rows
from pg_class, pg_namespace, pg_database, stv_tbl_perm
where pg_namespace.oid = relnamespace
and pg_class.oid = stv_tbl_perm.id
and pg_database.oid = stv_tbl_perm.db_id
and datname ='tickit'
group by datname, nspname, relname
order by datname, nspname, relname;

datname | nspname | relname  |  rows
--------+---------+----------+--------
tickit  | public  | category |     11
tickit  | public  | date     |    365
tickit  | public  | event    |   8798
tickit  | public  | listing  | 192497
tickit  | public  | sales    | 172456
tickit  | public  | users    |  49990
tickit  | public  | venue    |    202
(7 rows)
```

Daftar ID tabel, tipe data, nama kolom, dan nama tabel

Kueri berikut mencantumkan beberapa informasi tentang setiap tabel pengguna dan kolomnya: ID tabel, nama tabel, nama kolomnya, dan tipe data setiap kolom:

```
select distinct attrelid, rtrim(name), attname, typname
from pg_attribute a, pg_type t, stv_tbl_perm p
where t.oid=a.atttypid and a.attrelid=p.id
and a.attrelid between 100100 and 110000
and typname not in('oid','xid','tid','cid')
order by a.attrelid asc, typname, attname;

attrelid |  rtrim   |    attname     |  typname
---------+----------+----------------+-----------
  100133 | users    | likebroadway   | bool
  100133 | users    | likeclassical  | bool
  100133 | users    | likeconcerts   | bool
...
  100137 | venue    | venuestate     | bpchar
  100137 | venue    | venueid        | int2
  100137 | venue    | venueseats     | int4
  100137 | venue    | venuecity      | varchar
...
```

Hitung jumlah blok data untuk setiap kolom dalam tabel

Kueri berikut menggabungkan tabel STV_BLOCKLIST ke PG_CLASS untuk mengembalikan informasi penyimpanan untuk kolom dalam tabel PENJUALAN.

```
select col, count(*)
from stv_blocklist s, pg_class p
where s.tbl=p.oid and relname='sales'
group by col
order by col;

col | count
----+-------
  0 |     4
  1 |     4
  2 |     4
  3 |     4
  4 |     4
  5 |     4
```

```
  6 |     4
  7 |     4
  8 |     4
  9 |     8
 10 |     4
 12 |     4
 13 |     8
(13 rows)
```

Referensi konfigurasi

Anda dapat menggunakan konfigurasi untuk membantu menyesuaikan lingkungan Anda. Dengan Amazon Redshift, Anda dapat menyesuaikan dan mengoptimalkan lingkungan pergudangan data Anda dengan mengonfigurasi berbagai parameter dan pengaturan. Referensi konfigurasi menguraikan properti cluster yang tersedia, parameter database, dan opsi konfigurasi manajemen beban kerja (WLM). Anda dapat berkonsultasi dengan referensi ini untuk menyempurnakan kinerja, keamanan, dan alokasi sumber daya berdasarkan persyaratan spesifiknya. Referensi berikut memberikan panduan terperinci tentang memodifikasi konfigurasi ini untuk mencapai pengaturan pergudangan data yang Anda inginkan.

Topik

- Memodifikasi konfigurasi server
- analisis_ambang_persen
- cast_super_null_on_error
- datashare_break_glass_session_var
- datestyle
- default_array_search_null_handling
- default_geometri_encoding
- deskripsi_field_name_in_uppercase
- downcase_delimited_identifier
- enable_case_sensitive_identifier
- enable_case_sensitive_super_atribut
- mengaktifkan_pembulatan numerik
- mengaktifkan_hasil_cache_untuk_sesi
- enable_vacuum_boost
- error_pada_pembaruan_nondeterministik
- extra_float_digits
- interval_terban_composite_literal
- json_serialization_enable
- json_serialization_parse_bersarang_string
- klaster penskalaan maksimum

- maksimum_kursor_hasil_set_ukuran_hasil_
- upayan_login_gagal maksimal
- mv_enable_aqmv_untuk_sesi
- navigasi_super_null_on_error
- parse_super_null_on_error
- pg_federation_berulang_baca
- Query_group
- search_path
- Spektrum_enable_pseudo_column
- enable_spectrum_oid
- spectrum_query_maxerror
- statement_timeout
- menyimpan_proc_log_min_pesan-pesan
- timezone
- gunakan_fips_ssl
- wlm_query_slot_count

Memodifikasi konfigurasi server

Anda dapat mengubah konfigurasi server dengan cara berikut:

- Dengan menggunakan SET perintah untuk mengganti pengaturan untuk durasi sesi saat ini saja.

  Contoh:

  ```
  set extra_float_digits to 2;
  ```

- Dengan memodifikasi pengaturan grup parameter untuk cluster. Pengaturan grup parameter mencakup parameter tambahan yang dapat Anda konfigurasi. Untuk informasi selengkapnya, lihat Grup Parameter Amazon Redshift di Panduan Manajemen Amazon Redshift.
- Dengan menggunakan ALTER USER perintah untuk mengatur parameter konfigurasi ke nilai baru untuk semua sesi yang dijalankan oleh pengguna yang ditentukan.

  ```
  ALTER USER username SET parameter { TO | = } { value | DEFAULT }
  ```

Gunakan perintah SHOW untuk melihat pengaturan parameter saat ini. Gunakan SHOW ALL untuk melihat semua pengaturan yang dapat Anda konfigurasi dengan menggunakan SET perintah.

```
SHOW ALL;

name                        | setting
----------------------------+--------------
analyze_threshold_percent | 10
datestyle                   | ISO, MDY
extra_float_digits          | 2
query_group                 | default
search_path                 | $user, public
statement_timeout           | 0
timezone                    | UTC
wlm_query_slot_count        | 1
```

> **Note**
>
> Perhatikan bahwa parameter konfigurasi diterapkan ke database yang Anda sambungkan di gudang data Anda.

analisis_ambang_persen

Nilai (default dalam huruf tebal)

10, 0 hingga 100,0

Deskripsi

Menetapkan ambang batas untuk persentase baris yang diubah untuk menganalisis tabel. Untuk mengurangi waktu pemrosesan dan meningkatkan kinerja sistem secara keseluruhan, Amazon Redshift melewatkan ANALISIS untuk setiap tabel yang memiliki persentase baris yang diubah lebih rendah daripada yang ditentukan oleh. `analyze_threshold_percent` Misalnya, jika tabel berisi 100.000.000 baris dan 9.000.000 baris telah berubah sejak ANALYZE terakhir, maka secara default tabel dilewati karena kurang dari 10 persen baris telah berubah. Untuk menganalisis tabel ketika hanya sejumlah kecil baris telah berubah, atur `analyze_threshold_percent` ke jumlah kecil yang sewenang-wenang. Misalnya, jika Anda mengatur `analyze_threshold_percent` ke 0,01, maka tabel dengan 100.000.000 baris tidak akan dilewati jika setidaknya 10.000 baris

telah berubah. Untuk menganalisis semua tabel meskipun tidak ada baris yang berubah, atur `analyze_threshold_percent` ke 0.

Anda dapat memodifikasi `analyze_threshold_percent` parameter untuk sesi saat ini hanya dengan menggunakan perintah SET. Parameter tidak dapat dimodifikasi dalam grup parameter.

Contoh

```
set analyze_threshold_percent to 15;
set analyze_threshold_percent to 0.01;
set analyze_threshold_percent to 0;
```

cast_super_null_on_error

Nilai (default dalam huruf tebal)

aktif, mati

Deskripsi

Menentukan bahwa ketika Anda mencoba mengakses anggota objek atau elemen array yang tidak ada, Amazon Redshift mengembalikan nilai NULL jika kueri Anda dijalankan dalam mode lax default.

datashare_break_glass_session_var

Nilai (default dalam huruf tebal)

Tidak ada default. Nilai dapat berupa string karakter apa pun yang dihasilkan oleh Amazon Redshift ketika operasi terjadi yang tidak disarankan, seperti yang dijelaskan berikut.

Deskripsi

Menerapkan izin yang mengizinkan operasi tertentu yang umumnya tidak direkomendasikan untuk AWS Data Exchange datashare.

Secara umum, kami menyarankan agar Anda tidak menghapus atau mengubah datashare menggunakan pernyataan DROP AWS Data Exchange DATASHARE atau ALTER DATASHARE SET PUBLICACCESSIBLE. Untuk mengizinkan penghapusan AWS Data

Exchange atau perubahan data untuk mematikan pengaturan yang dapat diakses publik, atur `datashare_break_glass_session_var` variabel ke nilai satu kali. Nilai satu kali ini dihasilkan oleh Amazon Redshift dan disediakan dalam pesan kesalahan setelah upaya awal pada operasi yang dimaksud.

Setelah mengatur variabel ke nilai yang dihasilkan satu kali, jalankan pernyataan DROP DATASHARE atau ALTER DATASHARE lagi.

Untuk informasi selengkapnya, lihat Catatan penggunaan ALTER DATASHARE atau Catatan penggunaan DROP DATASHARE.

Contoh

```
set datashare_break_glass_session_var to '620c871f890c49';
```

datestyle

Nilai (default dalam huruf tebal)

Spesifikasi format (ISO, Postgres, SQL, atau Jerman), dan pemesanan year/month /hari (DMY, M DY, YMD).

- ISO - menggunakan gaya tanggal dari. YYYY-MM-DD HH:MM:SS
- Postgres — menggunakan gaya tanggal YYYY. MM-DD HH:MM:SS
- SQL — menggunakan gaya tanggal dari. MM-DD-YYYY HH:MM:SS
- Bahasa Jerman — menggunakan gaya tanggal dari. DD-MM-YYYY HH:MM:SS

Deskripsi

Menetapkan format tampilan untuk nilai tanggal dan waktu dan juga aturan untuk menafsirkan nilai input tanggal yang ambigu. String berisi dua parameter yang dapat Anda ubah secara terpisah atau bersama-sama.

Contoh

```
show datestyle;
```

```
DateStyle
-----------
ISO, MDY
(1 row)

set datestyle to 'SQL,DMY';
```

default_array_search_null_handling

Nilai (default dalam huruf tebal)

BENAR, SALAH

Deskripsi

Menentukan perilaku penanganan null untuk operasi pencarian array. Bila `default_array_search_null_handling` ya`TRUE`, nilai NULL diperlakukan sebagai elemen valid yang dapat dicari dalam array. Bila `default_array_search_null_handling` ya`FALSE`, pencarian kunci NULL mengembalikan NULL, dan jika array berisi nilai NULL tanpa ada kecocokan yang ditemukan, pencarian mengembalikan NULL.

Contoh

```
SET default_array_search_null_handling to TRUE;

-- ARRAY_CONTAINS: NULL search is allowed
SELECT ARRAY_CONTAINS(ARRAY('red', NULL, 'green'), NULL);
array_contains
----------------
 t
(1 row)

-- ARRAY_POSITION: Array can contain NULLs
SELECT ARRAY_POSITION(ARRAY('red', NULL, 'green'), 'blue');
 array_position
----------------
             -1
(1 row)
```

```
SET default_array_search_null_handling to FALSE;
```

```
-- ARRAY_CONTAINS: NULL search is disabled
SELECT ARRAY_CONTAINS(ARRAY('red', 'green'), NULL);
array_contains
----------------

(1 row)

-- ARRAY_POSITION: Array contains NULL but no match is found
SELECT ARRAY_POSITION(ARRAY('red', NULL, 'green'), 'blue');
 array_position
----------------

(1 row)
```

default_geometri_encoding

Nilai (default dalam huruf tebal)

1, 2

Deskripsi

Konfigurasi sesi yang menentukan apakah geometri spasial yang dibuat selama sesi ini dikodekan dengan kotak pembatas. Jika `default_geometry_encoding` ya1, maka geometri tidak dikodekan dengan kotak pembatas. Jika `default_geometry_encoding` ya2, maka geometri dikodekan dengan kotak pembatas. Untuk informasi selengkapnya tentang dukungan untuk kotak pembatas, lihatKotak pembatas.

deskripsi_field_name_in_uppercase

Nilai (default dalam huruf tebal)

mati (salah), aktif (benar)

Deskripsi

Menentukan apakah nama kolom yang dikembalikan oleh pernyataan SELECT adalah huruf besar atau kecil. Jika parameter ini aktif, nama kolom dikembalikan dalam huruf besar. Jika parameter ini

tidak aktif, nama kolom dikembalikan dalam huruf kecil. Amazon Redshift menyimpan nama kolom dalam huruf kecil terlepas dari pengaturan untuk. `describe_field_name_in_uppercase`

Contoh

```
set describe_field_name_in_uppercase to on;

show describe_field_name_in_uppercase;

DESCRIBE_FIELD_NAME_IN_UPPERCASE
--------------------------------
on
```

downcase_delimited_identifier

Nilai (default dalam huruf tebal)

aktif, mati

Deskripsi

Konfigurasi ini sedang dihentikan. Alih-alih gunakan`enable_case_sensitive_identifier`.

Mengaktifkan super parser untuk membaca bidang JSON yang menggunakan huruf besar atau huruf campuran. Juga memungkinkan dukungan kueri federasi untuk database PostgreSQL yang didukung dengan nama kasus campuran database, skema, tabel, dan kolom. Untuk menggunakan pengidentifikasi peka huruf besar/kecil, setel parameter ini ke nonaktif.

Catatan Penggunaan

- Jika Anda menggunakan fitur keamanan tingkat baris atau penyembunyian data dinamis, sebaiknya setel `downcase_delimited_identifier` nilai di grup parameter cluster atau grup kerja Anda. Ini memastikan bahwa `downcase_delimited_identifier` tetap konstan selama pembuatan dan melampirkan kebijakan, dan kemudian menanyakan relasi yang menerapkan kebijakan. Untuk informasi tentang keamanan tingkat baris, lihat. Row-level keamanan Untuk informasi tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.
- Saat Anda menyet `downcase_delimited_identifier` el ke nonaktif dan membuat tabel, Anda dapat mengatur nama kolom yang peka huruf besar/kecil. Saat Anda meny

`downcase_delimited_identifier` etel ke aktif dan menanyakan tabel, nama kolom dikurangi. Ini dapat menghasilkan hasil kueri yang berbeda dari saat `downcase_delimited_identifier` disetel ke nonaktif. Pertimbangkan contoh berikut:

```
SET downcase_delimited_identifier TO off;
--Amazon Redshift preserves case for column names and other identifiers.

--Create a table with two columns that are identical except for the case.
CREATE TABLE t ("c" int, "C" int);

INSERT INTO t VALUES (1, 2);

SELECT * FROM t;

 c | C
---+---
 1 | 2
(1 row)

SET enable_downcase_delimited_identifier TO on;
--Amazon Redshift no longer preserves case for column names and other identifiers.

SELECT * FROM t;

 c | c
---+---
 1 | 1
(1 row)
```

- Sebaiknya pengguna biasa yang menanyakan tabel dengan penyembunyian data dinamis atau kebijakan keamanan tingkat baris yang dilampirkan memiliki setelan downcase_delimited_identifier default. Untuk informasi selengkapnya, lihat Untuk informasi tentang keamanan tingkat baris, lihat. Row-level keamanan Untuk informasi tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.

enable_case_sensitive_identifier

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Nilai konfigurasi yang menentukan apakah pengidentifikasi nama database, skema, tabel, dan kolom peka huruf besar/kecil. Kasus pengidentifikasi nama dipertahankan ketika Anda menyertakan pengidentifikasi dalam tanda kutip ganda dan disetel `enable_case_sensitive_identifier` ke`true`. Kasus pengidentifikasi nama tidak dipertahankan, dan sebaliknya dikonversi ke huruf kecil, ketika Anda tidak menyertakan pengidentifikasi dalam tanda kutip ganda atau ketika Anda menyetel ke. `enable_case_sensitive_identifier` `false`

Kasus nama pengguna yang terlampir dalam tanda kutip ganda selalu dipertahankan terlepas dari pengaturan opsi `enable_case_sensitive_identifier` konfigurasi.

Contoh

Contoh berikut menunjukkan cara membuat dan menggunakan pengenal peka huruf besar/kecil untuk nama tabel dan kolom.

```
-- To create and use case sensitive identifiers
SET enable_case_sensitive_identifier TO true;

-- Create tables and columns with case sensitive identifiers
CREATE TABLE public."MixedCasedTable" ("MixedCasedColumn" int);

INSERT INTO public."MixedCasedTable" VALUES (1);
INSERT INTO public."MixedCasedTable" VALUES (2);
INSERT INTO public."MixedCasedTable" VALUES (3);
INSERT INTO public."MixedCasedTable" VALUES (4);
INSERT INTO public."MixedCasedTable" VALUES (5);

-- Now query with case sensitive identifiers
SELECT "MixedCasedColumn" FROM public."MixedCasedTable";

MixedCasedColumn
------------------
1
2
3
4
5

(5 rows)
```

```
SELECT * FROM public."MixedCasedTable" WHERE "MixedCasedColumn" = 1;

mixedcasedcolumn
------------------
1

(1 row)
```

Contoh berikut menunjukkan kapan kasus pengidentifikasi tidak dipertahankan.

```
-- To not use case sensitive identifiers
RESET enable_case_sensitive_identifier;

-- Mixed case identifiers are lowercased despite double quotation marks

CREATE TABLE "MixedCasedTable2" ("MixedCasedColumn" int);

CREATE TABLE MixedCasedTable2 (MixedCasedColumn int);

ERROR:  Relation "mixedcasedtable2" already exists


SELECT "MixedCasedColumn" FROM "MixedCasedTable2";

 mixedcasedcolumn
------------------
(0 rows)


SELECT MixedCasedColumn FROM MixedCasedTable2;

 mixedcasedcolumn
------------------
(0 rows)
```

Catatan Penggunaan

- Jika Anda menggunakan autorefresh untuk tampilan yang terwujud, sebaiknya setel `enable_case_sensitive_identifier` nilai di grup parameter cluster atau grup kerja Anda. Ini memastikan bahwa `enable_case_sensitive_identifier` tetap konstan saat tampilan materialisasi Anda disegarkan. Untuk informasi tentang autorefresh untuk tampilan yang terwujud,

lihat. Menyegarkan tampilan yang terwujud Untuk informasi tentang menyetel nilai konfigurasi dalam grup parameter, lihat grup parameter Amazon Redshift di Panduan Manajemen Amazon Redshift.

- Jika Anda menggunakan fitur keamanan tingkat baris atau penyembunyian data dinamis, sebaiknya setel `enable_case_sensitive_identifier` nilai di grup parameter cluster atau grup kerja Anda. Ini memastikan bahwa `enable_case_sensitive_identifier` tetap konstan selama pembuatan dan melampirkan kebijakan, dan kemudian menanyakan relasi yang menerapkan kebijakan. Untuk informasi tentang keamanan tingkat baris, lihat. Row-level keamanan Untuk informasi tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.
- Saat Anda menyet `enable_case_sensitive_identifier` el ke aktif dan membuat tabel, Anda dapat mengatur nama kolom yang peka huruf besar/kecil. Saat Anda menyet `enable_case_sensitive_identifier` el ke nonaktif dan menanyakan tabel, nama kolom dikurangi. Ini dapat menghasilkan hasil kueri yang berbeda dari saat `enable_case_sensitive_identifier` disetel ke aktif. Pertimbangkan contoh berikut:

```
SET enable_case_sensitive_identifier TO on;
--Amazon Redshift preserves case for column names and other identifiers.

--Create a table with two columns that are identical except for the case.
CREATE TABLE t ("c" int, "C" int);

INSERT INTO t VALUES (1, 2);

SELECT * FROM t;

 c | C
---+---
 1 | 2
(1 row)

SET enable_case_sensitive_identifier TO off;
--Amazon Redshift no longer preserves case for column names and other identifiers.

SELECT * FROM t;

 c | c
---+---
 1 | 1
(1 row)
```

- Sebaiknya pengguna biasa yang menanyakan tabel dengan penyembunyian data dinamis atau kebijakan keamanan tingkat baris yang dilampirkan memiliki setelan enable_case_sensitive_identifier default. Untuk informasi tentang keamanan tingkat baris, lihat. Row-level keamanan Untuk informasi tentang penyembunyian data dinamis, lihatPenyembunyian data dinamis.
- Untuk mereferensikan pengidentifikasi dengan huruf besar campuran menggunakan notasi titik, bungkus setiap pengenal peka huruf besar-kecil dalam tanda kutip ganda. Misalnya, `public."MixedCasedTable"."MixedCasedColumn"`.

enable_case_sensitive_super_atribut

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Nilai konfigurasi yang menentukan apakah menavigasi struktur tipe data SUPER dengan nama atribut yang tidak dibatasi peka huruf besar/kecil. Saat Anda menyetel `enable_case_sensitive_super_attribute` ke`true`, menavigasi struktur tipe SUPER dengan nama atribut yang tidak dibatasi akan peka huruf besar/kecil. Saat Anda menetapkan nilai ke`false`, menavigasi struktur tipe SUPER dengan nama atribut yang tidak dibatasi tidak peka huruf besar/kecil.

Saat Anda menyertakan nama atribut dalam tanda kutip ganda dan dis `enable_case_sensitive_identifier` et `true` el ke, case selalu dipertahankan, terlepas dari pengaturan opsi `enable_case_sensitive_super_attribute` konfigurasi.

`enable_case_sensitive_super_attribute`hanya berlaku untuk kolom dengan tipe data SUPER. Untuk semua kolom lainnya, pertimbangkan untuk menggunakan sebagai `enable_case_sensitive_identifier` gantinya.

Untuk informasi selengkapnya tentang menavigasi bidang JSON huruf besar dan huruf campuran, lihat. Mengakses bidang JSON dengan nama atau atribut bidang huruf besar dan huruf campuran

Contoh

Contoh berikut menunjukkan hasil memilih nilai SUPER dengan `enable_case_sensitive_super_attribute` diaktifkan dan dengan itu dinonaktifkan.

```
--Create a table with a SUPER column.
CREATE TABLE tbl (col SUPER);

--Insert values.
INSERT INTO tbl VALUES (json_parse('{
 "A": "HELLO", "a": "123"
}'));

SET enable_case_sensitive_super_attribute TO ON;

SELECT col.A FROM tbl;
  a
-----
 "HELLO"
(1 row)

SELECT col.a FROM tbl;
  a
-----
 "123"
(1 row)

SET enable_case_sensitive_super_attribute TO OFF;

SELECT col.A FROM tbl;
  a
-----
 "123"
(1 row)

SELECT col.a FROM tbl;
  a
-----
 "123"
(1 row)
```

Catatan Penggunaan

- Pandangan dan pandangan yang terwujud mengikuti nilai `enable_case_sensitive_super_attribute` pada saat pembuatannya. Late-binding tampilan, prosedur tersimpan, dan fungsi yang ditentukan pengguna mengikuti nilai `enable_case_sensitive_super_attribute` pada saat kueri.

- Jika Anda menggunakan autorefresh untuk tampilan yang terwujud, sebaiknya setel `enable_case_sensitive_identifier value` di grup parameter cluster atau grup kerja Anda. Ini memastikan bahwa `enable_case_sensitive_identifier` tetap konstan saat tampilan materialisasi Anda disegarkan. Untuk informasi tentang autorefresh untuk tampilan yang terwujud, lihat. Menyegarkan tampilan yang terwujud Untuk informasi tentang menyetel nilai konfigurasi dalam grup parameter, lihat grup parameter Amazon Redshift di Panduan Manajemen Amazon Redshift.
- Nama kolom dalam hasil pernyataan selalu dikurangi, terlepas dari nilainya. `enable_case_sensitive_super_attribute` Untuk membuat nama kolom peka huruf besar-kecil juga, aktifkan`enable_case_sensitive_identifier`.
- Sebaiknya pengguna biasa yang menanyakan tabel dengan kebijakan keamanan tingkat baris yang dilampirkan memiliki pengaturan default. `enable_case_sensitive_identifier` Untuk informasi selengkapnya, lihat Untuk informasi tentang keamanan tingkat baris, lihat. Row-level keamanan

mengaktifkan_pembulatan numerik

Nilai (default dalam huruf tebal)

aktif (benar), mati (salah)

Deskripsi

Menentukan apakah akan menggunakan pembulatan numerik. Jika `enable_numeric_rounding` yaon, Amazon Redshift membulatkan nilai NUMERIC saat mentransmisikannya ke tipe numerik lain, seperti INTEGER atau DECIMAL. Jika `enable_numeric_rounding` yaoff, Amazon Redshift memotong nilai NUMERIC saat mentransfernya ke tipe numerik lainnya. Untuk informasi selengkapnya tentang tipe numerik, lihatJenis numerik.

Contoh

```
--Create a table and insert the numeric value 1.5 into it.
CREATE TABLE t (a numeric(10, 2));

INSERT INTO t VALUES (1.5);

SET enable_numeric_rounding to ON;
--Amazon Redshift now rounds NUMERIC values when casting to other numeric types.
```

```
SELECT a::int FROM t;

 a
---
 2
(1 row)


SELECT a::decimal(10, 0) FROM t;

 a
---
 2
(1 row)


 SELECT a::decimal(10, 5) FROM t;

    a
---------
 1.50000
(1 row)


SET enable_numeric_rounding to OFF;
--Amazon Redshift now truncates NUMERIC values when casting to other numeric types.

SELECT a::int FROM t;

 a
---
 1
(1 row)


SELECT a::decimal(10, 0) FROM t;

 a
---
 1
(1 row)
```

```
SELECT a::decimal(10, 5) FROM t;

    a
---------
 1.50000
(1 row)
```

mengaktifkan_hasil_cache_untuk_sesi

Nilai (default dalam huruf tebal)

aktif (benar), mati (salah)

Deskripsi

Menentukan apakah akan menggunakan caching hasil kueri. Jika `enable_result_cache_for_session` yaon, Amazon Redshift memeriksa salinan hasil kueri yang valid dan di-cache saat kueri dikirimkan. Jika kecocokan ditemukan di cache hasil, Amazon Redshift menggunakan hasil yang di-cache dan tidak menjalankan kueri. Jika `enable_result_cache_for_session` yaoff, Amazon Redshift mengabaikan cache hasil dan menjalankan semua kueri saat dikirimkan.

Contoh

```
SET enable_result_cache_for_session TO off;
--Amazon Redshift now ignores the results cache
```

enable_vacuum_boost

Nilai (default dalam huruf tebal)

salah, benar

Deskripsi

Menentukan apakah akan mengaktifkan opsi vacuum boost untuk semua perintah VACUUM yang dijalankan dalam sesi. Jika `enable_vacuum_boost` yatrue, Amazon Redshift menjalankan semua perintah VACUUM dalam sesi dengan opsi BOOST. Jika `enable_vacuum_boost` yafalse,

Amazon Redshift tidak berjalan dengan opsi BOOST secara default. Untuk informasi selengkapnya tentang opsi BOOST, lihatVAKUM.

error_pada_pembaruan_nondeterministik

Nilai (default dalam huruf tebal)

salah, benar

Deskripsi

Menentukan apakah kueri UPDATE dengan beberapa kecocokan per baris mengembalikan kesalahan.

Contoh

```
SET error_on_nondeterministic_update TO true;

CREATE TABLE t1(x1 int, y1 int);

CREATE TABLE t2(x2 int, y2 int);

INSERT INTO t1 VALUES (1,10), (2,20), (3,30);

INSERT INTO t2 VALUES (2,40), (2,50);

UPDATE t1 SET y1=y2 FROM t2 WHERE x1=x2;

ERROR: Found multiple matches to update the same tuple.
```

extra_float_digits

Nilai (default dalam huruf tebal)

0, -15 hingga 2

Deskripsi

Menetapkan jumlah digit yang ditampilkan untuk nilai floating-point, termasuk float4 dan float8. Nilai ditambahkan ke jumlah digit standar (FLT_DIG atau DBL_DIG sesuai). Nilai dapat diatur setinggi

2, untuk memasukkan digit yang sebagian signifikan. Ini sangat berguna untuk mengeluarkan data float yang harus dipulihkan dengan tepat. Atau dapat diatur negatif untuk menekan angka yang tidak diinginkan.

Contoh

Contoh berikut ditetapkan `extra_float_digits` ke-2. Pertama, tunjukkan pengaturan parameter saat ini.

```
show all;
      name                      | setting
--------------------------+----------------
analyze_threshold_percent | 10
datestyle                 | ISO, MDY
extra_float_digits        | 2
query_group               | default
search_path               | $user, public
statement_timeout         | 0
timezone                  | UTC
wlm_query_slot_count      | 1
```

Kemudian, atur nilai baru ke-2.

```
set extra_float_digits to -2;
```

Akhirnya tunjukkan pengaturan parameter yang diperbarui.

```
show all;
      name                      | setting
--------------------------+----------------
analyze_threshold_percent | 10
datestyle                 | ISO, MDY
extra_float_digits        | -2
query_group               | default
search_path               | $user, public
statement_timeout         | 0
timezone                  | UTC
wlm_query_slot_count      | 1
```

interval_terban_composite_literal

Nilai (default dalam huruf tebal)

salah, benar

Deskripsi

Konfigurasi sesi yang memodifikasi nilai interval yang berisi bagian YEAR TO MONTH dan DAY TO SECOND.

Jika `interval_forbid_composite_literals` ya`true`, kesalahan dikembalikan jika interval dengan bagian YEAR TO MONTH dan DAY TO SECOND ditemui. Misalnya, SQL berikut berisi INTERVAL DAY TO SECOND dengan bagian YEAR TO MONTH dan DAY TO SECOND.

```
SELECT INTERVAL '1 year 1 day' DAY TO SECOND;
ERROR:  Interval Day To Second literal cannot contain year-month parts. Disable the GUC
 interval_forbid_composite_literals to suppress this error and silently discard the
 year-month part.
```

Jika `interval_forbid_composite_literals` ya`false`, Amazon Redshift menekan kesalahan dan memotong bagian YEAR TO MONTH dari nilai INTERVAL DAY TO SECOND. Misalnya, SQL berikut berisi INTERVAL DAY TO SECOND dengan bagian YEAR TO MONTH dan DAY TO SECOND.

```
SET interval_forbid_composite_literals to "false";
SELECT INTERVAL '1 year 1 day' DAY TO SECOND;


 intervald2s
------------------------------
1 days 0 hours 0 mins 0.0 secs
```

json_serialization_enable

Nilai (default dalam huruf tebal)

salah, benar

Deskripsi

Konfigurasi sesi yang memodifikasi perilaku serialisasi JSON dari data berformat ORC, JSON, Ion, dan Parquet. Jika `json_serialization_enable` ya`true`, semua koleksi tingkat atas secara otomatis diserialkan ke JSON dan dikembalikan sebagai VARCHAR (65535). Kolom nonkompleks tidak terpengaruh atau diserialisasi. Karena kolom koleksi diserialkan sebagai VARCHAR (65535), subbidang bersarangnya tidak dapat lagi diakses secara langsung sebagai bagian dari sintaks kueri (yaitu, dalam klausa filter). Jika `json_serialization_enable` ya`false`, koleksi tingkat atas tidak diserialkan ke JSON. Untuk informasi selengkapnya tentang serialisasi JSON bersarang, lihat. Serialisasi JSON bersarang kompleks

json_serialization_parse_bersarang_string

Nilai (default dalam huruf tebal)

salah, benar

Deskripsi

Konfigurasi sesi yang memodifikasi perilaku serialisasi JSON dari data berformat ORC, JSON, Ion, dan Parquet. Ketika `json_serialization_enable` keduanya `json_serialization_parse_nested_strings` dan benar, nilai string yang disimpan dalam tipe kompleks (seperti, peta, struktur, atau array) diurai dan ditulis sebaris langsung ke hasil jika mereka adalah JSON yang valid. Jika `json_serialization_parse_nested_strings` salah, string dalam tipe kompleks bersarang diserialisasi sebagai string JSON yang lolos. Untuk informasi selengkapnya, lihat Serialisasi tipe kompleks yang berisi string JSON.

klaster penskalaan maksimum

Nilai (default dalam huruf tebal)

1, 0 hingga 10

Deskripsi

Menetapkan jumlah maksimum cluster penskalaan konkurensi yang diizinkan saat penskalaan konkurensi diaktifkan. Tingkatkan nilai ini jika diperlukan lebih banyak penskalaan konkurensi.

Kurangi nilai ini untuk mengurangi penggunaan cluster penskalaan konkurensi dan biaya penagihan yang dihasilkan.

Jumlah maksimum cluster penskalaan konkurensi adalah kuota yang dapat disesuaikan. Untuk informasi selengkapnya, lihat kuota Amazon Redshift di Panduan Manajemen Amazon Redshift.

maksimum_kursor_hasil_set_ukuran_hasil_

Nilai (default dalam huruf tebal)

0 (default maksimum) - 14400000 MB

Deskripsi

`max_cursor_result_set_size`Parameter tidak lagi digunakan. Untuk informasi selengkapnya tentang ukuran set hasil kursor, lihatKendala kursor.

upayan_login_gagal maksimal

Nilai (default dalam huruf tebal)

5, 2 hingga 50

Deskripsi

Menetapkan jumlah upaya login kata sandi yang gagal berturut-turut yang diizinkan oleh Amazon Redshift sebelum mengunci pengguna database. Ketika jumlah upaya gagal berturut-turut mencapai nilai ini, Amazon Redshift mengunci pengguna. Pengguna yang terkunci tetap terkunci sampai pengguna super atau pengguna dengan hak istimewa ALTER USER membukanya. Login yang berhasil mengatur ulang penghitung. Parameter ini hanya berlaku untuk otentikasi berbasis kata sandi. Pengguna federasi tidak terpengaruh.

Parameter ini bersifat cluster-wide. Untuk mengaturnya, Anda harus menjadi superuser dan menggunakan perintah ALTER SYSTEM SET. Anda tidak dapat mengubahnya dengan perintah SET. Perubahan ini berlaku tanpa restart cluster. Rentang yang valid adalah 2 hingga 50; nilai di luar rentang ini mengembalikan kesalahan.

Contoh

```
ALTER SYSTEM SET max_failed_login_attempts = 10;
```

mv_enable_aqmv_untuk_sesi

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Menentukan apakah Amazon Redshift dapat melakukan penulisan ulang kueri otomatis dari tampilan yang terwujud di tingkat sesi.

navigasi_super_null_on_error

Nilai (default dalam huruf tebal)

meny ala, mati

Deskripsi

Menentukan bahwa ketika Anda mencoba menavigasi anggota objek atau elemen array yang tidak ada, Amazon Redshift mengembalikan nilai NULL jika kueri Anda dijalankan dalam mode lax default.

parse_super_null_on_error

Nilai (default dalam huruf tebal)

mati, aktif

Deskripsi

Menentukan bahwa ketika Amazon Redshift mencoba mengurai anggota objek atau elemen array yang tidak ada, Amazon Redshift mengembalikan nilai NULL jika kueri Anda dijalankan dalam mode ketat.

pg_federation_berulang_baca

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Menentukan tingkat isolasi transaksi kueri federasi untuk hasil dari database PostgreSQL.

- Jika pg_federation_repeatable_read benar, transaksi federasi diproses dengan semantik tingkat isolasi REPEATABLE READ. Ini adalah opsi default.
- Bila pg_federation_repeatable_read salah, transaksi federasi diproses dengan semantik tingkat isolasi READ COMMITTED.

Untuk informasi selengkapnya, lihat berikut ini:

- Pertimbangan saat mengakses data federasi dengan Amazon Redshift.
- Mengelola operasi penulisan bersamaan.

Contoh

Perintah berikut pg_federation_repeatable_read ditetapkan on untuk sesi. Perintah show menunjukkan nilai dari nilai yang ditetapkan.

```
set pg_federation_repeatable_read to on;
```

```
show pg_federation_repeatable_read;

pg_federation_repeatable_read
-----------------------------
on
```

Query_group

Nilai (default dalam huruf tebal)

Tidak ada default; nilainya bisa berupa string karakter apa pun.

Deskripsi

Menerapkan label yang ditentukan pengguna ke sekelompok kueri yang dijalankan selama sesi yang sama. Label ini ditangkap dalam log kueri. Anda dapat menggunakannya untuk membatasi hasil dari tabel STL_QUERY dan STV_INFLIGHT dan tampilan SVL_QLOG. Misalnya, Anda dapat menerapkan label terpisah ke setiap kueri yang Anda jalankan untuk mengidentifikasi kueri secara unik tanpa harus mencari ID mereka.

Parameter ini tidak ada di file konfigurasi server dan harus disetel saat runtime dengan perintah SET. Meskipun Anda dapat menggunakan string karakter panjang sebagai label, label dipotong menjadi 30 karakter di kolom LABEL tabel STL_QUERY dan tampilan SVL_QLOG (dan hingga 15 karakter di STV_INFLIGHT).

Dalam contoh berikut, query_group disetel ke**`Monday`**, kemudian beberapa kueri dijalankan dengan label itu.

```
set query_group to 'Monday';
SET
select * from category limit 1;
...
...
select query, pid, substring, elapsed, label
from svl_qlog where label ='Monday'
order by query;

query | pid  |        substring                   |  elapsed  | label
------+------+------------------------------------+-----------+--------
789   | 6084 | select * from category limit 1;    | 65468     | Monday
790   | 6084 | select query, trim(label) from ... | 1260327   | Monday
791   | 6084 | select * from svl_qlog where ..    | 2293547   | Monday
792   | 6084 | select count(*) from bigsales;     | 108235617 | Monday
...
```

search_path

Nilai (default dalam huruf tebal)

'$pengguna', publik, nama_skema

Daftar nama skema yang ada yang dipisahkan koma. Jika '$user' ada, maka skema yang memiliki nama yang sama seperti SESSION_USER diganti, jika tidak maka akan diabaikan.

Deskripsi

Menentukan urutan di mana skema dicari ketika objek (seperti tabel atau fungsi) direferensikan oleh nama sederhana tanpa komponen skema:

- Jalur pencarian tidak didukung dengan skema eksternal dan tabel eksternal. Tabel eksternal harus secara eksplisit memenuhi syarat oleh skema eksternal. Datashares berperilaku sebagai data eksternal dan oleh karena itu tidak didukung sebagai jalur pencarian.
- Ketika objek dibuat tanpa skema target tertentu, mereka ditempatkan di skema pertama yang terdaftar di jalur pencarian. Jika jalur pencarian kosong, sistem mengembalikan kesalahan.
- Ketika objek dengan nama yang identik ada dalam skema yang berbeda, yang ditemukan pertama kali di jalur pencarian digunakan.
- Objek yang tidak ada dalam skema mana pun di jalur pencarian hanya dapat direferensikan dengan menentukan skema yang berisi dengan nama yang memenuhi syarat (bertitik).
- Skema katalog sistem, pg_catalog, selalu dicari. Jika disebutkan di jalur, itu dicari dalam urutan yang ditentukan. Jika tidak, itu dicari sebelum salah satu item jalur.
- Skema tabel sementara sesi saat ini, pg_temp_nnn, selalu dicari jika ada. Itu dapat secara eksplisit dicantumkan di jalur dengan menggunakan alias pg_temp. Jika tidak terdaftar di jalur, itu dicari terlebih dahulu (bahkan sebelum pg_catalog). Namun, skema sementara hanya dicari untuk nama relasi (tabel, tampilan). Itu tidak dicari untuk nama fungsi.

Contoh

Contoh berikut membuat skema ENTERPRISE dan menetapkan search_path ke skema baru.

```
create schema enterprise;
set search_path to enterprise;
```

```
show search_path;

 search_path
-------------
 enterprise
(1 row)
```

Contoh berikut menambahkan skema ENTERPRISE ke search_path default.

```
set search_path to '$user', public, enterprise;
show search_path;

         search_path
-----------------------------
 "$user", public, enterprise
(1 row)
```

Contoh berikut menambahkan tabel FRONTIER ke skema ENTERPRISE.

```
create table enterprise.frontier (c1 int);
```

Ketika tabel PUBLIC.FRONTIER dibuat dalam database yang sama, dan pengguna tidak menentukan nama skema dalam kueri, lebih ENTERPRISE.FRONTIER di PUBLIC.FRONTIER utamakan.

```
create table public.frontier(c1 int);
insert into enterprise.frontier values(1);
select * from frontier;

frontier
----
(0 rows)

select * from enterprise.frontier;

c1
----
1
(1 row)
```

Spektrum_enable_pseudo_column

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Anda dapat menonaktifkan pembuatan pseudocolumn untuk sesi dengan mengatur parameter `spectrum_enable_pseudo_columns` konfigurasi ke`false`.

Contoh

Perintah berikut menonaktifkan pembuatan pseudocolumn untuk sesi.

```
set spectrum_enable_pseudo_columns to false;
```

enable_spectrum_oid

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Anda juga dapat menonaktifkan hanya `$spectrum_oid` pseudocolumn dengan mengatur parameter `enable_spectrum_oid` konfigurasi ke`false`.

Contoh

Perintah berikut menonaktifkan `$spectrum_oid` pseudocolumn dengan mengatur parameter `enable_spectrum_oid` konfigurasi ke. `false`

```
set enable_spectrum_oid to false;
```

spectrum_query_maxerror

Nilai (default dalam huruf tebal)

-1, bilangan bulat

Deskripsi

Anda dapat menentukan bilangan bulat untuk menunjukkan jumlah kesalahan maksimum yang harus diterima sebelum membatalkan kueri. Nilai negatif mematikan penanganan data kesalahan maksimum. Hasilnya masukSVL_SPECTRUM_SCAN_ERROR.

Contoh

Contoh berikut mengasumsikan data ORC yang berisi kelebihan karakter dan karakter yang tidak valid. Definisi kolom untuk `my_string` menentukan panjang 3 karakter. Berikut ini adalah contoh data untuk contoh ini:

```
my_string
---------------
abcdef
gh
ab
```

Perintah berikut mengatur jumlah kesalahan maksimum ke 1 dan melakukan kueri.

```
set spectrum_query_maxerror to 1;
SELECT my_string FROM orc_data;
```

Kueri berhenti dan hasilnya masukSVL_SPECTRUM_SCAN_ERROR.

statement_timeout

Nilai (default dalam huruf tebal)

0 (mematikan batasan), x milidetik

Deskripsi

Menghentikan pernyataan apa pun yang mengambil alih jumlah milidetik yang ditentukan.

`statement_timeout`Nilai adalah jumlah waktu maksimum kueri dapat dijalankan sebelum Amazon Redshift menghentikannya. Waktu ini termasuk perencanaan, antrian dalam manajemen beban kerja (WLM), dan waktu eksekusi. Bandingkan waktu ini dengan batas waktu WLM (max_execution_time) dan QMR (query_execution_time), yang hanya mencakup waktu eksekusi.

Jika batas waktu WLM (max_execution_time) juga ditentukan sebagai bagian dari konfigurasi WLM, yang lebih rendah dari statement_timeout dan max_execution_time digunakan. Untuk informasi selengkapnya, lihat Batas waktu WLM.

Contoh

Karena kueri berikut membutuhkan waktu lebih dari 1 milidetik, waktu habis dan dibatalkan.

```
set statement_timeout = 1;

select * from listing where listid>5000;
ERROR:  Query (150) canceled on user's request
```

menyimpan_proc_log_min_pesan-pesan

Nilai (default dalam huruf tebal)

LOG, INFO, PEMBERITAHUAN, PERINGATAN, PENGECUALIAN

Deskripsi

Menentukan tingkat pencatatan minimum pesan prosedur tersimpan yang dinaikkan. Pesan pada atau di atas level yang ditentukan dicatat. Defaultnya adalah LOG (semua pesan dicatat). Urutan level log dari tertinggi ke terendah adalah sebagai berikut:

1. PENGECUALIAN
2. WARNING
3. PEMBERITAHUAN
4. INFO
5. LOG

Misalnya jika Anda menentukan nilai PEMBERITAHUAN, maka pesan hanya dicatat untuk PEMBERITAHUAN, PERINGATAN, dan PENGECUALIAN.

timezone

Nilai (default dalam huruf tebal)

UTC, zona waktu

Sintaksis

```
SET timezone { TO | = } [ time_zone | DEFAULT ]

SET time zone [ time_zone | DEFAULT ]
```

Deskripsi

Menetapkan zona waktu untuk sesi saat ini. Zona waktu dapat berupa offset dari Universal Coordinated Time (UTC) atau nama zona waktu.

> **Note**
>
> Anda tidak dapat mengatur parameter `timezone` konfigurasi dengan menggunakan grup parameter cluster. Zona waktu dapat diatur hanya untuk sesi saat ini dengan menggunakan perintah SET. Untuk mengatur zona waktu untuk semua sesi yang dijalankan oleh pengguna database tertentu, gunakan ALTER USER perintah. UBAH PENGGUNA... SET TIMEZONE mengubah zona waktu untuk sesi berikutnya, bukan untuk sesi saat ini.

Saat Anda mengatur zona waktu menggunakan perintah `SET timezone` (satu kata) dengan salah satu TO atau=, Anda dapat menentukan time_zone sebagai nama zona waktu, offset POSIX-style format, atau offset ISO-8601 format, seperti yang ditunjukkan berikut.

```
SET timezone { TO | = } time_zone
```

Ketika Anda mengatur zona waktu menggunakan perintah SET zona waktu tanpa TO atau=, Anda dapat menentukan time_zone menggunakan INTERVAL dan juga nama zona waktu, offset POSIX-style format, atau offset ISO-8601 format, seperti yang ditunjukkan berikut.

```
SET time zone time_zone
```

Format zona waktu

Amazon Redshift mendukung format zona waktu berikut:

- Nama zona waktu
- INTERVAL
- POSIX-style spesifikasi zona waktu
- ISO-8601 mengimbangi

Karena singkatan zona waktu, seperti PST atau PDT, didefinisikan sebagai offset tetap dari UTC dan tidak menyertakan aturan waktu musim panas, perintah SET tidak mendukung singkatan zona waktu.

Untuk detail selengkapnya tentang format zona waktu, lihat berikut ini.

Nama zona waktu — Nama zona waktu penuh, seperti America/New _York. Nama zona waktu penuh dapat mencakup aturan penghematan siang hari. Perhatikan bahwa Amazon Redshift menggunakan Database Zona Waktu IANA sebagai sumber otoritatif spesifikasi zona waktu.

Berikut ini adalah contoh nama zona waktu:

- Etc/Greenwich
- America/New_York
- CST6CDT
- GB

> **Note**
>
> Banyak nama zona waktu, seperti EST, MST, NZ, dan UCT, juga merupakan singkatan.

Untuk melihat daftar nama zona waktu yang valid, jalankan perintah berikut.

```
select pg_timezone_names();
```

INTERVAL — Offset dari UTC. Misalnya, PST adalah — 8:00 atau —8 jam.

Berikut ini adalah contoh offset zona waktu INTERVAL:

- -- 8:00
- —8 jam
- 30 menit

POSIX-style format — Spesifikasi zona waktu dalam bentuk STDoffset atau STD offsetDST, di mana STD adalah singkatan zona waktu, offset adalah offset numerik dalam jam barat dari UTC, dan DST adalah singkatan zona penghematan cahaya siang opsional. Waktu penghematan musim panas diasumsikan satu jam lebih cepat dari offset yang diberikan.

POSIX-style format zona waktu menggunakan offset positif di sebelah barat Greenwich, berbeda dengan ISO-8601 konvensi, yang menggunakan offset positif di sebelah timur Greenwich.

Berikut ini adalah contoh zona POSIX-style waktu:

- PST8
- PST8PDT
- EST 5
- EST5EDT

> **Note**
>
> Amazon Redshift tidak memvalidasi spesifikasi zona POSIX-style waktu, jadi dimungkinkan untuk mengatur zona waktu ke nilai yang tidak valid. Misalnya, perintah berikut tidak mengembalikan kesalahan, meskipun mengatur zona waktu ke nilai yang tidak valid.
>
> ```
> set timezone to 'xxx36';
> ```

ISO-8601 Offset — Offset dari UTC dalam formulir`±[hh]:[mm]`.

Berikut ini adalah contoh ISO-8601 offset:

- - 8:00
- + 7:30

Contoh

Contoh berikut menetapkan zona waktu untuk sesi saat ini ke New York.

```
set timezone = 'America/New_York';
```

Contoh berikut menetapkan zona waktu untuk sesi saat ini ke UTC-8 (PST).

```
set timezone to '-8:00';
```

Contoh berikut menggunakan INTERVAL untuk mengatur zona waktu ke PST.

```
set timezone interval '-8 hours'
```

Contoh berikut mengatur ulang zona waktu untuk sesi saat ini ke zona waktu default sistem (UTC).

```
set timezone to default;
```

Untuk mengatur zona waktu bagi pengguna database, gunakan pernyataan ALTER USER... SET. Contoh berikut menetapkan zona waktu untuk dbuser ke New York. Nilai baru tetap ada untuk pengguna untuk semua sesi berikutnya.

```
ALTER USER dbuser SET timezone to 'America/New_York';
```

gunakan_fips_ssl

Nilai (default dalam huruf tebal)

benar, salah

Deskripsi

Nilai grup parameter yang menentukan apakah mode FIPS-compliant SSL digunakan. Jika `use_fips_ssl` ya`true`, maka mode FIPS-compliant SSL digunakan. Jika `use_fips_ssl` ya`false`, maka mode FIPS-compliant SSL tidak digunakan. Untuk informasi selengkapnya, lihat Mengonfigurasi opsi keamanan untuk koneksi di Panduan Manajemen Amazon Redshift.

Untuk mengonfigurasi parameter untuk cluster yang disediakan Amazon Redshift, lihat Tentang grup parameter di Panduan Manajemen Amazon Redshift. Untuk mengonfigurasi parameter untuk Redshift Serverless, lihat Mengon figurasi koneksi FIPS-compliant SSL ke Amazon Redshift Serverless di Panduan Manajemen Amazon Redshift, dan CreateWorkgroup atau UpdateWorkgroup di Referensi API Tanpa Server Redshift.

wlm_query_slot_count

Nilai (default dalam huruf tebal)

1, 1 hingga 50 (tidak dapat melebihi jumlah slot yang tersedia (tingkat konkurensi) untuk kelas layanan)

Deskripsi

Menetapkan jumlah slot kueri yang digunakan kueri.

Manajemen beban kerja (WLM) mencadangkan slot di kelas layanan sesuai dengan tingkat konkurensi yang ditetapkan untuk antrian. Misalnya, jika tingkat konkurensi diatur ke 5, maka kelas layanan memiliki 5 slot. WLM mengalokasikan memori yang tersedia untuk kelas layanan secara merata untuk setiap slot. Untuk informasi selengkapnya, lihat Manajemen beban kerja.

> **Note**
>
> Jika nilai wlm_query_slot_count lebih besar dari jumlah slot yang tersedia (tingkat konkurensi) untuk kelas layanan, kueri gagal. Jika Anda mengalami kesalahan, kurangi wlm_query_slot_count ke nilai yang diijinkan.

Untuk operasi di mana kinerja sangat dipengaruhi oleh jumlah memori yang dialokasikan, seperti menyedot debu, meningkatkan nilai wlm_query_slot_count dapat meningkatkan kinerja. Secara khusus, untuk perintah vakum lambat, periksa catatan yang sesuai dalam tampilan SVV_VACUUM_SUMMARY. Jika Anda melihat nilai tinggi (mendekati atau lebih tinggi dari 100) untuk sort_dan merge_increments dalam tampilan SVV_VACUUM_SUMMARY, pertimbangkan untuk meningkatkan nilai untuk wlm_query_slot_count saat berikutnya Anda menjalankan Vacuum terhadap tabel tersebut.

Meningkatkan nilai wlm_query_slot_count membatasi jumlah kueri bersamaan yang dapat dijalankan. Misalnya, misalkan kelas layanan memiliki tingkat konkurensi 5 dan wlm_query_slot_count disetel

ke 3. Saat kueri berjalan dalam sesi dengan wlm_query_slot_count disetel ke 3, maksimal 2 kueri bersamaan lagi dapat dijalankan dalam kelas layanan yang sama. Kueri berikutnya menunggu dalam antrian sampai kueri yang sedang berjalan selesai dan slot dibebaskan.

Contoh

Gunakan perintah SET untuk mengatur nilai wlm_query_slot_count untuk durasi sesi saat ini.

```
set wlm_query_slot_count to 3;
```

Riwayat dokumen

> **Note**
>
> Untuk deskripsi fitur baru di Amazon Redshift, lihat Apa yang baru.

Tabel berikut menjelaskan perubahan dokumentasi penting pada Panduan Pengembang Database Amazon Redshift setelah Mei 2018. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Versi API: 2012-12-01

Untuk daftar perubahan pada Panduan Manajemen Amazon Redshift, lihat Riwayat Dokumen Panduan Manajemen Amazon Redshift.

Untuk informasi selengkapnya tentang fitur baru, termasuk daftar perbaikan dan nomor versi cluster terkait untuk setiap rilis, lihat Riwayat Versi Cluster.

Perubahan	Deskripsi	Tanggal
Dukungan untuk integrasi Amazon DynamoDB Zero-ETL dengan Amazon Redshift	Anda sekarang dapat membuat integrasi Zero-ETL antara DynamoDB dan Amazon Redshift untuk mengaktifkan analitik hampir real-time pada data DynamoDB tanpa proses ETL tradisional.	Desember 1, 2024
Dukungan untuk integrasi Amazon Aurora Zero-ETL dengan pemfilteran data Amazon Redshift	Anda sekarang dapat menerapkan filter data ke integrasi Amazon Aurora Zero-ETL untuk mereplikasi data tertentu secara selektif ke Amazon Redshift.	1 Juni 2024

Dukungan untuk Multi-AZ penerapan untuk Amazon Redshift	Amazon Redshift sekarang mendukung penerapan Multi-AZ untuk klaster RA3, memberikan ketersediaan tinggi dengan menerapkan sumber daya komputasi di dua Zona Ketersediaan untuk terus beroperasi selama kejadian tak terduga di AZ.	Maret 1, 2024
Dukungan untuk integrasi Amazon Aurora Zero-ETL dengan Amazon Redshift	Anda sekarang dapat membuat integrasi Zero-ETL antara Aurora dan Amazon Redshift untuk mengaktif kan analitik dan pembelaja ran mesin yang hampir real-time menggunakan Amazon Redshift pada petabyte data transaksional dari Amazon Aurora.	1 November 2023
Dukungan untuk integrasi Amazon RDS Zero-ETL dengan Amazon Redshift	Anda sekarang dapat membuat integrasi Zero-ETL antara Amazon RDS dan Amazon Redshift untuk membuat data database RDS tersedia di Amazon Redshift hampir secara real-time untuk beban kerja analitik, ML, dan AI.	1 Juni 2023

Dukungan untuk konsumsi streaming lintas wilayah	Pengambilan streaming Amazon Redshift sekarang mendukung penyerapan streaming lintas wilayah di mana sumber Amazon Kinesis Data Streams atau topik Amazon MSK Anda dapat ditemukan di cluster Amazon Redshift yang Wilayah AWS berbeda dari cluster Amazon Redshift Anda.	1 November 2022
Dukungan untuk konsumsi streaming dari sumber Apache Kafka	Sekarang Anda dapat menyerap data streaming dari cluster Amazon MSK Provisioned, Amazon MSK Serverless, dan open-source Apache Kafka langsung ke Amazon Redshift menggunakan tampilan yang terwujud.	1 Juli 2022
Dukungan untuk Amazon Redshift Serverless	Amazon Redshift Serverless memudahkan menjalankan dan menskalakan analitik tanpa harus menyediakan dan mengelola infrastruktur gudang data. Menambahkan dokumentasi komprehensif untuk kelompok kerja tanpa server, ruang nama, pemantauan penggunaan, dan tampilan sistem khusus tanpa server.	Juni 29, 2022

Dukungan untuk penyerapan streaming dari Amazon Kinesis Data Streams	Sekarang Anda dapat menyerap data streaming dari Amazon Kinesis Data Streams langsung ke Amazon Redshift menggunakan tampilan yang terwujud, sehingga menghilangkan kebutuhan untuk melakukan penyusunan data di Amazon S3 sebelum diserap ke Amazon Redshift.	1 April 2022
Dukungan untuk data spasial dengan tipe data GEOMETRY dan GEOGRAPHY	Amazon Redshift sekarang mendukung data spasial dengan tipe data GEOMETRY dan GEOGRAPHY, fungsi SQL spasial, dan kemampuan memuat shapefile untuk analisis geospasial dan kueri berbasis lokasi.	1 Maret 2022
Dukungan untuk AQUA (Advanced Query Accelerator)	AQUA adalah cache terdistribusi dan dipercepat perangkat keras yang memungkinkan Amazon Redshift berjalan hingga sepuluh kali lebih cepat daripada gudang data cloud lainnya dengan secara otomatis meningkatkan jenis kueri tertentu dengan perangkat keras yang dirancang khusus.	Januari 1, 2022

Kemampuan Amazon Redshift ML yang ditingkatkan	Menambahkan dokumentasi komprehensif untuk Amazon Redshift ML termasuk pernyataan CREATE MODEL, dukungan untuk XGBoost, K-means pengelompokan, model klasifikasi multi-kelas, fitur penjelasan model, dan integrasi dengan untuk ML otomatis. Amazon SageMaker AI	1 Desember 2021
Dukungan yang ditingkatkan untuk data semistruktur dengan tipe data SUPER	Kemampuan yang ditingkatkan untuk tipe data SUPER untuk menyimpan dan menanyakan data semistruktur seperti JSON, Avro, dan Ion dengan peningkatan kinerja, pengetikan dinamis, dan dukungan untuk memuat data semistruktur hingga ukuran 16 MB.	November 1, 2021
Dukungan untuk kemampuan berbagi data yang ditingkatkan	Fungsionalitas berbagi data yang ditingkatkan dengan dukungan untuk berbagi data lintas cluster dan lintas akun, integrasi dengan Amazon Redshift Serverless, dan berbagi data standar antara berbagai jenis cluster.	1 Oktober 2021
Dukungan untuk geometri 3D dan 4D spasial dan fungsi spasial baru	Anda sekarang dapat menggunakan fungsi spasial tambahan dan dukungan geometri 3D dan 4D ditambahkan ke beberapa fungsi.	19 Agustus 2021

Dukungan untuk pengkodean kompresi kolom untuk pengoptimalan tabel otomatis	Anda dapat menentukan opsi ENCODE AUTO untuk tabel untuk secara otomatis mengelola pengkodean kompresi untuk semua kolom dalam tabel.	Agustus 3, 2021
Dukungan untuk beberapa pernyataan SQL atau pernyataan SQL dengan parameter menggunakan Amazon Redshift Data API	Anda sekarang dapat menjalankan beberapa pernyataan SQL atau pernyataan dengan parameter dengan Amazon Redshift Data API.	28 Juli 2021
Dukungan untuk kolasi yang tidak peka huruf besar/kecil dengan penggantian tingkat kolom	Anda sekarang dapat menggunakan klausa COLLATE dalam pernyataan CREATE DATABASE untuk menentukan kolasi default.	24 Juni 2021
Dukungan untuk berbagi data di seluruh akun	Anda sekarang dapat berbagi data di seluruh Akun AWS.	30 April 2021
Dukungan untuk kueri data hierarkis dengan CTE rekursif	Anda sekarang dapat menggunakan ekspresi tabel umum rekursif (CTE) di SQL Anda.	29 April 2021
Dukungan untuk kueri lintas database	Anda sekarang dapat menanyakan data di seluruh database dalam cluster.	10 Maret 2021

Dukungan untuk kontrol akses halus pada perintah COPY dan UNLOAD	Anda sekarang dapat memberikan hak istimewa untuk menjalankan perintah COPY dan UNLOAD kepada pengguna dan grup tertentu di cluster Amazon Redshift Anda untuk membuat kebijakan kontrol akses yang lebih halus.	12 Januari 2021
Dukungan untuk JSON asli dan data semi-terstruktur	Anda sekarang dapat menentukan tipe data SUPER.	9 Desember 2020
Dukungan untuk kueri federasi ke MySQL	Anda sekarang dapat menulis query federasi ke mesin MySQL yang didukung.	9 Desember 2020
Dukungan untuk berbagi data	Anda sekarang dapat berbagi data di seluruh cluster Amazon Redshift.	9 Desember 2020
Dukungan untuk pengoptimalan tabel otomatis	Anda sekarang dapat menentukan distribusi otomatis dan mengurutkan kunci.	9 Desember 2020
Dukungan untuk Amazon Redshift ML	Anda sekarang dapat membuat, melatih, dan menerapkan model pembelajaran mesin (ML).	8 Desember 2020
Dukungan untuk penyegaran otomatis dan penulisan ulang kueri dari tampilan yang terwujud	Anda sekarang dapat memperbarui tampilan yang terwujud dengan penyegaran otomatis dan kinerja kueri dapat ditingkatkan dengan penulisan ulang otomatis.	11 November 2020

Dukungan untuk tipe data TIME dan TIMETZ	Anda sekarang dapat membuat tabel dengan tipe data TIME dan TIMETZ. Tipe data TIME menyimpan waktu tanpa informasi zona waktu, dan TIMETZ menyimpan waktu hari termasuk informasi zona waktu	11 November 2020
Dukungan untuk UDF Lambda dan tokenisasi	Anda sekarang dapat menulis UDF Lambda untuk mengaktif kan tokenisasi data eksternal.	26 Oktober 2020
Dukungan untuk mengubah pengkodean kolom tabel	Anda sekarang dapat mengubah pengkodean kolom tabel.	20 Oktober 2020
Dukungan untuk query di seluruh database	Amazon Redshift sekarang dapat menanyakan seluruh database dalam cluster.	15 Oktober 2020
Dukungan untuk HyperLogLog Sketsa	Amazon Redshift sekarang dapat menyimpan dan memproses HyperLogL ogSketches.	2 Oktober 2020
Dukungan untuk Apache Hudi dan Danau Delta	Peningkatan untuk membuat tabel eksternal untuk Redshift Spectrum.	24 September 2020
Dukungan untuk penyempur naan untuk menanyakan data spasial	Peningkatan termasuk memuat shapefile dan beberapa fungsi SQL spasial baru.	15 September 2020

Tampilan terwujud mendukung tabel eksternal	Anda dapat membuat tampilan terwujud di Amazon Redshift yang mereferensikan sumber data eksternal.	19 Juni 2020
Dukungan untuk menulis ke tabel eksternal	Anda dapat menulis ke tabel eksternal dengan menjalankan CREATE EXTERNAL TABLE AS SELECT untuk menulis ke tabel eksternal baru atau INSERT INTO untuk menyisipkan data ke tabel eksternal yang ada.	8 Juni 2020
Dukungan untuk kontrol penyimpanan untuk skema	Pembaruan perintah dan tampilan yang mengelola kontrol penyimpanan untuk skema.	2 Juni 2020
Dukungan untuk ketersediaan umum kueri federasi	Informasi yang diperbarui tentang kueri data dengan kueri federasi.	16 April 2020
Dukungan untuk fungsi spasial tambahan	Menambahkan deskripsi fungsi spasial tambahan.	2 April 2020
Dukungan untuk ketersediaan umum tampilan yang terwujud	Tampilan terwujud umumnya tersedia dimulai dengan versi cluster 1.0.13059.	19 Februari 2020
Dukungan untuk hak istimewa tingkat kolom	Column-level hak istimewa tersedia mulai dengan versi cluster 1.0.13059.	19 Februari 2020

MENGGANTI TABEL	Anda dapat menggunakan perintah ALTER TABLE dengan klausa ALTER DISTSTYLE ALL untuk mengubah gaya distribusi tabel.	11 Februari 2020
Dukungan untuk kueri federasi	Memperbarui panduan untuk menjelaskan kueri federasi dengan CREATE EXTERNAL SCHEMA yang diperbarui.	3 Desember 2019
Dukungan untuk ekspor danau data	Memperbarui panduan untuk menjelaskan parameter baru dari perintah UNLOAD.	3 Desember 2019
Dukungan untuk data spasial	Memperbarui panduan untuk menjelaskan dukungan untuk data spasial.	21 November 2019
Dukungan untuk konsol baru	Memperbarui panduan untuk menggambarkan konsol Amazon Redshift baru.	11 November 2019
Dukungan untuk pengurutan tabel otomatis	Amazon Redshift dapat secara otomatis mengurutkan data tabel.	7 November 2019
Dukungan untuk opsi VACUUM BOOST	Anda dapat menggunakan opsi BOOST saat menyedot debu meja.	7 November 2019
Dukungan untuk kolom IDENTITAS default	Anda dapat membuat tabel dengan kolom IDENTITAS default.	19 September 2019

Dukungan untuk pengkodean kompresi AZ64	Anda dapat menyandikan beberapa kolom dengan pengkodean kompresi AZ64.	19 September 2019
Dukungan untuk prioritas kueri	Anda dapat mengatur prioritas kueri dari antrian WLM otomatis.	22 Agustus 2019
Dukungan untuk AWS Lake Formation	Anda dapat menggunakan Katalog Data Formasi Danau dengan Amazon Redshift Spectrum.	8 Agustus 2019
PRESET COMPUPDATE	Anda dapat menggunak an perintah COPY dengan COMPUPDATE PRESET untuk mengaktifkan Amazon Redshift untuk memilih pengkodean kompresi.	13 Juni 2019
UBAH KOLOM	Anda dapat menggunak an perintah ALTER TABLE dengan ALTER COLUMN untuk meningkatkan ukuran kolom VARCHAR.	22 Mei 2019
Dukungan untuk prosedur tersimpan	Anda dapat menentukan prosedur PL/pgSQL tersimpan di Amazon Redshift.	24 April 2019
Dukungan untuk konfigura si manajemen beban kerja otomatis (WLM)	Anda dapat mengaktifkan Amazon Redshift untuk berjalan dengan WLM otomatis.	24 April 2019

UNLOAD ke Zstandard	Anda dapat menggunakan perintah UNLOAD untuk menerapkan kompresi Zstandard ke teks dan file nilai yang dipisahkan koma (CSV) yang dibongkar ke Amazon S3.	3 April 2019
Penskalaan konkurensi	Saat penskalaan konkurens i diaktifkan, Amazon Redshift secara otomatis menambahk an kapasitas cluster tambahan saat Anda membutuhkannya untuk memproses peningkatan kueri baca bersamaan.	21 Maret 2019
UNLOAD ke CSV	Anda dapat menggunakan perintah UNLOAD untuk membongkar ke file yang diformat sebagai teks CSV.	13 Maret 2019
Gaya distribusi AUTO	Untuk mengaktifkan distribus i otomatis, Anda dapat menentukan gaya distribus i AUTO dengan pernyataa n CREATE TABLE. Saat Anda mengaktifkan distribus i otomatis, Amazon Redshift menetapkan gaya distribus i optimal berdasarkan data tabel. Perubahan distribusi terjadi di latar belakang, dalam beberapa detik.	23 Januari 2019

COPY dari Parquet mendukung SMALLINT	COPY sekarang mendukung pemuatan dari file berformat Parquet ke kolom yang menggunakan tipe data SMALLINT. Untuk informasi selengkapnya, lihat SALIN dari Format Data Columnar	2 Januari 2019
JATUHKAN DATABASE EKSTERNAL	Anda dapat menghapus database eksternal dengan menyertakan klausa DROP EXTERNAL DATABASE dengan perintah DROP SCHEMA.	Selasa, 03 Desember 2018
Cross-region MEMBONGKAR	Anda dapat MEMUNLOAD ke bucket Amazon S3 di Wil AWS ayah lain dengan menentukan parameter REGION.	31 Oktober 2018
Hapus vakum otomatis	Amazon Redshift secara otomatis menjalankan operasi VACUUM DELETE di latar belakang, jadi Anda jarang, jika pernah, perlu menjalank an vacuum DELETE ONLY. Amazon Redshift menjadwal kan VACUUM DELETE untuk berjalan selama periode beban berkurang dan menjeda operasi selama periode beban tinggi.	31 Oktober 2018

Distribusi otomatis	Jika Anda tidak menentukan gaya distribusi dengan pernyataan CREATE TABLE, Amazon Redshift menetapkan gaya distribusi optimal berdasarkan data tabel. Perubahan distribusi terjadi di latar belakang, dalam beberapa detik.	31 Oktober 2018
Kontrol akses berbutir halus untuk AWS Glue Data Catalog	Anda sekarang dapat menentukan tingkat akses ke data yang disimpan di AWS Glue Data Catalog.	15 Oktober 2018
UNLOAD dengan tipe data	Anda dapat menentukan opsi MANIFEST VERBOSE dengan perintah UNLOAD untuk menambahkan metadata ke file manifes, termasuk nama dan tipe data kolom, ukuran file, dan jumlah baris.	10 Oktober 2018
Tambahkan beberapa partisi menggunakan pernyataan ALTER TABLE tunggal	Untuk tabel eksternal Redshift Spectrum, Anda dapat menggabungkan beberapa klausa PARTITION dalam satu pernyataan ALTER TABLE ADD. Untuk informasi selengkapnya, lihat Mengubah Contoh Tabel Eksternal.	10 Oktober 2018

UNLOAD dengan header	Anda dapat menentukan opsi HEADER dengan perintah UNLOAD untuk menambahkan baris header yang berisi nama kolom di bagian atas setiap file output.	19 September 2018
Tabel dan tampilan sistem baru	Dokumentasi SVL_S3retries, SV L_USER_INFO, dan STL_DISK_FULL_DIAG ditambahkan. https://docs.aws.amazon.com/redshift/latest/dg/r_STL_DISK_FULL_DIAG.html	31 Agustus 2018
Dukungan untuk data bersarang di Amazon Redshift Spectrum	Anda sekarang dapat menanyakan data bersarang yang disimpan di tabel Amazon Redshift Spectrum. Untuk informasi selengkapnya, lihat Tutorial: Mengkueri Data Bersarang dengan Amazon Redshift Spectrum.	8 Agustus 2018
SQA aktif secara default	Akselerasi kueri pendek (SQA) sekarang diaktifkan secara default untuk semua cluster baru. SQA menggunakan pembelajaran mesin untuk memberikan kinerja yang lebih tinggi, hasil yang lebih cepat, dan prediktabilitas waktu eksekusi kueri yang lebih baik. Untuk informasi selengkapnya, lihat Akselerasi Kueri Pendek.	8 Agustus 2018

Penasihat Amazon Redshift	Sekarang Anda bisa mendapatkan rekomenda si khusus tentang cara meningkatkan kinerja cluster dan mengurangi biaya operasi dari Amazon Redshift Advisor. Untuk informasi selengkap nya, lihat Amazon Redshift Advisor.	26 Juli 2018
Referensi alias langsung	Anda sekarang dapat merujuk ke ekspresi alias segera setelah Anda mendefini sikannya. Untuk informasi selengkapnya, lihat PILIH Daftar.	18 Juli 2018
Tentukan jenis kompresi saat membuat tabel eksternal	Anda sekarang dapat menentukan jenis kompresi saat membuat tabel eksternal dengan Amazon Redshift Spectrum. Untuk informasi selengkapnya, lihat Membuat Tabel Eksternal.	27 Juni 2018
PG_LAST_UNLOAD_ID	Dokumentasi ditambahkan untuk fungsi Informasi Sistem baru: PG_LAST_UNLOAD_ID. Untuk informasi selengkapnya, lihat PG_LAST_UNLOAD_ID.	27 Juni 2018
UBAH NAMA KOLOM TABEL	ALTER TABLE sekarang mendukung penggantian nama kolom untuk tabel eksternal. Untuk informasi selengkapnya, lihat Meng ubah Contoh Tabel Eksternal.	7 Juni 2018

Pembaruan lebih awal

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Pengembang Database Amazon Redshift sebelum Juni 2018.

Ubah	Deskripsi	Tanggal diubah
COPY dari Parquet termasuk SMALLINT	COPY sekarang mendukung pemuatan dari file berformat Parquet ke kolom yang menggunakan tipe data SMALLINT. Untuk informasi selengkapnya, lihat SALIN dari format data kolumnar	2 Januari 2019
SALIN dari format kolumnar	COPY sekarang mendukung pemuatan dari file di Amazon S3 yang menggunakan format data kolumnar Parquet dan ORC. Untuk informasi selengkapnya, lihat SALIN dari format data kolumnar	17 Mei 2018
Waktu berjalan maksimum dinamis untuk SQA	Secara default, manajemen beban kerja (WLM) sekarang secara dinamis menetapkan nilai untuk waktu berjalan maksimum akselerasi kueri pendek (SQA) berdasarkan analisis beban kerja cluster Anda. Untuk informasi selengkapnya, lihat Waktu proses maksimum untuk kueri pendek.	17 Mei 2018
Kolom baru di STL_LOAD_COMMITS	Tabel sistem STL_LOAD_COMMITS memiliki kolom baru,. `file_format`	10 Mei 2018
Kolom baru di STL_HASHJOIN dan tabel log sistem lainnya	Tabel sistem STL_HASHJOIN memiliki tiga kolom baru,,, `hash_segment` dan. `hash_step` `checksum` Juga, `checksum` a ditambahkan ke STL_MERGEJOIN, STL_NESTLOOP, STL_HASH, STL_SCAN, STL_SORT, STL_LIMIT, dan STL_PROJECT.	17 Mei 2018
Kolom baru di STL_AGGR	Tabel sistem STL_AGGR memiliki dua kolom baru, dan. `resizes` `flushable`	19 April 2018

Ubah	Deskripsi	Tanggal diubah
Opsi baru untuk fungsi REGEX	Untuk fungsi REGEXP_INSTR dan REGEXP_SUBSTR, Anda sekarang dapat menentukan kemunculan kecocokan yang akan digunakan dan apakah akan melakukan kecocokan peka huruf besar-kecil. REGEXP_INSTR juga memungkinkan Anda menentukan apakah akan mengembalikan posisi karakter pertama pertandingan atau posisi karakter pertama setelah akhir pertandingan.	22 Maret 2018
Kolom baru dalam tabel sistem	Kolom tombstonedblocks, tossedblocks, dan batched_by ditambahkan ke tabel sistem. STL_COMM_STATS Kolom localslice ditambahkan ke tampilan STV_SLICE sistem.	22 Maret 2018
Tambahkan dan jatuhkan kolom di tabel eksternal	ALTER TABLEsekarang mendukung ADD COLUMN dan DROP COLUMN untuk tabel eksternal Amazon Redshift Spectrum.	22 Maret 2018
Redshift Spectrum Wilayah baru AWS	Redshift Spectrum sekarang tersedia di Wilayah Mumbai dan São Paulo. Untuk mengetahui daftar Wilayah yang didukung, lihat Wilayah Spektrum Pergeseran Merah Amazon.	22 Maret 2018
Batas tabel meningkat menjadi 20.000	Jumlah tabel maksimum sekarang 20.000 untuk tipe node cluster 8xlarge. Batas untuk tipe node besar dan xlarge adalah 9.900. Untuk informasi selengkapnya, lihat Batas dan kuota.	13 Maret 2018
Dukungan Redshift Spectrum untuk JSON dan Ion	Menggunakan Redshift Spectrum, Anda dapat mereferensikan file dengan data skalar dalam format data JSON atau Ion. Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.	26 Februari 2018

Ubah	Deskripsi	Tanggal diubah
Rantaian peran IAM untuk Redshift Spectrum	Anda dapat mengrantai AWS Identity and Access Management (IAM) peran sehingga cluster Anda dapat mengambil peran lain yang tidak dilampirkan ke cluster, termasuk peran milik AWS akun lain. Untuk informasi selengkapnya, lihat Mengrantai peran IAM di Amazon Redshift Spectrum.	Februari 1, 2018
TAMBAHKAN PARTISI mendukung JIKA TIDAK ADA	Klausa ADD PARTITION untuk ALTER TABLE sekarang mendukung opsi IF NOT EXISTS. Untuk informasi selengkapnya, lihat ALTER TABLE.	11 Januari 2018
Data TANGGAL untuk tabel eksternal	Tabel eksternal Redshift Spectrum sekarang mendukung tipe data DATE. Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.	11 Januari 2018
Redshift Spectrum Wilayah baru AWS	Redshift Spectrum sekarang tersedia di Singapura, Sydney, Seoul, dan Wilayah Frankfurt. Untuk daftar Wilayah yang didukung AWS , lihatWilayah Spektrum Pergeseran Merah Amazon.	16 November 2017
Akselerasi kueri singkat dalam manajemen beban kerja Amazon Redshift (WLM)	Akselerasi kueri pendek (SQA) memprioritaskan kueri berjalan pendek yang dipilih di depan kueri yang berjalan lebih lama. SQA mengeksekusi kueri yang berjalan singkat di ruang khusus, sehingga kueri SQA tidak dipaksa menunggu dalam antrian di belakang kueri yang lebih lama. Dengan SQA, kueri yang berjalan pendek mulai dieksekusi lebih cepat dan pengguna melihat hasilnya lebih cepat. Untuk informasi selengkapnya, lihat Akselerasi kueri singkat.	16 November 2017

Ubah	Deskripsi	Tanggal diubah
WLM menugaskan kembali kueri yang dilompat	Alih-alih membatalkan dan memulai ulang kueri yang dilompat, manajemen beban kerja Amazon Redshift (WLM) sekarang menetapkan kembali kueri yang memenuhi syarat ke antrian baru. Ketika WLM menetapkan kembali kueri, ia memindahkan kueri ke antrian baru dan melanjutkan eksekusi, yang menghemat waktu dan sumber daya sistem. Kueri yang dilompat yang tidak memenuhi syarat untuk ditugaskan kembali akan dimulai ulang atau dibatalkan. Untuk informasi selengkapnya, lihat Melompat antrian kueri WLM.	16 November 2017
Akses log sistem untuk pengguna	Di sebagian besar tabel log sistem yang terlihat oleh pengguna, baris yang dihasilkan oleh pengguna lain tidak terlihat oleh pengguna biasa secara default. Untuk mengizinkan pengguna biasa melihat semua baris dalam tabel yang terlihat pengguna, termasuk baris yang dihasilkan oleh pengguna lain, jalankan ALTER USER atau BUAT PENGGUNA dan atur parameter SYSLOG ACCESS ke UNRESTRICTED.	16 November 2017
Caching hasil	DenganCaching hasil, saat Anda menjalankan kueri, Amazon Redshift menyimpan hasilnya. Saat Anda menjalankan kueri lagi, Amazon Redshift memeriksa salinan hasil kueri yang valid dan di-cache. Jika kecocokan ditemukan di cache hasil, Amazon Redshift menggunakan hasil yang di-cache dan tidak menjalankan kueri. Caching hasil diaktifkan secara default. Untuk mematikan cache hasil, atur parameter mengaktifkan_hasil_cache_untuk_sesi konfigurasi ke`off`.	16 November 2017
Fungsi metadata kolom	PG_GET_COLSdan PG_GET_LATE_BINDING_VIEW_COLS mengembalikan metadata kolom untuk tabel Amazon Redshift, tampilan, dan tampilan yang mengikat terlambat.	16 November 2017

Ubah	Deskripsi	Tanggal diubah
Melompat antrian WLM untuk CTAS	Manajemen beban kerja Amazon Redshift (WLM) kini mendukung pernyataan query queue hopping for CREATE TABLE AS (CTAS) serta kueri read-only, seperti pernyataan SELECT. Untuk informasi selengkapnya, lihat Melompat antrian kueri WLM.	19 Oktober 2017
File manifes Amazon Redshift Spectrum	Saat membuat tabel eksternal Redshift Spectrum, Anda dapat menentukan file manifes yang mencantum kan lokasi file data di Amazon S3. Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.	19 Oktober 2017
Amazon Redshift Spectrum Wilayah baru AWS	Redshift Spectrum sekarang tersedia di Wilayah UE (Irlandia) dan Asia Pasifik (Tokyo). Untuk daftar Wilayah yang didukung AWS , lihatBatasan Amazon Redshift Spectrum.	19 Oktober 2017
Amazon Redshift Spectrum menambahkan format file	Anda sekarang dapat membuat tabel eksternal Redshift Spectrum berdasarkan format file data Regex, OpenCSV, dan Avro. Untuk informasi selengkapnya, lihat CREATE EXTERNAL TABLE.	5 Oktober 2017
Pseudocolumn untuk tabel eksternal Amazon Redshift Spectrum	Anda dapat memilih `$path` dan `$size` pseudocol umn dalam tabel eksternal Redshift Spectrum untuk melihat lokasi dan ukuran file data yang direferensikan di Amazon S3. Untuk informasi selengkapnya, lihat Pseudokolom.	5 Oktober 2017
Fungsi untuk memvalidasi JSON	Anda dapat menggunakan IS_VALID_JSON_ARRA Y fungsi IS_VALID_JSON dan untuk memeriksa pemformatan JSON yang valid. Fungsi JSON lainnya sekarang memiliki `null_if_invalid` argumen opsional.	5 Oktober 2017

Ubah	Deskripsi	Tanggal diubah
LISTAGG BERBEDA	Anda dapat menggunakan klausa DISTINCT dengan fungsi LIST agregat dan fungsi LIST jendela untuk menghilangkan nilai duplikat dari ekspresi yang ditentukan sebelum menggabungkan.	5 Oktober 2017
Lihat nama kolom dalam huruf besar	Untuk melihat nama kolom dalam hasil SELECT dalam huruf besar, Anda dapat mengatur parameter deskripsi_field_name_in_uppercase konfigurasi `ketrue`.	5 Oktober 2017
Lewati baris header di tabel eksternal	Anda dapat mengatur `skip.header.line.count` properti dalam CREATE EXTERNAL TABLE perintah untuk melewati baris header di awal file data Redshift Spectrum.	5 Oktober 2017
Pindai jumlah baris	Aturan monitor kueri WLM menggunakan metrik scan_row_count untuk mengembalikan jumlah baris dalam langkah pemindaian. Jumlah baris adalah jumlah total baris yang dipancarkan sebelum memfilter baris yang ditandai untuk dihapus (baris hantu) dan sebelum menerapkan filter kueri yang ditentukan pengguna. Untuk informasi selengkapnya, lihat Metrik pemantauan kueri untuk Amazon Redshift disediakan.	21 September 2017
Fungsi yang ditentukan pengguna SQL	Fungsi yang ditentukan pengguna SQL skalar (UDF) menggabungkan klausa SQL SELECT yang dieksekusi ketika fungsi dipanggil dan mengembalikan nilai tunggal. Untuk informasi selengkapnya, lihat UDF SQL skalar.	31 Agustus 2017

Ubah	Deskripsi	Tanggal diubah
Late-binding pandangan	Tampilan pengikatan terlambat tidak terikat pada objek database yang mendasarinya, seperti tabel dan fungsi yang ditentukan pengguna. Akibatnya, tidak ada ketergantungan antara tampilan dan objek yang dirujukkannya. Anda dapat membuat tampilan meskipun objek yang direferensikan tidak ada. Karena tidak ada ketergantungan, Anda dapat melepaska n atau mengubah objek yang direferensikan tanpa mempengaruhi tampilan. Amazon Redshift tidak memeriksa dependensi sampai tampilan ditanyaka n. Untuk membuat tampilan pengikatan terlambat, tentukan klausa WITH NO SCHEMA BINDING dengan pernyataan CREATE VIEW Anda. Untuk informasi selengkapnya, lihat CREATE VIEW.	31 Agustus 2017
Fungsi OCTET_LENGTH	OCTET_LENGTHmengembalikan panjang string yang ditentukan sebagai jumlah byte.	18 Agustus 2017
Jenis file ORC dan Grok didukung	Amazon Redshift Spectrum sekarang mendukung format data ORC dan Grok untuk file data Redshift Spectrum. Untuk informasi selengkapnya, lihat File data untuk kueri di Amazon Redshift Spectrum.	18 Agustus 2017
RegexSerD e sekarang didukung	Amazon Redshift Spectrum sekarang mendukung format RegexSerDe data. Untuk informasi selengkap nya, lihat File data untuk kueri di Amazon Redshift Spectrum.	19 Juli 2017
Kolom baru ditambahkan ke SVV_TABLES dan SVV_COLUMNS	Kolom `domain_name` dan `remarks` ditambahk an keSVV_KOLOM. Kolom komentar ditambahkan keSVV_TABEL.	19 Juli 2017
Tampilan sistem SVV_TABLES dan SVV_COLUMNS	T SVV_TABEL ampilan dan SVV_KOLOM sistem memberikan informasi tentang kolom dan detail lainnya untuk tabel dan tampilan lokal dan eksternal.	7 Juli 2017

Ubah	Deskripsi	Tanggal diubah
VPC tidak lagi diperlukan untuk Amazon Redshift Spectrum dengan Amazon EMR Hive metastore	Redshift Spectrum menghapus persyaratan bahwa cluster Amazon Redshift dan cluster Amazon EMR harus berada di VPC yang sama dan subnet yang sama saat menggunakan metastore Amazon EMR Hive. Untuk informasi selengkapnya, lihat Bekerja dengan katalog eksternal di Amazon Redshift Spectrum.	7 Juli 2017
UNLOAD ke ukuran file yang lebih kecil	Secara default, UNLOAD membuat beberapa file di Amazon S3 dengan ukuran maksimum 6,2 GB. Untuk membuat file yang lebih kecil, tentukan MAXFILESIZE dengan perintah UNLOAD. Anda dapat menentukan ukuran file maksimum antara 5 MB dan 6,2 GB. Untuk informasi selengkapnya, lihat MEMBONGKAR.	7 Juli 2017
PROPERTI TABEL	Anda sekarang dapat mengatur parameter TABLE PROPERTIES num CREATE EXTERNAL TABLE Ro ALTER TABLE ws untuk atau memperbarui statistik tabel untuk mencerminkan jumlah baris dalam tabel.	6 Juni 2017
MENGANALISIS KOLOM PREDIKAT	Untuk menghemat waktu dan sumber daya cluster, Anda dapat memilih untuk menganalisis hanya kolom yang kemungkinan akan digunakan sebagai predikat. Ketika Anda menjalankan ANALYZE dengan klausa PREDICATE COLUMNS, operasi analisis hanya mencakup kolom yang telah digunakan dalam gabungan, kondisi filter, atau grup berdasarkan klausa, atau digunakan sebagai kunci pengurutan atau kunci distribusi. Untuk informasi selengkapnya, lihat Menganalisis tabel.	25 Mei 2017

Ubah	Deskripsi	Tanggal diubah
Kebijakan IAM untuk Amazon Redshift Spectrum	Untuk memberikan akses ke bucket Amazon S3 hanya menggunakan Redshift Spectrum, Anda dapat menyertakan kondisi yang memungkinkan akses untuk agen `AWS Redshift/Spectrum` pengguna. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk Amazon Redshift Spectrum.	25 Mei 2017
Pemindaian Rekursif Spektrum Amazon Redshift	Redshift Spectrum sekarang memindai file di subfolder serta folder yang ditentukan di Amazon S3. Untuk informasi selengkapnya, lihat Tabel eksternal untuk Redshift Spectrum.	25 Mei 2017
Aturan pemantauan kueri	Menggunakan aturan pemantauan kueri WLM, Anda dapat menentukan batas kinerja berbasis metrik untuk antrian WLM dan menentukan tindakan apa yang harus diambil ketika kueri melampaui batas tersebut —log, hop, atau batalkan. Anda menentukan aturan pemantauan kueri sebagai bagian dari konfigurasi manajemen beban kerja (WLM) Anda. Untuk informasi selengkapnya, lihat Aturan pemantauan kueri WLM.	April 21, 2017
Spektrum Pergeseran Merah Amazon	Menggunakan Redshift Spectrum, Anda dapat menanyakan dan mengambil data secara efisien dari file di Amazon S3 tanpa harus memuat data ke dalam tabel. Kueri Redshift Spectrum dijalankan dengan sangat cepat terhadap kumpulan data besar karena Redshift Spectrum memindai file data langsung di Amazon S3. Sebagian besar pemrosesan terjadi di lapisan Amazon Redshift Spectrum, dan sebagian besar data tetap ada di Amazon S3. Beberapa cluster dapat secara bersamaan menanyakan kumpulan data yang sama di Amazon S3 tanpa perlu membuat salinan data untuk setiap cluster. Untuk informasi selengkapnya, lihat Spektrum Pergeseran Merah Amazon	19 April 2017

Ubah	Deskripsi	Tanggal diubah
Tabel sistem baru untuk mendukung Redshift Spectrum	Tampilan sistem baru berikut telah ditambahkan untuk mendukung Redshift Spectrum: • SVL_S3QUERY • SVL_S3QUERY_SUMMARY • SVV_KOLON_EKSTERNAL_ • SVV_EXTERNAL_DATABASE • SVV_PARTISI_EKSTERNAL • SVV_EXTERNAL_TABEL • PG_EXTERNAL_SCHEMA	19 April 2017
APPROXIMATE PERCENTILE_DISC fungsi agregat	Fungsi PERKIRAAN PERCENTILE_DISC agregat sekarang tersedia.	4 April 2017
Server-side enkripsi dengan KMS	Anda sekarang dapat membongkar data ke Amazon S3 menggunakan enkripsi sisi server dengan AWS Key Management Service kunci (). SSE-KMS Selain itu, MENYALIN sekarang secara transparan memuat file KMS-encrypted data dari Amazon S3. Untuk informasi selengkapnya, lihat MEMBONGKAR.	9 Februari 2017

Ubah	Deskripsi	Tanggal diubah
Sintaks otorisasi baru	Anda sekarang dapat menggunakan parameter IAM_ROLE, MASTER_SYMMETRIC_KEY, ACCESS_KEY_ID, SECRET_ACCESS_KEY, dan SESSION_TOKEN untuk memberikan otorisasi dan akses informasi untuk perintah COPY, UNLOAD, dan CREATE LIBRARY. Sintaks otorisasi baru menyediakan alternatif yang lebih fleksibel untuk menyediakan argumen string tunggal ke parameter CREDENTIALS. Untuk informasi selengkapnya, lihat Parameter otorisasi.	9 Februari 2017
Peningkatan batas skema	Anda sekarang dapat membuat hingga 9.900 skema per cluster. Untuk informasi selengkapnya, lihat BUAT SKEMA.	9 Februari 2017
Pengkodean tabel default	CREATE TABLEdan ALTER TABLE sekarang tetapkan pengkodean kompresi LZO ke sebagian besar kolom baru. Kolom yang didefinisikan sebagai kunci pengurutan, kolom yang didefinisikan sebagai tipe data BOOLEAN, REAL, atau DOUBLE PRECISION, dan tabel sementara ditetapkan pengkodean RAW secara default. Untuk informasi selengkapnya, lihat ENCODE.	6 Februari 2017
Pengkodean kompresi ZSTD	Amazon Redshift sekarang mendukung pengkodean kompresi kolom ZSTD.	Januari 19, 2017
Fungsi agregat PERCENTILE_CONT dan MEDIAN	PERSENTILE_CONTdan sekarang MEDIAN tersedia sebagai fungsi agregat serta fungsi jendela.	Januari 19, 2017

Ubah	Deskripsi	Tanggal diubah
User-defined fungsi (UDF) Pengguna Logging	Anda dapat menggunakan modul logging Python untuk membuat pesan kesalahan dan peringatan yang ditentukan pengguna di UDF Anda. Setelah eksekusi kueri, Anda dapat menanyakan SVL_UDF_LOG tampilan sistem untuk mengambil pesan yang dicatat. Untuk informasi selengkapnya tentang pesan yang ditentukan pengguna, lihat Kesalahan pencatatan dan peringatan di Python UDF	8 Desember 2016
ANALISIS KOMPRESI perkiraan pengurangan	Perintah ANALYZE COMPRESSION sekarang melaporkan perkiraan persentase pengurangan ruang disk untuk setiap kolom. Untuk informasi selengkapnya, lihat MENGANALISIS KOMPRESI.	10 November 2016
Batas koneksi	Anda sekarang dapat menetapkan batas pada jumlah koneksi database yang diizinkan untuk dibuka pengguna secara bersamaan. Anda juga dapat membatasi jumlah koneksi bersamaan untuk database. Untuk informasi selengkapnya, lihat BUAT PENGGUNA dan BUAT BASIS DATA.	10 November 2016
Peningkatan urutan penyortiran COPY	COPY sekarang secara otomatis menambahkan baris baru ke wilayah tabel yang diurutkan saat Anda memuat data Anda dalam urutan kunci penguruta n. Untuk persyaratan khusus untuk mengaktifkan peningkatan ini, lihat Muat data Anda dalam urutan kunci	10 November 2016
CTAS dengan kompresi	CREATE TABLE AS (CTAS) sekarang secara otomatis menetapkan pengkodean kompresi ke tabel baru berdasarkan tipe data kolom. Untuk informasi selengkapnya, lihat Warisan atribut kolom dan tabel.	Oktober 28, 2016

Ubah	Deskripsi	Tanggal diubah
Cap waktu dengan tipe data zona waktu	Amazon Redshift sekarang mendukung timestamp dengan tipe data zona waktu (TIMESTAMPTZ). Juga, beberapa fungsi baru telah ditambahkan untuk mendukung tipe data baru. Untuk informasi selengkapnya, lihat Fungsi tanggal dan waktu.	29 September 2016
Analisis ambang	Untuk mengurangi waktu pemrosesan dan meningkatkan kinerja sistem secara keseluruhan untuk MENGANALISA operasi, Amazon Redshift melewatkan analisis tabel jika persentase baris yang telah berubah sejak perintah ANALYZE terakhir dijalankan lebih rendah dari ambang analisis yang ditentukan oleh parameter. analisis_ambang_persen Secara default, `analyze_threshold_percent` adalah 10.	9 Agustus 2016
Tabel sistem STL_RESTARTED_SESSIONS baru	Saat Amazon Redshift memulai ulang sesi, catat STL_RESTARTED_SESSIONS ID proses baru (PID) dan PID lama.	9 Agustus 2016
Memperbarui dokumentasi Fungsi Tanggal dan Waktu	Menambahkan ringkasan fungsi dengan tautan keFungsi tanggal dan waktu, dan memperbarui referensi fungsi untuk konsistensi.	24 Juni 2016
Kolom baru di STL_CONNECTION_LOG	Tabel STL_CONNECTION_LOG sistem memiliki dua kolom baru untuk melacak koneksi SSL. Jika Anda secara rutin memuat log audit ke tabel Amazon Redshift, Anda perlu menambahkan kolom baru berikut ke tabel target: sslcompression dan sslexpansion.	5 Mei 2016

Ubah	Deskripsi	Tanggal diubah
MD5-hash kata sandi	Anda dapat menentukan kata sandi untuk ALTER USER perintah BUAT PENGGUNA atau dengan memberikan MD5-hash string kata sandi dan nama pengguna.	21 April 2016
Kolom baru di STV_TBL_PERM	`backup`Kolom dalam STV_TBL_PERM tampilan sistem menunjukkan apakah tabel disertakan dalam snapshot cluster. Untuk informasi selengkapnya, lihat BACKUP.	21 April 2016
No-backup tabel	Untuk tabel, seperti tabel pementasan, yang tidak akan berisi data penting, Anda dapat menentuka n BACKUP NO di CREATE TABLE AS pernyataa n CREATE TABLE atau untuk mencegah Amazon Redshift menyertakan tabel dalam snapshot otomatis atau manual. Menggunakan tabel tanpa cadangan menghemat waktu pemrosesan saat membuat snapshot dan memulihkan dari snapshot serta mengurangi ruang penyimpanan di Amazon S3.	7 April 2016
Ambang penghapusan VAKUM	Secara default, VAKUM perintah sekarang mengklaim kembali ruang sedemikian rupa sehingga setidaknya 95 persen dari baris yang tersisa tidak ditandai untuk dihapus. Akibatnya, VACUUM biasanya membutuhk an waktu yang jauh lebih sedikit untuk fase hapus dibandingkan dengan mengklaim kembali 100 persen baris yang dihapus. Anda dapat mengubah ambang default untuk satu tabel dengan menyertakan parameter TO PERCENT saat menjalankan perintah VACUUM.	7 April 2016
Tabel sistem SVV_TRANS ACTIONS	T SVV_TRANSAKSI ampilan sistem mencatat informasi tentang transaksi yang saat ini memegang kunci pada tabel dalam database.	7 April 2016

Ubah	Deskripsi	Tanggal diubah
Menggunakan peran IAM untuk mengakses sumber daya lain AWS	Untuk memindahkan data antara cluster Anda dan AWS sumber daya lain, seperti Amazon S3, DynamoDB, Amazon EMR, atau Amazon EC2, cluster Anda harus memiliki izin untuk mengakses sumber daya dan melakukan tindakan yang diperlukan. Sebagai alternatif yang lebih aman untuk menyediakan pasangan kunci akses dengan perintah COPY, UNLOAD, atau CREATE LIBRARY, sekarang Anda dapat menentukan peran IAM yang digunakan cluster Anda untuk otentikasi dan otorisasi. Untuk informasi selengkapnya, lihat Role-based kontrol akses.	29 Maret 2016
Ambang pengurutan VACUUM	Perintah VACUUM sekarang melewatkan fase pengurutan untuk setiap tabel di mana lebih dari 95 persen baris tabel sudah diurutkan. Anda dapat mengubah ambang pengurutan default untuk satu tabel dengan menyertakan parameter TO PERCENT saat menjalankan VAKUM perintah.	Maret 17, 2016
Kolom baru di STL_CONNECTION_LOG	Tabel STL_CONNECTION_LOG sistem memiliki tiga kolom baru. Jika Anda secara rutin memuat log audit ke tabel Amazon Redshift, Anda perlu menambahkan kolom baru berikut ke tabel target: sslversion, sslcipher, dan mtu.	Maret 17, 2016
UNLOAD dengan kompresi bzip2	Anda sekarang memiliki opsi untuk MEMBONGKAR menggunakan kompresi bzip2.	Februari 8, 2016
UBAH TABEL DITAMBAHKAN	UBAH TABEL DITAMBAHKANmenambahkan baris ke tabel target dengan memindahkan data dari tabel sumber yang ada. ALTER TABLE APPEND biasanya jauh lebih cepat daripada operasi serupa CREATE TABLE AS atau IN INSERT TO karena data dipindahkan, tidak diduplikasi.	Februari 8, 2016

Ubah	Deskripsi	Tanggal diubah
Melompat antrian kueri WLM	Jika manajemen beban kerja (WLM) membatalkan kueri read-only, seperti pernyataan SELECT, karena batas waktu WLM, WLM mencoba merutekan kueri ke antrian pencocokan berikutnya. Untuk informasi selengkapnya, lihat Melompat antrian kueri WLM	7 Januari 2016
UBAH HAK ISTIMEWA DEFAULT	Anda dapat menggunakan UBAH HAK ISTIMEWA DEFAULT perintah untuk menentukan kumpulan hak akses default yang akan diterapkan ke objek yang dibuat di masa mendatang oleh pengguna yang ditentukan.	Desember 10, 2015
kompresi file bzip2	Per MENYALIN intah ini mendukung pemuatan data dari file yang dikompresi menggunakan bzip2.	Desember 10, 2015
NULLS PERTAMA dan NULLS TERAKHIR	Anda dapat menentukan apakah klausa ORDER BY harus memberi peringkat NULLS pertama atau terakhir dalam kumpulan hasil. Untuk informasi selengkapnya, lihat Klausa ORDER BY dan Ringkasan sintaks fungsi jendela.	19 November 2015
Kata kunci REGION untuk CREATE LIBRARY	Jika bucket Amazon S3 yang berisi file pustaka UDF tidak berada di Wil AWS ayah yang sama dengan cluster Amazon Redshift Anda, Anda dapat menggunakan opsi REGION untuk menentuka n wilayah tempat data berada. Untuk informasi selengkapnya, lihat BUAT PERPUSTAKAAN.	19 November 2015
User-defined fungsi skalar (UDF)	Anda sekarang dapat membuat fungsi skalar khusus yang ditentukan pengguna untuk mengimplementasika n fungsionalitas pemrosesan non-SQL yang disediaka n baik oleh Redshift-supported modul Amazon di Pustaka Standar Python 2.7 atau UDF kustom Anda sendiri berdasarkan bahasa pemrograman Python. Untuk informasi selengkapnya, lihat User-defined fungsi di Amazon Redshift.	11 September 2015

Ubah	Deskripsi	Tanggal diubah
Properti dinamis dalam konfigurasi WLM	Parameter konfigurasi WLM sekarang mendukung penerapan beberapa properti secara dinamis. Properti lain tetap berubah statis dan mengharuskan cluster terkait di-boot ulang sehingga perubahan konfigura si dapat diterapkan. Untuk informasi selengkapnya, lihat Properti konfigurasi dinamis dan statis WLM dan Manajemen beban kerja.	3 Agustus 2015
Fungsi LISTAGG	Dan Fungsi LISTAGG mengembalikan Fungsi jendela LISTAGG string yang dibuat dengan menggabungkan satu set nilai kolom.	30 Juli 2015
Parameter yang tidak digunakan lagi	Parameter konfigurasi max_cursor_result_set_size tidak digunakan lagi. Ukuran set hasil kursor dibatasi berdasarkan jenis node cluster. Untuk informasi selengkapnya, lihat Kendala kursor.	24 Juli 2015
Dokumentasi perintah COPY yang direvisi	Referensi MENYALIN perintah telah direvisi secara ekstensif untuk membuat materi lebih ramah dan lebih mudah diakses.	Juli 15, 2015
SALIN dari format Avro	Per MENYALIN intah ini mendukung pemuatan data dalam format Avro dari file data di Amazon S3, Amazon EMR, dan dari host jarak jauh menggunakan SSH. Untuk informasi selengkapnya, lihat AVRO dan Salin dari contoh Avro.	8 Juli 2015
STV_START UP_RECOVE RY_STATE	Tabel STV_STARTUP_RECOVERY_STATE sistem mencatat keadaan tabel yang terkunci sementara selama operasi restart cluster. Amazon Redshift menempatkan kunci sementara pada tabel saat sedang diproses untuk menyelesaikan transaksi basi setelah restart cluster.	25 Mei 2015

Ubah	Deskripsi	Tanggal diubah
ORDER BY opsional untuk fungsi peringkat	Klausa ORDER BY sekarang opsional untuk fungsi peringkat jendela tertentu. Untuk informasi selengkapnya, lihat Fungsi jendela CUME_DIST, Fungsi jendela DENSE_RANK, Fungsi jendela RANK, Fungsi jendela NTILE, Fungsi jendela PERCENT_RANK, dan Fungsi jendela ROW_NUMBER.	25 Mei 2015
Kunci pengurutan yang disisipkan	Kunci pengurutan yang disisipkan memberikan bobot yang sama untuk setiap kolom dalam kunci pengurutan. Menggunakan kunci pengurutan yang disisipkan alih-alih kunci majemuk default secara signifikan meningkatkan kinerja untuk kueri yang menggunakan predikat terbatas pada kolom pengurutan sekunder, terutama untuk tabel besar. Penyortiran interleaved juga meningkatkan kinerja keseluruhan ketika beberapa kueri memfilter pada kolom yang berbeda dalam tabel yang sama. Untuk informasi selengkapnya, lihat Urutkan kunci dan CREATE TABLE.	11 Mei 2015
Topik kinerja kueri penyetelan yang direvisi	Penyetelan kinerja kueritelah diperluas untuk memasukkan kueri baru untuk menganalisis kinerja kueri dan lebih banyak contoh. Juga, topik telah direvisi agar lebih jelas dan lebih lengkap. Praktik terbaik Amazon Redshift untuk merancang kuerimemiliki informasi lebih lanjut tentang cara menulis kueri untuk meningkatkan kinerja.	Maret 23, 2015
SVL_QUERY_QUEUE_INFO	T SVL_QUERY_QUEUE_INFO ampilan merangkum detail untuk kueri yang menghabiskan waktu dalam antrian kueri WLM atau antrian komit.	19 Februari 2015

Ubah	Deskripsi	Tanggal diubah
SVV_TABLE_INFO	Anda dapat menggunakan SVV_TABLE_INFO tampilan untuk mendiagnosis dan mengatasi masalah desain tabel yang dapat memengaruhi kinerja kueri, termasuk masalah dengan pengkodean kompresi, kunci distribusi, gaya pengurutan, kemiringan distribusi data, ukuran tabel, dan statistik.	19 Februari 2015
UNLOAD menggunakan enkripsi file sisi server	Per MEMBONGKAR intah sekarang secara otomatis menggunakan enkripsi sisi server Amazon S3 (SSE) untuk mengenkripsi semua file data yang dibongkar. Server-side enkripsi menambahkan lapisan keamanan data lain dengan sedikit atau tanpa perubahan kinerja.	Oktober 31, 2014
Fungsi jendela CUME_DIST	Mengh Fungsi jendela CUME_DIST itung distribusi kumulatif nilai dalam jendela atau partisi.	Oktober 31, 2014
Fungsi MONTHS_BETWEEN	Men Fungsi MONTHS_BETWEEN entukan jumlah bulan antara dua tanggal.	Oktober 31, 2014
Fungsi NEXT_DAY	Meng fungsi NEXT_DAY embalikan tanggal instance pertama dari hari tertentu yang lebih lambat dari tanggal yang diberikan.	Oktober 31, 2014
Fungsi jendela PERCENT_RANK	Mengh Fungsi jendela PERCENT_RANK itung peringkat persen dari baris tertentu.	Oktober 31, 2014
Fungsi jendela RATIO_TO_REPORT	Fungsi jendela RATIO_TO_REPORTMenghitung rasio nilai dengan jumlah nilai di jendela atau partisi.	Oktober 31, 2014
Fungsi TRANSLATE	Meng Fungsi TRANSLATE ganti semua kemuncula n karakter tertentu dalam ekspresi tertentu dengan pengganti yang ditentukan.	Oktober 31, 2014

Ubah	Deskripsi	Tanggal diubah
Fungsi NVL2	Meng Fungsi NVL2 embalikan salah satu dari dua nilai berdasarkan apakah ekspresi tertentu mengevaluasi ke NULL atau TIDAK NULL.	16 Oktober 2014
Fungsi jendela MEDIAN	Mengh Fungsi jendela MEDIAN itung nilai median untuk rentang nilai di jendela atau partisi.	16 Oktober 2014
PADA SEMUA TABEL DI SKEMA klausa schema_name untuk perintah GRANT dan REVOKE	Per HIBAH MENCABUT intah and telah diperbaru i dengan klausa ON ALL TABLES IN SCHEMA schema_name. Klausa ini memungkinkan Anda untuk menggunakan perintah tunggal untuk mengubah hak istimewa untuk semua tabel dalam skema.	16 Oktober 2014
Klausa IF EXISTS untuk perintah DROP SCHEMA, DROP TABLE, DROP USER, dan DROP VIEW	Per DROP SCHEMA intahMEJA DROP,JATUHKAN PENGGUNA,, dan MENJATUHKAN TAMPILAN telah diperbarui dengan klausa IF EXISTS. Klausa ini menyebabkan perintah tidak membuat perubahan dan mengembalikan pesan daripada berakhir dengan kesalahan jika objek yang ditentukan tidak ada.	16 Oktober 2014
Jika tidak ada klausa untuk perintah CREATE SCHEMA dan CREATE TABLE	Per BUAT SKEMA CREATE TABLE intah and telah diperbarui dengan klausa IF NOT EXISTS. Klausa ini menyebabkan perintah tidak membuat perubahan dan mengembalikan pesan daripada berakhir dengan kesalahan jika objek yang ditentukan sudah ada.	16 Oktober 2014
Dukungan COPY untuk UTF-16 pengkodean	Perintah COPY sekarang mendukung pemuatan dari file data yang menggunakan UTF-16 encoding serta UTF-8 encoding. Untuk informasi selengkapnya, lihat ENCODING.	29 September 2014

Ubah	Deskripsi	Tanggal diubah
Tutorial manajemen beban kerja baru	Tutorial: Mengkonfigurasi antrian manajemen beban kerja manual (WLM)memandu Anda melalui proses mengonfigurasi antrian Manajemen Beban Kerja (WLM) untuk meningkatkan pemrosesan kueri dan alokasi sumber daya.	25 September 2014
Enkripsi AES 128-bit	Perintah COPY sekarang mendukung enkripsi AES 128-bit serta enkripsi AES 256-bit saat memuat dari file data yang dienkripsi menggunakan enkripsi sisi klien Amazon S3. Untuk informasi selengkapnya, lihat Memuat file data terenkripsi dari Amazon S3.	29 September 2014
Fungsi PG_LAST_UNLOAD_COUNT	Fungsi PG_LAST_UNLOAD_COUNT mengembalikan jumlah baris yang diproses dalam operasi UNLOAD terbaru. Untuk informasi selengkapnya, lihat PG_LAST_UNLOAD_COUNT.	15 September 2014
Bagian kueri pemecahan masalah baru	Pemecahan masalah kuerimenyediakan referensi cepat untuk mengidentifikasi dan mengatasi beberapa masalah paling umum dan paling serius yang mungkin Anda temui dengan kueri Amazon Redshift.	7 Juli 2014
Tutorial memuat data baru	Tutorial: Memuat data dari Amazon S3memandu Anda melalui proses memuat data ke tabel database Amazon Redshift Anda dari file data dalam bucket Amazon S3, dari awal hingga akhir.	1 Juli 2014
Fungsi jendela PERCENTILE_CONT	Fungsi jendela PERCENTILE_CONTadalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu. Dibutuhkan nilai persentil dan spesifikasi pengurutan, dan mengembalikan nilai interpolasi yang akan jatuh ke dalam nilai persentil yang diberikan sehubungan dengan spesifikasi pengurutan.	30 Juni 2014

Ubah	Deskripsi	Tanggal diubah
Fungsi jendela PERCENTILE_DISC	Fungsi jendela PERCENTILE_DISCadalah fungsi distribusi terbalik yang mengasumsikan model distribusi diskrit. Dibutuhkan nilai persentil dan spesifikasi pengurutan dan mengembalikan elemen dari himpunan.	30 Juni 2014
Fungsi TERBESAR dan TERKECIL	Fungsi TERBESAR dan TERKECILFungsi mengembalikan nilai terbesar atau terkecil dari daftar ekspresi.	30 Juni 2014
Cross-region SALINAN	Per MENYALIN intah ini mendukung pemuatan data dari bucket Amazon S3 atau tabel Amazon DynamoDB yang terletak di wilayah yang berbeda dari cluster Amazon Redshift. Untuk informasi selengkapnya, lihat REGION referensi perintah COPY.	30 Juni 2014
Praktik Terbaik diperluas	Praktik terbaik Amazon Redshifttelah diperluas, ditata ulang, dan dipindahkan ke bagian atas hierarki navigasi untuk membuatnya lebih dapat ditemukan.	28 Mei 2014
UNLOAD ke satu file	Per MEMBONGKAR intah secara opsional dapat membongkar data tabel secara serial ke satu file di Amazon S3 dengan menambahkan opsi PARALLEL OFF. Jika ukuran data lebih besar dari ukuran file maksimum 6,2 GB, UNLOAD membuat file tambahan.	6 Mei 2014
Fungsi REGEXP	FungsiREGEXP_COUNT,REGEXP_INSTR, dan REGEXP_GANTI memanipulasi string berdasarkan pencocokan pola ekspresi reguler.	6 Mei 2014
SALIN dari Amazon EMR	Per MENYALIN intah ini mendukung pemuatan data langsung dari cluster Amazon EMR. Untuk informasi selengkapnya, lihat Memuat data dari Amazon EMR.	18 April 2014

Ubah	Deskripsi	Tanggal diubah
Kenaikan batas konkurensi WLM	Anda sekarang dapat mengonfigurasi manajemen beban kerja (WLM) untuk menjalankan hingga 50 kueri secara bersamaan dalam antrian kueri yang ditentuka n pengguna. Peningkatan ini memberi pengguna lebih banyak fleksibilitas untuk mengelola kinerja sistem dengan memodifikasi konfigurasi WLM. Untuk informasi selengkapnya, lihat Menerapkan manual WLM	18 April 2014
Parameter konfigurasi baru untuk mengelola ukuran kursor	Parameter `max_cursor_result_set_size` konfigurasi mendefinisikan ukuran maksimum data, dalam megabyte, yang dapat dikembalikan per kumpulan hasil kursor dari kueri yang lebih besar. Nilai parameter ini juga mempengaruhi jumlah kursor bersamaan untuk cluster, memungkinkan Anda untuk mengonfigurasi nilai yang meningkatkan atau mengurangi jumlah kursor untuk cluster Anda. Untuk informasi selengkapnya, lihat MENYATAKAN panduan ini dan Meng onfigurasi Ukuran Maksimum Kumpulan Hasil Kurs or di Panduan Manajemen Amazon Redshift.	Maret 28, 2014
SALIN dari format JSON	Per MENYALIN intah ini mendukung pemuatan data dalam format JSON dari file data di Amazon S3 dan dari host jarak jauh menggunakan SSH. Untuk informasi selengkapnya, lihat catatan SALIN dari format JSON penggunaan.	25 Maret 2014
Tabel sistem baru STL_PLAN_INFO	STL_PLAN_INFOTabel melengkapi perintah EXPLAIN sebagai cara lain untuk melihat rencana kueri.	25 Maret 2014
Fungsi baru REGEXP_SU BSTR	Meng Fungsi REGEXP_SUBSTR embalikan karakter yang diekstrak dari string dengan mencari pola ekspresi reguler.	25 Maret 2014

Ubah	Deskripsi	Tanggal diubah
Kolom baru untuk STL_COMMIT_STATS	STL_COMM_STATSTabel memiliki dua kolom baru: `numxids` dan`oldestxid` .	6 Maret 2014
SALIN dari dukungan SSH untuk gzip dan lzop	Per MENYALIN intah ini mendukung kompresi gzip dan lzop saat memuat data melalui koneksi SSH.	Februari 13, 2014
Fungsi baru	Meng Fungsi jendela ROW_NUMBER embalikan jumlah baris saat ini. Meng Fungsi STRTOL konversi ekspresi string dari sejumlah basis yang ditentukan ke nilai integer yang setara. PG_CANCEL_BACKENDdan PG_TERMINATE_BACKEND memungkinkan pengguna untuk membatalkan kueri dan koneksi sesi. HARI TERAKHIRFungsi telah ditambahkan untuk kompatibilitas Oracle.	Februari 13, 2014
Tabel sistem baru	Tabel STL_COMM_STATS sistem menyediakan metrik yang terkait dengan kinerja komit, termasuk waktu berbagai tahap komit dan jumlah blok yang dilakukan.	Februari 13, 2014
FETCH dengan cluster simpul tunggal	Saat menggunakan kursor pada cluster simpul tunggal, jumlah maksimum baris yang dapat diambil menggunakan AMBIL perintah adalah 1000. FETCH FORWARD ALL tidak didukung untuk cluster simpul tunggal.	Februari 13, 2014
Strategi redistribusi DS_DIST_ALL_INNER	DS_DIST_ALL_INNER dalam output rencana Explain menunjukkan bahwa seluruh tabel bagian dalam didistribusikan kembali ke satu irisan karena tabel luar menggunakan DISTSTYLE ALL. Untuk informasi selengkapnya, lihat Contoh tipe gabungan dan Mengevaluasi rencana kueri.	13 Januari 2014

Ubah	Deskripsi	Tanggal diubah
Tabel sistem baru untuk kueri	Amazon Redshift telah menambahkan tabel sistem baru yang dapat digunakan pelanggan untuk mengevaluasi eksekusi kueri untuk penyetelan dan pemecahan masalah. Untuk informasi lebih lanjut, lihatSVL_COMPILE,STL_SCAN,STL_RETURN, STL_SAVESTL_ALERT_EVENT_LOG.	13 Januari 2014
Single-node kursor	Kursor sekarang didukung untuk cluster simpul tunggal. Cluster simpul tunggal dapat memiliki dua kursor terbuka sekaligus, dengan set hasil maksimum 32 GB. Pada cluster simpul tunggal, sebaiknya setel parameter Ukuran Cache ODBC menjadi 1.000. Untuk informasi selengkapnya, lihat MENYATAKAN.	13 Desember 2013
SEMUA gaya distribusi	Distribusi SEMUA dapat secara dramatis mempersingkat waktu eksekusi untuk jenis kueri tertentu. Ketika tabel menggunakan gaya distribusi ALL, salinan tabel didistribusikan ke setiap node. Karena tabel secara efektif dikolokasikan dengan setiap tabel lainnya, tidak diperlukan redistribusi selama eksekusi kueri. Distribusi SEMUA tidak sesuai untuk semua tabel karena meningkatkan persyaratan penyimpanan dan waktu muat. Untuk informasi selengkapnya, lihat Distribusi data untuk optimasi kueri.	11 November 2013
SALIN dari host jarak jauh	Selain memuat tabel dari file data di Amazon S3 dan dari tabel Amazon DynamoDB, perintah COPY dapat memuat data teks dari cluster Amazon EMR, instans Amazon EC2, dan host jarak jauh lainnya dengan menggunakan koneksi SSH. Amazon Redshift menggunakan beberapa koneksi SSH simultan untuk membaca dan memuat data secara paralel. Untuk informasi selengkapnya, lihat Memuat data dari host jarak jauh.	11 November 2013

Ubah	Deskripsi	Tanggal diubah
Persentase memori WLM digunakan	Anda dapat menyeimbangkan beban kerja dengan menetapkan persentase memori tertentu untuk setiap antrian dalam konfigurasi manajemen beban kerja (WLM) Anda. Untuk informasi selengkapnya, lihat Menerapkan manual WLM.	11 November 2013
PERKIRAAN JUMLAH (BERBEDA)	Kueri yang menggunakan APPROXIMATE COUNT (DISTINCT) dieksekusi jauh lebih cepat, dengan kesalahan relatif sekitar 2%. Fungsi APPROXIMATE COUNT (DISTINCT) menggunakan HyperLogLog algoritma. Untuk informasi selengkapnya, lihat Fungsi COUNT.	11 November 2013
Fungsi SQL baru untuk mengambil detail kueri terbaru	Empat fungsi SQL baru mengambil detail tentang kueri terbaru dan perintah COPY. Fungsi baru membuatnya lebih mudah untuk menanyakan tabel log sistem, dan dalam banyak kasus memberikan detail yang diperlukan tanpa perlu mengakses tabel sistem. Untuk informasi lebih lanjut, lihatPG_BACKEND_PID,PG_LAST_COPY_ID,PG_LAST_COPY_COUNT,PG_LAST_QUERY_ID.	1 November 2013
Opsi MANIFEST untuk UNLOAD	Opsi MANIFEST untuk perintah UNLOAD melengkapi opsi MANIFEST untuk perintah COPY. Menggunakan opsi MANIFEST dengan UNLOAD secara otomatis membuat file manifes yang secara eksplisit mencantumkan file data yang dibuat di Amazon S3 oleh operasi pembongkaran. Anda kemudian dapat menggunakan file manifes yang sama dengan perintah COPY untuk memuat data. Untuk informasi selengkapnya, lihat Membongkar data ke Amazon S3 dan Contoh UNLOAD.	1 November 2013

Ubah	Deskripsi	Tanggal diubah
Opsi MANIFEST untuk COPY	Anda dapat menggunakan opsi MANIFEST dengan MENYALIN perintah untuk secara eksplisit mencantum kan file data yang akan dimuat dari Amazon S3.	Oktober 18, 2013
Tabel sistem untuk pertanyaa n pemecahan masalah	Menambahkan dokumentasi untuk tabel sistem yang digunakan untuk memecahkan masalah kueri. Bagian Tampilan STL untuk logging ini sekarang berisi dokumentasi untuk tabel sistem berikut: STL_AGGR, STL_BCAST, STL_DIST, STL_DELETE, STL_HASH, STL_HASHJOIN, STL_INSERT, STL_LIMIT, STL_MERGE, STL_MERGEJOIN, STL_NESTL OOP, STL_PARSE, STL_PROJECT, STL_SCAN, STL_SORT, STL_SORT, L_UNIK, STL_JENDELA.	3 Oktober 2013
Fungsi CONVERT_T IMEZONE	Meng Fungsi CONVERT_TIMEZONE konversi stempel waktu dari satu zona waktu ke zona waktu lainnya, dengan opsi untuk secara otomatis menyesuaikan waktu musim panas.	3 Oktober 2013
Fungsi SPLIT_PART	Mem Fungsi SPLIT_PART bagi string pada pembatas yang ditentukan dan mengembalikan bagian pada posisi yang ditentukan.	3 Oktober 2013
Tabel sistem STL_USERLOG	STL_USERLOGmerekam rincian untuk perubahan yang terjadi ketika pengguna database dibuat, diubah, atau dihapus.	3 Oktober 2013
Pengkodean kolom LZO dan kompresi file LZOP.	Pengkodean kompresi kolom LZO menggabungkan rasio kompresi yang sangat tinggi dengan kinerja yang baik. COPY dari Amazon S3 mendukung pemuatan dari file yang dikompresi menggunakan LZOP kompresi.	19 September 2013

Ubah	Deskripsi	Tanggal diubah
JSON, ekspresi reguler, dan kursor	Menambahkan dukungan untuk mengurai string JSON, pencocokan pola menggunakan ekspresi reguler, dan menggunakan kursor untuk mengambil kumpulan data besar melalui koneksi ODBC. Lihat informasi selengkapnya di Fungsi JSON, Pattern-matching kondisi, dan MENYATAKAN.	10 September 2013
Opsi ACCEPTINVCHAR untuk COPY	Anda dapat berhasil memuat data yang berisi UTF-8 karakter tidak valid dengan menentukan opsi ACCEPTINVCHAR dengan perintah. MENYALIN	Agustus 29, 2013
Opsi CSV untuk COPY	Per MENYALIN intah sekarang mendukung pemuatan dari file input berformat CSV.	9 Agustus 2013
CRC32	Mel Fungsi CRC32 akukan pemeriksaan redundansi siklik.	9 Agustus 2013
Karakter pengganti WLM	Manajemen beban kerja (WLM) mendukung wildcard untuk menambahkan grup pengguna dan grup kueri ke antrian. Untuk informasi selengkapnya, lihat Wildcard.	1 Agustus 2013
Batas waktu WLM	Untuk membatasi jumlah waktu kueri dalam antrian WLM tertentu diizinkan untuk digunakan, Anda dapat mengatur nilai batas waktu WLM untuk setiap antrian. Untuk informasi selengkapnya, lihat Batas waktu WLM.	1 Agustus 2013
Opsi COPY baru 'otomatis' dan 'epochsecs'	Per MENYALIN intah melakukan pengenalan otomatis format tanggal dan waktu. Format waktu baru, 'epochsecs' dan 'epochmillisecs' memungkinkan COPY untuk memuat data dalam format zaman.	Juli 25, 2013
Fungsi CONVERT_TIMEZONE	Meng Fungsi CONVERT_TIMEZONE konversi stempel waktu dari satu zona waktu ke zona waktu lainnya.	Juli 25, 2013
Fungsi FUNC_SHA1	Meng Fungsi FUNC_SHA1 konversi string menggunakan algoritma SHA1.	Juli 15, 2013

Ubah	Deskripsi	Tanggal diubah
waktu_eksekusi maksimal	Untuk membatasi jumlah waktu kueri yang diizinkan untuk digunakan, Anda dapat mengatur parameter max_execution_time sebagai bagian dari konfigurasi WLM. Untuk informasi selengkapnya, lihat Memodifikasi konfigurasi WLM.	22 Juli 2013
Four-byte UTF-8 karakter	Tipe data VARCHAR sekarang mendukung karakter empat byte UTF-8 . Five-byte atau UTF-8 karakter yang lebih panjang tidak didukung. Untuk informasi selengkapnya, lihat Penyimpanan dan rentang.	Juli 18, 2013
SVL_QERROR	Tampilan sistem SVL_QERROR telah usang.	12 Juli 2013
Riwayat Dokumen yang Direvisi	Halaman Riwayat Dokumen sekarang menunjukkan tanggal dokumentasi diperbarui.	12 Juli 2013
STL_UNLOAD_LOG	STL_UNLOAD_LOGmencatat detail untuk operasi pembongkaran.	5 Juli 2013
Parameter ukuran pengambilan JDBC	Untuk menghindari kesalahan sisi klien di luar memori saat mengambil kumpulan data besar menggunakan JDBC, Anda dapat mengaktifkan klien untuk mengambil data dalam batch dengan menyetel parameter ukuran pengambilan JDBC. Untuk informasi selengkapnya, lihat Mengatur parameter ukuran pengambilan JDBC.	27 Juni 2013
UNLOAD file terenkripsi	MEMBONGKARsekarang mendukung pembongkaran data tabel ke file terenkripsi di Amazon S3.	22 Mei 2013
Kredensial sementara	MENYALINdan MEMBONGKAR sekarang mendukung penggunaan kredenSIAL sementara.	11 April 2013
Ditambahkan klarifikasi	Diskusi yang diklarifikasi dan diperluas tentang Merancang Tabel dan Memuat Data.	Februari 14, 2013

Ubah	Deskripsi	Tanggal diubah
Menambahkan praktik terbaik	Ditambahkan Praktik terbaik Amazon Redshift untuk merancang tabel danPraktik terbaik Amazon Redshift untuk memuat data.	Februari 14, 2013
Batasan kata sandi yang diklarifikasi	Batasan kata sandi yang diklarifikasi untuk CREATE USER dan ALTER USER, berbagai revisi kecil.	Februari 14, 2013
Panduan baru	Ini adalah rilis pertama dari Panduan Pengembang Amazon Redshift.	Februari 14, 2013