Data Warehouse Service

Melhores práticas

Edição	01
Data	2025-12-02

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registadas e permissões

e outras marcas registadas da Huawei são marcas registadas da Huawei Technologies Co., Ltd. Todos as outras marcas registadas e os nomes registados mencionados neste documento são propriedade dos seus respectivos detentores.

Aviso

Os produtos, os serviços e as funcionalidades adquiridos são estipulados pelo contrato estabelecido entre a Huawei Cloud e o cliente. Os produtos, os serviços e as funcionalidades descritos neste documento, no todo ou em parte, podem não estar dentro do âmbito de aquisição ou do âmbito de uso. Salvo especificação em contrário no contrato, todas as declarações, informações e recomendações neste documento são fornecidas "TAL COMO ESTÃO" sem garantias ou representações de qualquer tipo, sejam expressas ou implícitas.

As informações contidas neste documento estão sujeitas a alterações sem aviso prévio. Foram feitos todos os esforços na preparação deste documento para assegurar a exatidão do conteúdo, mas todas as declarações, informações e recomendações contidas neste documento não constituem uma garantia de qualquer tipo, expressa ou implícita.

Índice

1 Importação e exportação

1.1 Melhores práticas para importação de dados

Importar dados do OBS em paralelo

- Dividir um arquivo de dados em vários arquivos

  Importar uma enorme quantidade de dados leva um longo período de tempo e consome muitos recursos de computação.

  Para melhorar o desempenho da importação de dados do OBS, divida um arquivo de dados em vários arquivos da forma mais uniforme possível antes de importá-lo para o OBS. O número preferencial de arquivos divididos é um múltiplo inteiro da quantidade de DN.

- Verificar arquivos de dados antes e depois de uma importação

  Ao importar dados do OBS, primeiro importe seus arquivos para o bucket do OBS e, em seguida, verifique se o bucket contém todos os arquivos corretos e apenas esses arquivos.

  Após a conclusão da importação, execute a instrução **SELECT** para verificar se os arquivos necessários foram importados.

- Garantir que não há caracteres chineses contidos em caminhos usados para importar dados para ou exportar dados do OBS.

Usar o GDS para importar dados

- A distorção de dados faz com que o desempenho da consulta se deteriore. Antes de importar todos os dados de uma tabela contendo mais de 10 milhões de registros, é aconselhável importar alguns dos dados e verificar se há distorção de dados e se as chaves de distribuição precisam ser alteradas. Solucione problemas de distorção de dados, se houver. É caro abordar a distorção de dados e alterar as chaves de distribuição após uma grande quantidade de dados ter sido importada. Para obter detalhes, consulte **Verificação de distorção de dados**.
- Para acelerar a importação, é aconselhável dividir arquivos e usar várias ferramentas do Gauss Data Service (GDS) para importar dados em paralelo. Uma tarefa de importação pode ser dividida em várias tarefas de importação simultâneas. Se várias tarefas de importação usarem o mesmo GDS, você poderá especificar o parâmetro **-t** para habilitar a importação simultânea de vários threads do GDS. Para evitar I/O física e gargalos de rede, é aconselhável montar GDSs em diferentes discos físicos e NICs.

- Se a I/O e as NICs do GDS não atingirem seus gargalos físicos, você poderá habilitar o SMP no GaussDB(DWS) para aceleração. SMP irá multiplicar a pressão sobre GDSs. Note-se que a adaptação SMP é implementada com base na pressão da CPU do GaussDB(DWS) em vez da pressão do GDS. Para obter mais informações sobre SMP, consulte **Sugestões para configurações de parâmetros de SMP**.
- Para a comunicação adequada entre GDSs e GaussDB(DWS), é aconselhável usar redes de 10GE. As redes de 1GE não podem suportar a transmissão de dados de alta velocidade e, como resultado, não podem garantir a comunicação adequada entre GDSs e GaussDB(DWS). Para maximizar a taxa de importação de um único arquivo, certifique-se de que uma rede de 10GE seja usada e que a taxa de I/O do grupo de discos de dados seja maior que o limite superior do recurso de processamento de núcleo único do GDS (cerca de 400 MB/s).
- Semelhante à importação de tabela única, certifique-se de que a taxa de I/O seja maior que a taxa de transferência máxima da rede na importação simultânea.
- Recomenda-se que a proporção entre a quantidade do GDS e a quantidade de DN esteja na faixa de 1:3 a 1:6.
- Para melhorar a eficiência da importação de dados em lotes para tabelas particionadas armazenadas em coluna, os dados são armazenados em buffer antes de serem gravados em um disco. Você pode especificar o número de buffers e o tamanho do buffer definindo a **partition_mem_batch** e **partition_max_cache_size**, respectivamente. Valores menores indicam quanto mais lenta a importação em lote para tabelas particionadas de armazenamento de colunas. Quanto maiores os valores, maior o consumo de memória.

Usar INSERT para inserir várias linhas

Se a instrução **COPY** não puder ser usada e você precisar de inserções SQL, use uma inserção de várias linhas sempre que possível. A compactação de dados é ineficiente quando você adiciona dados de apenas uma linha ou algumas linhas de cada vez.

Inserções de várias linhas melhoram o desempenho por lotes até uma série de inserções. O exemplo a seguir insere três linhas em uma tabela de três colunas usando uma única instrução **INSERT**. Esta ainda é uma pequena inserção, mostrada simplesmente para ilustrar a sintaxe de uma inserção de várias linhas. Para obter detalhes sobre como criar uma tabela, consulte **Criação de uma tabela**.

Para inserir várias linhas de dados na tabela **customer_t1**, execute a seguinte instrução:

```
INSERT INTO customer_t1 VALUES
(6885, 'maps', 'Joes'),
(4321, 'tpcds', 'Lily'),
(9527, 'world', 'James');
```

Para obter mais detalhes e exemplos, consulte **INSERT**.

Usar a instrução COPY para importar dados

A instrução **COPY** importa dados de bancos de dados locais e remotos em paralelo. **COPY** importa grandes quantidades de dados de forma mais eficiente do que as instruções **INSERT**.

Para obter detalhes sobre como usar a instrução COPY, consulte **Executação da instrução COPY FROM STDIN para importar dados**.

Usar um meta-comando de gsql para importar dados

O comando **\copy** pode ser usado para importar dados depois de fazer logon em um banco de dados por meio de qualquer cliente de **gsql**. Ao contrário da instrução **COPY**, o comando **\copy** lê ou grava em um arquivo.

Os dados lidos ou gravados usando o comando **\copy** são transferidos através da conexão entre o servidor e o cliente e podem não ser eficientes. A instrução **COPY** é recomendada quando a quantidade de dados é grande.

Para obter detalhes sobre como usar o comando **\copy**, consulte **Usar o meta-comando \copy para importar dados**.

NOTA

\copy só se aplica à importação de dados em pequenos lotes com formatos uniformes, mas capacidade de tolerância a erros fraca. GDS ou **COPY** são preferidos para a importação de dados.

1.2 Guia de prática do GDS

- Antes de instalar o GDS, certifique-se de que os parâmetros do sistema do servidor em que o GDS está implementado sejam consistentes com os do cluster de banco de dados.
- Certifique-se de que a rede física funcione corretamente para comunicação entre GDS e GaussDB(DWS). Uma rede de 10GE é recomendada. A rede de 1GE não pode garantir uma comunicação suave entre GDS e GaussDB(DWS), porque não pode suportar a pressão de transmissão de dados de alta velocidade e é propensa a desconexão. Para maximizar a taxa de importação de um único arquivo, certifique-se de que uma rede de 10GE seja usada e que a taxa de I/O do grupo de discos de dados seja maior que o limite superior do recurso de processamento de núcleo único do GDS (cerca de 400 MB/s).
- Planeje a implementação do serviço com antecedência. Recomenda-se que um ou dois GDSs sejam implantados em um RAID de um servidor de dados. Recomenda-se que a proporção entre a quantidade do GDS e a quantidade de DN esteja na faixa de 1:3 a 1:6. Não implemente muitos processos do GDS em um carregador. Implemente apenas um processo do GDS se uma NIC de 1GE for usada e não mais do que quatro processos do GDS se uma NIC de 10GE for usada.
- Divida hierarquicamente os diretórios de dados para dados importados e exportados pelo GDS com antecedência. Não coloque muitos arquivos em um diretório de dados e exclua arquivos expirados em tempo hábil.
- Planeje adequadamente o conjunto de caracteres do banco de dados de destino. É aconselhável usar UTF8 em vez dos caracteres SQL_ASCII que podem facilmente incorrer em codificação mista. Ao exportar dados usando o GDS, certifique-se de que o conjunto de caracteres da tabela estrangeira seja o mesmo do cliente. Ao importar dados, certifique-se de que o cliente e o conteúdo do arquivo de dados usam o mesmo método de codificação.
- Se o conjunto de caracteres do banco de dados, cliente ou tabela estrangeira não puder ser alterado, execute o comando **iconv** para alterar manualmente o conjunto de caracteres.

```
#Note: -f indicates the character set of the source file, and -t indicates
the target character set.
iconv -f utf8 -t gbk utf8.txt -o gbk.txt
```

- Para obter detalhes sobre as práticas de importação do GDS, consulte **Usar o GDS para importar dados**.

- O GDS suporta os formatos CSV, TEXT e FIXED. O formato padrão é TEXT. O formato binário não é suportado. No entanto, a função encode/decode pode ser usada para processar dados do tipo binário. Exemplo:

  Exporte uma tabela binária.

  ```
  -- Create a table.
  CREATE TABLE blob_type_t1
  (
      BT_COL BYTEA
  ) DISTRIBUTE BY REPLICATION;
  -- Create a foreign table.
  CREATE FOREIGN TABLE f_blob_type_t1( BT_COL  text ) SERVER gsmpp_server
  OPTIONS (LOCATION 'gsfs://127.0.0.1:7789/', FORMAT 'text', DELIMITER
  E'\x08',  NULL '', EOL '0x0a' ) WRITE ONLY;
  INSERT INTO blob_type_t1 VALUES(E'\\xDEADBEEF');
  INSERT INTO blob_type_t1 VALUES(E'\\xDEADBEEF');
  INSERT INTO blob_type_t1 VALUES(E'\\xDEADBEEF');
  INSERT INTO blob_type_t1 VALUES(E'\\xDEADBEEF');
  INSERT INTO f_blob_type_t1 select encode(BT_COL,'base64') from blob_type_t1;
  ```

  Importe uma tabela binária.

  ```
  -- Create a table.
  CREATE TABLE blob_type_t2
  (
      BT_COL BYTEA
  ) DISTRIBUTE BY REPLICATION;
  -- Create a foreign table.
  CREATE FOREIGN TABLE f_blob_type_t2( BT_COL  text ) SERVER gsmpp_server
  OPTIONS (LOCATION 'gsfs://127.0.0.1:7789/f_blob_type_t1.dat.0', FORMAT
  'text', DELIMITER E'\x08',  NULL '', EOL '0x0a' );
  insert into  blob_type_t2 select decode(BT_COL,'base64') from f_blob_type_t2;
  SELECT * FROM blob_type_t2;
     bt_col
  ------------
   \xdeadbeef
   \xdeadbeef
   \xdeadbeef
   \xdeadbeef
  (4 rows)
  ```

- Não exporte repetidamente dados da mesma tabela estrangeira. Caso contrário, o arquivo exportado anteriormente será sobrescrito.
- Se você não tiver certeza se o arquivo está no formato CSV padrão, é aconselhável definir o parâmetro **quote** para caracteres invisíveis, como **0x07**, **0x08** ou **0x1b**, para importar e exportar dados usando o GDS. Isso evita falhas de tarefa causadas por formato de arquivo incorreto.

  ```
  CREATE FOREIGN TABLE foreign_HR_staffS_ft1
  (
    MANAGER_ID     NUMBER(6),
    section_ID     NUMBER(4)
  ) SERVER gsmpp_server OPTIONS (location 'file:///input_data/*', format 'csv',
  mode 'private', quote '0x07', delimiter ',') WITH err_HR_staffS_ft1;
  ```

- O GDS suporta importação e exportação simultâneas. O parâmetro **gds -t** é usado para definir o tamanho do pool de threads e controlar o número máximo de threads de trabalho concorrentes. Mas não acelera uma única tarefa SQL. O valor padrão de **gds -t** é **8**, e o limite superior é **200**. Ao usar a função pipe para importar e exportar dados, verifique se o valor de **-t** é maior ou igual ao número de serviços simultâneos.
- Se o delimitador de uma tabela estrangeira do GDS consistir em vários caracteres, não use os mesmos caracteres no formato TEXT, por exemplo **---**.
- O GDS importa um único arquivo por meio de várias tabelas em paralelo para melhorar o desempenho da importação de dados. (Apenas arquivos CSV e TXT podem ser importados.)


```
-- Create a target table.
CREATE TABLE pipegds_widetb_1 (city integer, tel_num varchar(16), card_code
varchar(15), phone_code vcreate table pipegds_widetb_3 (city integer, tel_num
varchar(16), card_code varchar(15), phone_code varchar(16), region_code
varchar(6), station_id varchar(10), tmsi varchar(20), rec_date integer(6),
rec_time integer(6), rec_type numeric(2), switch_id  varchar(15), attach_city
varchar(6), opc varchar(20), dpc varchar(20));

-- Create a foreign table that contains the file_sequence column.
CREATE FOREIGN TABLE gds_pip_csv_r_1( like pipegds_widetb_1) SERVER
gsmpp_server  OPTIONS (LOCATION 'gsfs://127.0.0.1:8781/wide_tb.txt', FORMAT
'text', DELIMITER E'|+|', NULL '', file_sequence '5-1');

CREATE FOREIGN TABLE gds_pip_csv_r_2( like pipegds_widetb_1) SERVER
gsmpp_server  OPTIONS (LOCATION 'gsfs://127.0.0.1:8781/wide_tb.txt', FORMAT
'text', DELIMITER E'|+|', NULL '', file_sequence '5-2');

CREATE FOREIGN TABLE gds_pip_csv_r_3( like pipegds_widetb_1) SERVER
gsmpp_server  OPTIONS (LOCATION 'gsfs://127.0.0.1:8781/wide_tb.txt', FORMAT
'text', DELIMITER E'|+|', NULL '', file_sequence '5-3');

CREATE FOREIGN TABLE gds_pip_csv_r_4( like pipegds_widetb_1) SERVER
gsmpp_server  OPTIONS (LOCATION 'gsfs://127.0.0.1:8781/wide_tb.txt', FORMAT
'text', DELIMITER E'|+|', NULL '', file_sequence '5-4');

CREATE FOREIGN TABLE gds_pip_csv_r_5( like pipegds_widetb_1) SERVER
gsmpp_server  OPTIONS (LOCATION 'gsfs://127.0.0.1:8781/wide_tb.txt', FORMAT
'text', DELIMITER E'|+|', NULL '', file_sequence '5-5');

-- Import the wide_tb.txt file to the pipegds_widetb_1 table in parallel.
\parallel on
INSERT INTO pipegds_widetb_1 SELECT * FROM gds_pip_csv_r_1;
INSERT INTO pipegds_widetb_1 SELECT * FROM gds_pip_csv_r_2;
INSERT INTO pipegds_widetb_1 SELECT * FROM gds_pip_csv_r_3;
INSERT INTO pipegds_widetb_1 SELECT * FROM gds_pip_csv_r_4;
INSERT INTO pipegds_widetb_1 SELECT * FROM gds_pip_csv_r_5;
\parallel off
```

Para obter detalhes sobre **file_sequence**, consulte **CREATE FOREIGN TABLE (para importação e exportação do GDS)**.

1.3 Tutorial: importar dados do OBS para um cluster

Visão geral

Esta prática demonstra como fazer upload de dados de amostra para o OBS e importar dados do OBS para a tabela de destino no GaussDB(DWS), ajudando você a aprender rapidamente como importar dados do OBS para um cluster do GaussDB(DWS).

Você pode importar dados no formato TXT, CSV, ORC, PARQUET, CARBONDATA ou JSON do OBS para um cluster do GaussDB(DWS) para consulta.

Este tutorial usa o formato CSV como um exemplo para descrever como executar as seguintes operações:

- Gere arquivos de dados em formato CSV.
- Crie um bucket do OBS na mesma região que o cluster do GaussDB(DWS) e carregue os arquivos de dados para o bucket do OBS.
- Crie uma tabela estrangeira para importar dados do bucket do OBS para clusters do GaussDB(DWS).
- Inicie o GaussDB(DWS), crie uma tabela e importe dados do OBS para a tabela.

- Analise os erros de importação com base nas informações da tabela de erros e corrija esses erros.

Tempo estimado: 30 minutos

Preparar arquivos de dados de origem

- Arquivo de dados **product_info0.csv**

```
100,XHDK-A,2017-09-01,A,2017 Shirt Women,red,M,328,2017-09-04,715,good!
205,KDKE-B,2017-09-01,A,2017 T-shirt Women,pink,L,584,2017-09-05,40,very good!
300,JODL-X,2017-09-01,A,2017 T-shirt men,red,XL,15,2017-09-03,502,Bad.
310,QQPX-R,2017-09-02,B,2017 jacket women,red,L,411,2017-09-05,436,It's nice.
150,ABEF-C,2017-09-03,B,2017 Jeans Women,blue,M,123,2017-09-06,120,good.
```

- Arquivo de dados **product_info1.csv**

```
200,BCQP-E,2017-09-04,B,2017 casual pants men,black,L,997,2017-09-10,301,good
quality.
250,EABE-D,2017-09-10,A,2017 dress women,black,S,841,2017-09-15,299,This
dress fits well.
108,CDXK-F,2017-09-11,A,2017 dress women,red,M,85,2017-09-14,22,It's really
amazing to buy.
450,MMCE-H,2017-09-11,A,2017 jacket women,white,M,114,2017-09-14,22,very good.
260,OCDA-G,2017-09-12,B,2017 woolen coat women,red,L,2004,2017-09-15,826,Very
comfortable.
```

- Arquivo de dados **product_info2.csv**

```
980,"ZKDS-J",2017-09-13,"B","2017 Women's Cotton Clothing","red","M",112,,,
98,"FKQB-I",2017-09-15,"B","2017 new shoes men","red","M",4345,2017-09-18,5473
50,"DMQY-K",2017-09-21,"A","2017 pants
men","red","37",28,2017-09-25,58,"good","good","good"
80,"GKLW-l",2017-09-22,"A","2017 Jeans Men","red","39",58,2017-09-25,72,"Very
comfortable."
30,"HWEC-L",2017-09-23,"A","2017 shoes
women","red","M",403,2017-09-26,607,"good!"
40,"IQPD-M",2017-09-24,"B","2017 new pants
Women","red","M",35,2017-09-27,52,"very good."
50,"LPEC-N",2017-09-25,"B","2017 dress Women","red","M",29,2017-09-28,47,"not
good at all."
60,"NQAB-O",2017-09-26,"B","2017 jacket
women","red","S",69,2017-09-29,70,"It's beautiful."
70,"HWNB-P",2017-09-27,"B","2017 jacket women","red","L",30,2017-09-30,55,"I
like it so much"
80,"JKHU-Q",2017-09-29,"C","2017 T-shirt","red","M",90,2017-10-02,82,"very
good."
```

Passo 1 Crie um arquivo de texto, abra-o usando uma ferramenta de edição local (por exemplo, Visual Studio Code) e copie os dados de exemplo para o arquivo de texto.

Passo 2 Escolha **Format** > **Encode in UTF-8 without BOM**.

Passo 3 Escolha **File** > **Save as**.

Passo 4 Na caixa de diálogo exibida, insira o nome do arquivo, defina a extensão do nome de arquivo como .csv e clique em **Save**.

----Fim

Carregar dados para o OBS

Passo 1 Armazene os três arquivos de dados de origem CSV no intervalo do OBS.

1. Faça logon no console de gerenciamento do OBS.

   Clique em **Service List** e escolha **Object Storage Service** para abrir o console de gerenciamento do OBS.

2. Crie um bucket.

   Para obter detalhes sobre como criar um bucket do OBS, consulte **Criação de um bucket** em Primeiros passos no Object Storage Service.

   Por exemplo, crie dois buckets denominados **mybucket** e **mybucket02**.

   AVISO

   Certifique-se de que os dois buckets estejam na mesma região que o cluster do GaussDB(DWS). Esta prática utiliza a região CN-Hong Kong como exemplo.

3. Crie uma pasta.

   Para obter detalhes, consulte **Creating a Folder** no *Guia de operação de console do Object Storage Service* .

   Exemplos:

   – Crie uma pasta chamada **input_data** no bucket do OBS **mybucket**.
   – Crie uma pasta chamada **input_data** no bucket do OBS **mybucket02**.

4. Carregue os arquivos.

   Para obter detalhes, consulte **Carregamento de um objeto** no *Guia de operação de console do Object Storage Service*.

   Exemplos:

   – Carregue os seguintes arquivos de dados para a pasta **input_data** no bucket do OBS **mybucket**:

     ```
     product_info0.csv
     product_info1.csv
     ```

   – Carregue o seguinte arquivo de dados para a **input_data** no bucket do OBS **mybucket02**:

     ```
     product_info2.csv
     ```

Passo 2 Conceda a permissão de leitura do bucket do OBS para o usuário que importará dados.

Ao importar dados do OBS para um cluster, o usuário deve ter a permissão de leitura para os buckets do OBS onde os arquivos de dados de origem estão localizados. Você pode configurar a ACL para os buckets do OBS para conceder a permissão de leitura a um usuário específico.

Para obter detalhes, consulte **Configuração de uma ACL de bucket** no *Guia de operação de console do Object Storage Service*.

----Fim

Criar uma tabela estrangeira

Passo 1 Conecte-se ao banco de dados do GaussDB(DWS).

Passo 2 Crie uma tabela estrangeira.

NOTA

- ACCESS_KEY e SECRET_ACCESS_KEY

  Esses parâmetros especificam o AK e a SK usados para acessar o OBS por um usuário. Substitua-os pelos AK e SK reais.

  Para obter uma chave de acesso, faça logon no console de gerenciamento, mova o cursor para o nome de usuário no canto superior direito, clique em **My Credential** e clique em **Access Keys** no painel de navegação à esquerda. Na página **Access Keys**, você pode exibir os IDs de chave de acesso (AKs) existentes. Para obter o AK e a SK, clique em **Create Access Key** para criar e baixar uma chave de acesso.

- // AK e SK codificados rigidamente ou em texto não criptografado são arriscados. Para fins de segurança, criptografe seu AK e SK e armazene-os no arquivo de configuração ou nas variáveis de ambiente.

```
DROP FOREIGN TABLE IF EXISTS product_info_ext;
CREATE FOREIGN TABLE product_info_ext
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
SERVER gsmpp_server
OPTIONS(
LOCATION 'obs://mybucket/input_data/product_info | obs://mybucket02/input_data/
product_info',
FORMAT 'CSV' ,
DELIMITER ',',
ENCODING 'utf8',
HEADER 'false',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
FILL_MISSING_FIELDS 'true',
IGNORE_EXTRA_DATA 'true'
)
READ ONLY
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

Se as seguintes informações forem exibidas, a tabela estrangeira foi criada:

```
CREATE FOREIGN TABLE
```

----Fim

Importar dados

Passo 1 Crie uma tabela denominada **product_info** no banco de dados do GaussDB(DWS) para armazenar os dados importados do OBS.

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
```



```
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
WITH (
orientation = column,
compression=middle
)
DISTRIBUTE BY hash (product_id);
```

Passo 2 Execute **INSERT** para importar dados do OBS para a tabela de destino **product_info** por meio da tabela estrangeira **product_info_ext**.

```
INSERT INTO product_info SELECT * FROM product_info_ext;
```

Passo 3 Execute **SELECT** para visualizar os dados importados do OBS para GaussDB(DWS).

```
SELECT * FROM product_info;
```

As seguintes informações são exibidas no final do resultado da consulta:

```
(20 rows)
```

Passo 4 Execute **VACUUM FULL** na tabela **product_info**.

```
VACUUM FULL product_info;
```

Passo 5 Atualize as estatísticas da tabela **product_info**.

```
ANALYZE product_info;
```

----Fim

Excluir recursos

Passo 1 Se você tiver realizado consultas após a importação de dados, execute a instrução a seguir para excluir a tabela de destino:

```
DROP TABLE product_info;
```

Se a seguinte saída for exibida, a tabela estrangeira foi excluída:

```
DROP TABLE
```

Passo 2 Execute a instrução a seguir para excluir a tabela estrangeira:

```
DROP FOREIGN TABLE product_info_ext;
```

Se a seguinte saída for exibida, a tabela estrangeira foi excluída:

```
DROP FOREIGN TABLE
```

----Fim

1.4 Tutorial: usar o GDS para importar dados de um servidor remoto

Visão geral

Esta prática demonstra como usar General Data Service (GDS) para importar dados de um servidor remoto para GaussDB(DWS).

GaussDB(DWS) permite importar dados em formato TXT, CSV ou FIXED.

Neste tutorial, você irá:

- Gerar os arquivos de dados de origem no formato CSV a serem usados neste tutorial.
- Carregar os arquivos de dados de origem para um servidor de dados.
- Criar tabelas estrangeiras usadas para importar dados de um servidor de dados para GaussDB(DWS) por meio do GDS.
- Iniciar GaussDB(DWS), crie uma tabela e importe dados para a tabela.
- Analisar os erros de importação com base nas informações da tabela de erros e corrija esses erros.

Preparar um ECS como servidor do GDS

Para obter detalhes sobre como comprar um ECS, consulte "**Compra de um ECS**" em *Primeiros passos do Elastic Cloud Server*. Após a compra, faça logon no ECS consultando **Efetuar logon em um ECS de Linux**.

NOTA

- O sistema operacional do ECS deve ser suportado pelo pacote do GDS.
- O ECS e o DWS estão na mesma região, VPC e sub-rede.
- A regra do grupo de segurança do ECS deve permitir o acesso ao cluster do DWS, ou seja, a regra de entrada do grupo de segurança é a seguinte:
  - **Protocolo**: **TCP**
  - **Porta**: **5000**
  - **Origem**: selecione **IP Address** e digite o endereço IP do cluster GaussDB(DWS), por exemplo, **192.168.0.10/32**.
- Se o firewall estiver habilitado no ECS, verifique se a porta de escuta do GDS está ativada no firewall:
  ```
  iptables  -I INPUT -p tcp -m tcp --dport <gds_port> -j ACCEPT
  ```

Baixar o pacote do GDS

Passo 1 Efetue logon no console do GaussDB(DWS).

Passo 2 Na árvore de navegação à esquerda, clique em **Connections**.

Passo 3 Selecione o cliente do GDS da versão correspondente na lista suspensa de **CLI Client**.

Selecione uma versão com base na versão do cluster e no SO em que o cliente está instalado.

Passo 4 Clique em **Download**.

----Fim

Preparar arquivos de dados de origem

- Arquivo de dados **product_info0.csv**
  ```
  100,XHDK-A,2017-09-01,A,2017 Shirt Women,red,M,328,2017-09-04,715,good!
  205,KDKE-B,2017-09-01,A,2017 T-shirt Women,pink,L,584,2017-09-05,40,very good!
  300,JODL-X,2017-09-01,A,2017 T-shirt men,red,XL,15,2017-09-03,502,Bad.
  310,QQPX-R,2017-09-02,B,2017 jacket women,red,L,411,2017-09-05,436,It's nice.
  150,ABEF-C,2017-09-03,B,2017 Jeans Women,blue,M,123,2017-09-06,120,good.
  ```
- Arquivo de dados **product_info1.csv**
  ```
  200,BCQP-E,2017-09-04,B,2017 casual pants men,black,L,997,2017-09-10,301,good
  quality.
  ```



```
250,EABE-D,2017-09-10,A,2017 dress women,black,S,841,2017-09-15,299,This
dress fits well.
108,CDXK-F,2017-09-11,A,2017 dress women,red,M,85,2017-09-14,22,It's really
amazing to buy.
450,MMCE-H,2017-09-11,A,2017 jacket women,white,M,114,2017-09-14,22,very good.
260,OCDA-G,2017-09-12,B,2017 woolen coat women,red,L,2004,2017-09-15,826,Very
comfortable.
```

- Arquivo de dados **product_info2.csv**

```
980,"ZKDS-J",2017-09-13,"B","2017 Women's Cotton Clothing","red","M",112,,,
98,"FKQB-I",2017-09-15,"B","2017 new shoes men","red","M",4345,2017-09-18,5473
50,"DMQY-K",2017-09-21,"A","2017 pants
men","red","37",28,2017-09-25,58,"good","good","good"
80,"GKLW-l",2017-09-22,"A","2017 Jeans Men","red","39",58,2017-09-25,72,"Very
comfortable."
30,"HWEC-L",2017-09-23,"A","2017 shoes
women","red","M",403,2017-09-26,607,"good!"
40,"IQPD-M",2017-09-24,"B","2017 new pants
Women","red","M",35,2017-09-27,52,"very good."
50,"LPEC-N",2017-09-25,"B","2017 dress Women","red","M",29,2017-09-28,47,"not
good at all."
60,"NQAB-O",2017-09-26,"B","2017 jacket
women","red","S",69,2017-09-29,70,"It's beautiful."
70,"HWNB-P",2017-09-27,"B","2017 jacket women","red","L",30,2017-09-30,55,"I
like it so much"
80,"JKHU-Q",2017-09-29,"C","2017 T-shirt","red","M",90,2017-10-02,82,"very
good."
```

Passo 1 Crie um arquivo de texto, abra-o usando uma ferramenta de edição local (por exemplo, Visual Studio Code) e copie os dados de exemplo para o arquivo de texto.

Passo 2 Escolha **Format** > **Encode in UTF-8 without BOM**.

Passo 3 Escolha **File** > **Save as**.

Passo 4 Na caixa de diálogo exibida, insira o nome do arquivo, defina a extensão do nome de arquivo como.csv e clique em **Save**.

Passo 5 Efetue logon no servidor do GDS como usuário **root**.

Passo 6 Crie o diretório **/input_data** para armazenar o arquivo de dados.

```
mkdir -p /input_data
```

Passo 7 Use MobaXterm para fazer upload dos arquivos de dados de origem para o diretório criado.

----Fim

Instalar e iniciar o GDS

Passo 1 Faça logon no servidor do GDS como usuário **root** e crie o diretório **/opt/bin/dws** para armazenar o pacote do GDS.

```
mkdir -p /opt/bin/dws
```

Passo 2 Carregue o pacote do GDS para o diretório criado.

Por exemplo, carregue o pacote **dws_client_8.1.*x*_redhat_x64.zip** para o diretório criado.

Passo 3 Vá para o diretório e descompactar o pacote.

```
cd /opt/bin/dws
unzip dws_client_8.1.x_redhat_x64.zip
```

Passo 4 Crie um usuário (**gds_user**) e o grupo de usuários (**gdsgrp**) ao qual o usuário pertence. Este usuário é usado para iniciar o GDS e deve ter permissão para ler o diretório do arquivo de dados de origem.


```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

Passo 5 Altere o proprietário do pacote do GDS e o diretório do arquivo de dados de origem para **gds_user** e altere o grupo de usuários para **gdsgrp**.

```
chown -R gds_user:gdsgrp /opt/bin/dws/gds
chown -R gds_user:gdsgrp /input_data
```

Passo 6 Mude para o usuário **gds_user**.

```
su - gds_user
```

Se a versão atual do cluster for 8.0.*x* ou anterior, pule **Passo 7** e vá para **Passo 8**.

Se a versão atual do cluster for 8.1.*x* ou posterior, vá para a próxima etapa.

Passo 7 Execute o script do qual o ambiente depende (aplicável apenas a 8.1.*x*).

```
cd /opt/bin/dws/gds/bin
source gds_env
```

Passo 8 Inicie o GDS.

```
/opt/bin/dws/gds/bin/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
l /opt/bin/dws/gds/gds_log.txt -D
```

Substitua as peças em itálico conforme necessário.

- **-d** *dir*: diretório para armazenar arquivos de dados que contêm dados a serem importados. Esta prática usa **/input_data/** como um exemplo.
- **-p** *ip:port*: endereço IP de escuta e porta para GDS. O valor padrão é **127.0.0.1**. Substitua-o pelo endereço IP de uma rede 10GE com a qual possa se comunicar GaussDB(DWS). O número da porta varia de 1024 a 65535. O valor padrão é **8098**. Esta prática usa **192.168.0.90:5000** como um exemplo.
- **-H** *address_string*: hosts que têm permissão para se conectar e usar o GDS. O valor deve estar no formato CIDR. Defina este parâmetro para permitir que um cluster de GaussDB(DWS) acesse o GDS para importação de dados. Certifique-se de que o segmento de rede cubra todos os hosts em um cluster de GaussDB(DWS).
- **-l** *log_file*: diretório de log do GDS e nome do arquivo de log. Esta prática usa **/opt/bin/dws/gds/gds_log.txt** como um exemplo.
- **-D**: GDS em modo daemon. Este parâmetro é usado apenas no Linux.

----Fim

Criar uma tabela estrangeira

Passo 1 Use um cliente de SQL para se conectar ao banco de dados de GaussDB(DWS).

Passo 2 Crie a seguinte tabela estrangeira:

CUIDADO

LOCATION: substitua-o pelo endereço do GDS real e número da porta.

```
DROP FOREIGN TABLE IF EXISTS product_info_ext;
CREATE FOREIGN TABLE product_info_ext
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
```



```
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
SERVER gsmpp_server
OPTIONS(
LOCATION 'gsfs://192.168.0.90:5000/*',
FORMAT 'CSV' ,
DELIMITER ',',
ENCODING 'utf8',
HEADER 'false',
FILL_MISSING_FIELDS 'true',
IGNORE_EXTRA_DATA 'true'
)
READ ONLY
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

Se as seguintes informações forem exibidas, a tabela estrangeira foi criada:

```
CREATE FOREIGN TABLE
```

----Fim

Importar dados

Passo 1 Execute as instruções a seguir para criar a tabela **product_info** no GaussDB(DWS) para armazenar os dados importados:

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
WITH (
orientation = column,
compression=middle
)
DISTRIBUTE BY hash (product_id);
```

Passo 2 Importe dados de arquivos de dados de origem para a tabela **product_info** por meio da tabela estrangeira **product_info_ext**.

```
INSERT INTO product_info SELECT * FROM product_info_ext ;
```

Se as seguintes informações forem exibidas, os dados foram importados:

```
INSERT 0 20
```

Passo 3 Execute a instrução **SELECT** para exibir os dados importados para GaussDB(DWS).

```
SELECT count(*) FROM product_info;
```

Se as seguintes informações forem exibidas, os dados foram importados:


```
count
-------
     20
(1 row)
```

Passo 4 Execute **VACUUM FULL** na tabela **product_info**.

```
VACUUM FULL product_info
```

Passo 5 Atualize as estatísticas da tabela **product_info**.

```
ANALYZE product_info;
```

----Fim

Interromper o GDS

Passo 1 Faça logon no servidor de dados em que o GDS está instalado como usuário **gds_user**.

Passo 2 Execute as seguintes operações para parar o GDS:

1. Consulte o ID do processo do GDS. O ID do processo do GDS é 128954.

   ```
   ps -ef|grep gds
   gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data/ -p
   192.168.0.90:5000 -l /opt/bin/gds/gds_log.txt  -D
   gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
   ```

2. Execute o comando **kill** para interromper o GDS. **128954** indica o ID do processo do GDS.

   ```
   kill -9 128954
   ```

----Fim

Excluir recursos

Passo 1 Execute o seguinte comando para excluir a tabela de destino **product_info**:

```
DROP TABLE product_info;
```

Se as seguintes informações forem exibidas, a tabela foi excluída:

```
DROP TABLE
```

Passo 2 Execute o seguinte comando para excluir a tabela estrangeira **product_info_ext**:

```
DROP FOREIGN TABLE product_info_ext;
```

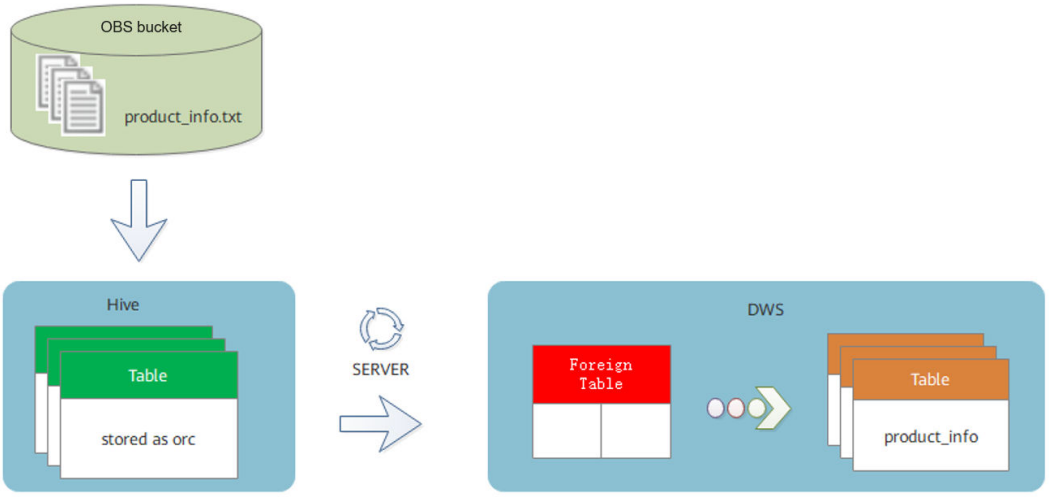
Se as seguintes informações forem exibidas, a tabela foi excluída:

```
DROP FOREIGN TABLE
```

----Fim

1.5 Tutorial: exibir ou importar dados de Hive do MRS

Neste tutorial, uma tabela estrangeira HDFS é criada para permitir que o GaussDB(DWS) acesse remotamente ou leia fontes de dados do MRS.



Preparação do ambiente

Crie um cluster de GaussDB(DWS). Verifique se os clusters do MRS e do GaussDB(DWS) estão na mesma região, AZ e sub-rede da VPC e se os clusters podem se comunicar uns com os outros.

Procedimento

Esta prática leva cerca de 1 hora. O processo básico é o seguinte:

1. Crie um cluster de análise do MRS (selecione Hive, Spark e Tez).
2. Você pode fazer upload de um arquivo de dados TXT local para um bucket do OBS, importar o arquivo para o Hive por meio do bucket do OBS e importar o arquivo da tabela de armazenamento TXT para a tabela de armazenamento ORC.
3. Crie uma conexão de fonte de dados do MRS.
4. Crie um servidor estrangeiro.
5. Crie uma tabela estrangeira,
6. Importe a tabela local do DWS através de uma tabela estrangeira.

Criar um cluster do MRS

Passo 1 Faça logon no **consolo da Huawei Cloud**, escolha **Analytics** > **MapReduce Service** e clique em **Buy Cluster**. Clique na guia **Custom Config**, configure os parâmetros do software e clique em **Next**.

Tabela 1-1 Configuração de software

Parâmetro	Valor
Region	CN-Hong Kong
Cluster Name	mrs_01
Version	Normal

Parâmetro	Valor
Cluster Version	MRS 1.9.2 (recomendado) **NOTA** ● Para clusters da versão 8.1.1.300 e posteriores, os clusters do MRS suportam as versões *, 1.7.*, 1.8.*, 1.9.*, 2.0.*, 3.0.*, 3.1.* e posteriores (* indica um número). ● Para clusters anteriores à versão 8.1.1.300, os clusters do MRS suportam as versões 1.6.*, 1.7.*, 1.8.*, 1.9.* e 2.0.* (* indica um número).
Cluster Type	Analysis Cluster
Metadata	Local

Passo 2 Configure os parâmetros de hardware e clique em **Next**.

Tabela 1-2 Configuração de hardware

Parâmetro	Valor
Billing Mode	Pay-per-use
AZ	AZ2
VPC	vpc-01
Subnet	subnet-01
Security Group	Auto create
EIP	*10.x.x.x*
Enterprise Project	default
Master	2
Analysis Core	3
Analysis Task	0

Passo 3 Quando você tiver concluído as configurações avançadas com base na tabela a seguir, clique em **Buy Now** e aguarde cerca de 15 minutos. O cluster foi criado com êxito.

Tabela 1-3 Configuração avançada

Parâmetro	Valor
Tag	test01
Hostname Prefix	(Opcional) Prefixo para o nome de um ECS ou BMS no cluster.
Auto Scaling	Mantenha o valor padrão.
Bootstrap Action	Mantenha o valor padrão. O MRS 3.x não suporta este parâmetro.

Parâmetro	Valor
Agency	Mantenha o valor padrão.
Data Disk Encryption	Esta função está desativada por padrão. Mantenha o valor padrão.
Alarm	Mantenha o valor padrão.
Rule Name	Mantenha o valor padrão.
Topic Name	Selecione um tópico.
Kerberos Authentication	Este parâmetro é ativado por padrão.
Username	admin
Password	Essa senha é usada para efetuar logon na página de gerenciamento de cluster.
Confirm Password	Digite a senha do usuário **admin** novamente.
Login Mode	Password
Username	root
Password	Essa senha é usada para efetuar logon remotamente no ECS.
Confirm Password	Digite a senha do usuário **root** novamente.
Secure Communications	Selecione **Enable**.

----Fim

Preparar a fonte de dados da tabela ORC do MRS

Passo 1 Crie um arquivo **product_info.txt** no PC local, copie os seguintes dados para o arquivo e salve o arquivo no PC local.

```
100,XHDK-A-1293-#fJ3,2017-09-01,A,2017 Autumn New Shirt
Women,red,M,328,2017-09-04,715,good
205,KDKE-B-9947-#kL5,2017-09-01,A,2017 Autumn New Knitwear
Women,pink,L,584,2017-09-05,406,very good!
300,JODL-X-1937-#pV7,2017-09-01,A,2017 autumn new T-shirt
men,red,XL,1245,2017-09-03,502,Bad.
310,QQPX-R-3956-#aD8,2017-09-02,B,2017 autumn new jacket
women,red,L,411,2017-09-05,436,It's really super nice
150,ABEF-C-1820-#mC6,2017-09-03,B,2017 Autumn New Jeans
Women,blue,M,1223,2017-09-06,1200,The seller's packaging is exquisite
200,BCQP-E-2365-#qE4,2017-09-04,B,2017 autumn new casual pants
men,black,L,997,2017-09-10,301,The clothes are of good quality.
250,EABE-D-1476-#oB1,2017-09-10,A,2017 autumn new dress
women,black,S,841,2017-09-15,299,Follow the store for a long time.
108,CDXK-F-1527-#pL2,2017-09-11,A,2017 autumn new dress
women,red,M,85,2017-09-14,22,It's really amazing to buy
450,MMCE-H-4728-#nP9,2017-09-11,A,2017 autumn new jacket
women,white,M,114,2017-09-14,22,Open the package and the clothes have no odor
260,OCDA-G-2817-#bD3,2017-09-12,B,2017 autumn new woolen coat
women,red,L,2004,2017-09-15,826,Very favorite clothes
```



```
980,ZKDS-J-5490-#cW4,2017-09-13,B,2017 Autumn New Women's Cotton
Clothing,red,M,112,2017-09-16,219,The clothes are small
98,FKQB-I-2564-#dA5,2017-09-15,B,2017 autumn new shoes
men,green,M,4345,2017-09-18,5473,The clothes are thick and it's better this
winter.
150,DMQY-K-6579-#eS6,2017-09-21,A,2017 autumn new underwear
men,yellow,37,2840,2017-09-25,5831,This price is very cost effective
200,GKLW-l-2897-#wQ7,2017-09-22,A,2017 Autumn New Jeans
Men,blue,39,5879,2017-09-25,7200,The clothes are very comfortable to wear
300,HWEC-L-2531-#xP8,2017-09-23,A,2017 autumn new shoes
women,brown,M,403,2017-09-26,607,good
100,IQPD-M-3214-#yQ1,2017-09-24,B,2017 Autumn New Wide Leg Pants
Women,black,M,3045,2017-09-27,5021,very good.
350,LPEC-N-4572-#zX2,2017-09-25,B,2017 Autumn New Underwear
Women,red,M,239,2017-09-28,407,The seller's service is very good
110,NQAB-O-3768-#sM3,2017-09-26,B,2017 autumn new underwear
women,red,S,6089,2017-09-29,7021,The color is very good
210,HWNB-P-7879-#tN4,2017-09-27,B,2017 autumn new underwear
women,red,L,3201,2017-09-30,4059,I like it very much and the quality is good.
230,JKHU-Q-8865-#uO5,2017-09-29,C,2017 Autumn New Clothes with Chiffon
Shirt,black,M,2056,2017-10-02,3842,very good
```

Passo 2 Efetue logon no console do OBS, clique em **Create Bucket**, configure os seguintes parâmetros e clique em **Create Now**.

Tabela 1-4 Parâmetros do bucket

Parâmetro	Valor
Region	CN-Hong Kong
Data Redundancy Policy	Single-AZ Storage
Bucket Name	mrs-datasource
Default Storage Class	Standard
Bucket Policy	Private
Default Encryption	Disable
Direct Reading	Disable
Enterprise Project	default
Tags	-

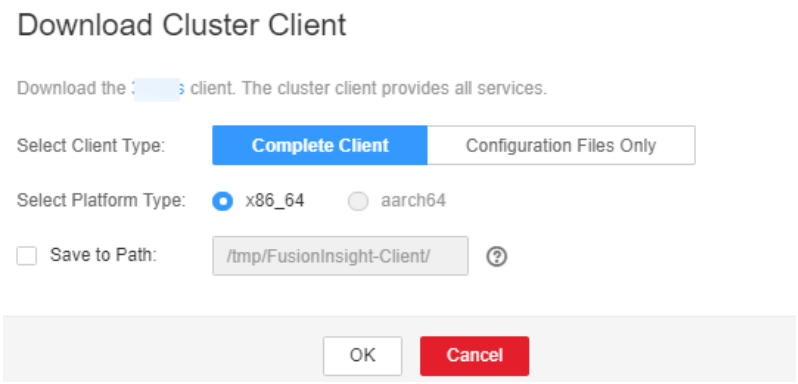
Passo 3 Depois que o bucket for criado, clique no nome do bucket e escolha **Object** > **Upload Object** para fazer upload do arquivo **product_info.txt** para o bucket do OBS.

Passo 4 Volte para o console do MRS e clique no nome do cluster do MRS criado. Na página **Dashboard**, clique no botão Synchronize ao lado de **IAM User Sync**. A sincronização leva cerca de 5 minutos.

Passo 5 Clique em **Nodes** e clique em um nó principal. Na página exibida, alterne para a guia **EIPs**, clique em **Bind EIP**, selecione um EIP existente e clique em **OK**. Se nenhum EIP estiver disponível, crie um. Registre o EIP.

Passo 6 Baixe o cliente.

1. Volte para a página de cluster do MRS. Clique no nome do cluster. Na página de guia **Dashboard** da página de detalhes do cluster, clique em **Access Manager**. Se uma mensagem for exibida indicando que o EIP precisa ser vinculado, vincule um EIP primeiro.
2. Na caixa de diálogo**Access MRS Manager**, clique em **OK**. Você será redirecionado para a página de logon do MRS Manager. Digite o nome de usuário **admin** e sua senha para fazer logon no MRS Manager. A senha é aquela que você digitou ao criar o cluster do MRS.
3. Escolha **Cluster** > *Name of the desired cluster* > **Dashboard** > **More** > **Download Client**. A caixa de diálogo **Download Cluster Client** é exibida.



NOTA

Para obter o cliente de uma versão anterior, escolha **Services** > **Download Client** e defina **Select Client Type** como **Configuration Files Only**.

Passo 7 Determine o nó principal ativo.

1. Use o SSH para fazer logon no nó anterior como usuário **root**. Execute o seguinte comando para alternar para o usuário **omm**:

   su - omm
2. Execute o seguinte comando para consultar o nó principal ativo. Na saída do comando, o nó cujo valor de **HAActive** está **active** é o nó principal ativo.

   sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

Passo 8 Efetue logon no nó principal ativo como usuário **root** e atualize a configuração do cliente do nó de gerenciamento ativo.

cd /opt/client

sh refreshConfig.sh /opt/client *Full_path_of_client_configuration_file_package*

Neste tutorial, execute o seguinte comando:

sh refreshConfig.sh /opt/client /tmp/MRS-client/MRS_Services_Client.tar

Passo 9 Alterne para o usuário **omm** e vá para o diretório onde o cliente de Hive está localizado.

su - omm

cd /opt/client

Passo 10 Crie a tabela **product_info** cujo formato de armazenamento é TEXTFILE no Hive.

1. Importe variáveis de ambiente para o diretório **/opt/client**.

source bigdata_env

2. Efetue logon no cliente de Hive.

   beeline

3. Execute os seguintes comandos SQL em seqüência para criar um banco de dados de demonstração e a tabela **product_info**:

```
CREATE DATABASE demo;
USE demo;
DROP TABLE product_info;

CREATE TABLE product_info
(
    product_price                int            ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    int            ,
    product_comment_time         date           ,
    product_comment_num          int        ,
    product_comment_content      varchar(200)
)
row format delimited fields terminated by ','
stored as TEXTFILE;
```

Passo 11 Importe o arquivo **product_info.txt** para o Hive.

1. Volte para o cluster MRS, clique em **Files** > **Import Data**.
2. **OBS Path**: localize o arquivo **product_info.txt** no bucket do OBS criado e clique em **Yes**.
3. **HDFS Path**: selecione **/user/hive/warehouse/demo.db/product_info/** e clique em **Yes**.
4. Clique em **OK** para importar os dados da tabela **product_info**.

Passo 12 Crie uma tabela ORC e importe dados para a tabela.

1. Execute os seguintes comandos SQL para criar uma tabela ORC:

```
DROP TABLE product_info_orc;

CREATE TABLE product_info_orc
(
    product_price                int            ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    int            ,
    product_comment_time         date           ,
    product_comment_num          int            ,
    product_comment_content      varchar(200)
)
row format delimited fields terminated by ','
stored as orc;
```

2. Insira dados na tabela **product_info** na tabela ORC do Hive **product_info_orc**.

```
insert into product_info_orc select * from product_info;
```

3. Consulte se a importação de dados foi bem-sucedida.

```
select * from product_info_orc;
```

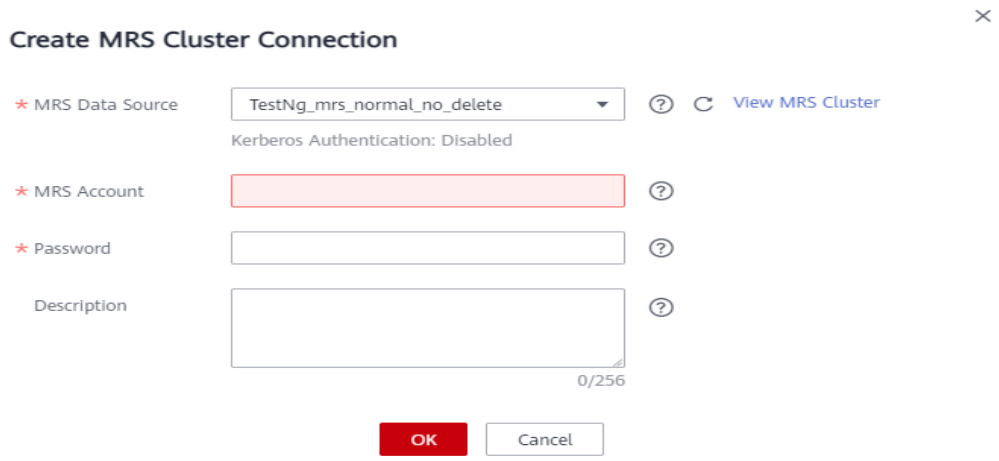
----Fim

Criar uma conexão de fonte de dados do MRS

Passo 1 Faça logon no console do GaussDB(DWS) e clique no cluster de armazém de dados criado. Verifique se os clusters do GaussDB(DWS) e MRS estão na mesma região, AZ e sub-rede da VPC.

Passo 2 Clique na guia **MRS Data Source** e clique em **Create MRS Cluster Connection**.

Passo 3 Selecione a origem de dados **mrs_01** criada na etapa anterior, insira o nome da conta do MRS **admin** e sua senha e clique em **OK**.



----Fim

Criar um servidor estrangeiro

Passo 1 Use o Data Studio para conectar-se ao cluster do GaussDB(DWS) criado.

Passo 2 Crie um usuário *dbuser* que tenha permissão para criar bancos de dados.

```
CREATE USER dbuser WITH CREATEDB PASSWORD 'password';
```

Passo 3 Mude para o usuário *dbuser*.

```
SET ROLE dbuser PASSWORD 'password';
```

Passo 4 Crie um banco de dados *mydatabase*.

```
CREATE DATABASE mydatabase;
```

Passo 5 Execute as seguintes etapas para alternar para o banco de dados *mydatabase*:

1. Na janela **Object Browser** do cliente do Data Studio, clique com o botão direito do mouse na conexão de banco de dados e selecione **Refresh** no menu de atalho. O novo banco de dados é exibido.
2. Clique com o botão direito do mouse no nome do banco de dados *mydatabase* e selecione **Connect to DB** no menu de atalho.
3. Clique com o botão direito do mouse no nome do banco de dados *mydatabase* e selecione **Open Terminal** no menu de atalho. A janela de comando SQL para conexão com um banco de dados é exibida. Execute os seguintes passos na janela.

Passo 6 Conceda a permissão para criar servidores externos ao usuário dbuser. Em 8.1.1 e versões posteriores, você também precisa conceder a permissão para usar o modo público.


```
GRANT ALL ON FOREIGN DATA WRAPPER hdfs_fdw TO dbuser;
In GRANT ALL ON SCHEMA public TO dbuser; //8.1.1 and later versions, common users
do not have permission on the public mode and need to grant permission. In
versions earlier than 8.1.1, you do not need to perform this operation.
```

O nome do **FOREIGN DATA WRAPPER** deve ser **hdfs_fdw**. *dbuser* indica o nome de usuário de **CREATE SERVER**.

Passo 7 Conceda ao usuário *dbuser* a permissão para usar tabelas estrangeiras.

```
ALTER USER dbuser USEFT;
```

Passo 8 Alterne para o banco de dados Postgres e consulte o servidor estrangeiro criado automaticamente pelo sistema após a criação da fonte de dados do MRS.

```
SELECT * FROM pg_foreign_server;
```

Informação semelhante à seguinte é exibida:

```
                    srvname                        | srvowner | srvfdw | srvtype |
srvversion | srvacl |
srvoptions
-----------------------------------------------+----------+--------+---------
+------------+--------
+-------------------------------------------------------------------------------
-----------------------------------
 gsmpp_server                                  |       10 |  13673 |
|            |        |
 gsmpp_errorinfo_server                        |       10 |  13678 |
|            |        |
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |
|            |        |
{"address=192.168.1.245:9820,192.168.1.218:9820",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(3 rows)
```

Passo 9 Mudar para a base de dados *mydatabase* e mudar para o usuário *dbuser*.

```
SET ROLE dbuser PASSWORD 'password';
```

Passo 10 Crie um servidor estrangeiro.

O nome do servidor, o endereço e o caminho de configuração devem ser os mesmos em **Passo 8**.

```
CREATE SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca FOREIGN DATA
WRAPPER HDFS_FDW
OPTIONS
(
address '192.168.1.245:9820,192.168.1.218:9820',   //The intranet IP addresses of
the active and standby master nodes on the MRS management plane, which can be
used to communicate with GaussDB(DWS).
hdfscfgpath '/MRS/8f79ada0-d998-4026-9020-80d6de2692ca',
type 'hdfs'
);
```

Passo 11 Veja o servidor estrangeiro.

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

O servidor é criado com êxito se forem apresentadas informações semelhantes às seguintes:

```
                    srvname                        | srvowner | srvfdw | srvtype |
srvversion | srvacl |
srvoptions
-----------------------------------------------+----------+--------+---------
+------------+--------
+-------------------------------------------------------------------------------
-----------------------------------
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |
```



```
|              |          |
{"address=192.168.1.245:9820,192.168.1.218:9820",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(1 row)
```

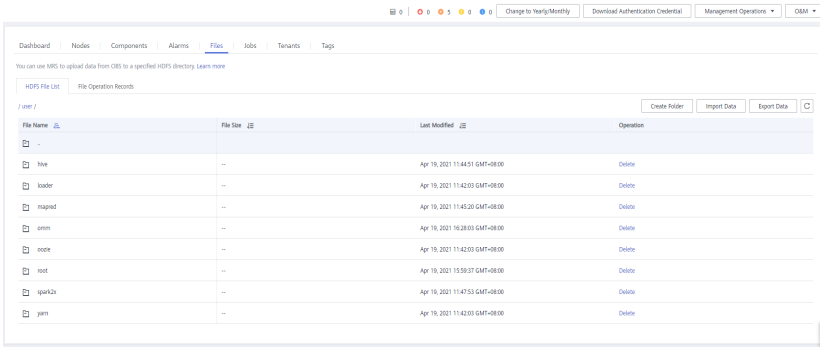
----Fim

Criar uma tabela estrangeira

Passo 1 Obtenha o caminho do arquivo **product_info_orc** do Hive.

1. Efetue logon no console do MRS.
2. Escolha **Cluster** > **Active Cluster** e clique no nome do cluster a ser consultado para entrar na página que exibe as informações básicas do cluster.
3. Clique em **Files** e clique em **HDFS File List**.
4. Vá para o diretório de armazenamento dos dados a serem importados para o cluster do GaussDB(DWS) e registre o caminho.

Figura 1-1 Verificar o caminho de armazenamento de dados no MRS



Passo 2 Crie uma tabela estrangeira. Defina **SERVER** como o nome do servidor externo criado em **Passo 10** e **foldername** como o caminho obtido em **Passo 1**.

```
DROP FOREIGN TABLE IF EXISTS foreign_product_info;

CREATE FOREIGN TABLE foreign_product_info
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca
OPTIONS (
format 'orc',
encoding 'utf8',
foldername '/user/hive/warehouse/demo.db/product_info_orc/'
)
DISTRIBUTE BY ROUNDROBIN;
```

----Fim

Importação de dados

Passo 1 Crie uma tabela local para importação de dados.

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

Passo 2 Importe dados para a tabela de destino a partir da tabela estrangeira.

```
INSERT INTO product_info SELECT * FROM foreign_product_info;
```

Passo 3 Consulte o resultado da importação.

```
SELECT * FROM product_info;
```

----Fim

1.6 Tutorial: importar fontes de dados do GaussDB(DWS) remotas

Na era da análise convergente de Big Data, os clusters do GaussDB(DWS) na mesma região podem se comunicar uns com os outros. Esta prática demonstra como importar dados de um cluster do GaussDB(DWS) remoto para o cluster do GaussDB(DWS) local usando tabelas estrangeiras.

O procedimento de demonstração é o seguinte: instale o cliente de banco de dados gsql em um ECS, conecte-se ao GaussDB(DWS) usando gsql e importe dados do GaussDB(DWS) remoto usando uma tabela estrangeira.

Procedimento geral

Essa prática leva cerca de 40 minutos. O processo básico é o seguinte:

1. **Preparativos**
2. **Criar um ECS**
3. **Criar um cluster e baixar o pacote de ferramentas**
4. **Importar fontes de dados usando o GDS**
5. **Importar dados do GaussDB(DWS) remoto usando uma tabela estrangeira**

Preparativos

Você registrou uma conta da Huawei e ativou a Huawei Cloud. A conta não pode estar em atraso ou congelada.

Criar um ECS

Para obter detalhes, consulte **Compra de um ECS**. Após a compra de um ECS, faça logon no ECS consultando **Efetuar logon em um ECS de Linux**.

AVISO

Ao criar um ECS, verifique se o ECS e os clusters do GaussDB(DWS) a serem criados estão na mesma sub-rede da VPC e na mesma região e AZ . O SO do ECS é o mesmo do cliente de gsql ou GDS (o CentOS 7.6 é usado como exemplo) e a senha é usada para logon.

Criar um cluster e baixar o pacote de ferramentas

Passo 1 Faça logon no console de gerenciamento da Huawei Cloud.

Passo 2 Escolha **Service List** > **Analytics** > **Data Warehouse Service**. Na página exibida, clique em **Create Cluster** no canto superior direito.

Passo 3 Configure parâmetros de acordo com **Tabela 1-5**.

Tabela 1-5 Configuração de software

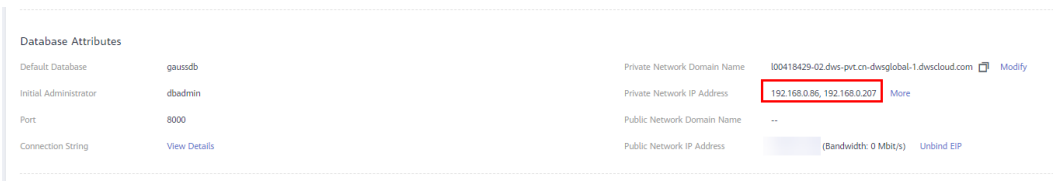
Parâmetro	Configuração
Region	Selecione CN-Hong Kong . **NOTA** ● **CN-Hong Kong** é usado como exemplo. Você pode selecionar outras regiões, conforme necessário. Certifique-se de que todas as operações sejam realizadas na mesma região. ● Verifique se o GaussDB(DWS) e o ECS estão na mesma região, AZ e sub-rede da VPC.
AZ	AZ2
Resource	Armazém de dados padrão
Compute Resource	ECS
Storage Type	SSD em nuvem
CPU Architecture	x86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD) **NOTA** Se esse flavor estiver esgotado, selecione outras AZs ou flavors.

Parâmetro	Configuração
Hot Storage	100 GB/nó
Nodes	3
Cluster Name	dws-demo01
Administrator Account	dbadmin
Administrator Password	Senha definida pelo usuário
Confirm Password	Senha
Database Port	8000
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24) **AVISO** Verifique se o cluster e o ECS estão na mesma sub-rede da VPC.
Security Group	Automatic creation
EIP	Buy now
Bandwidth	1 Mbit/s
Advanced Settings	Padrão

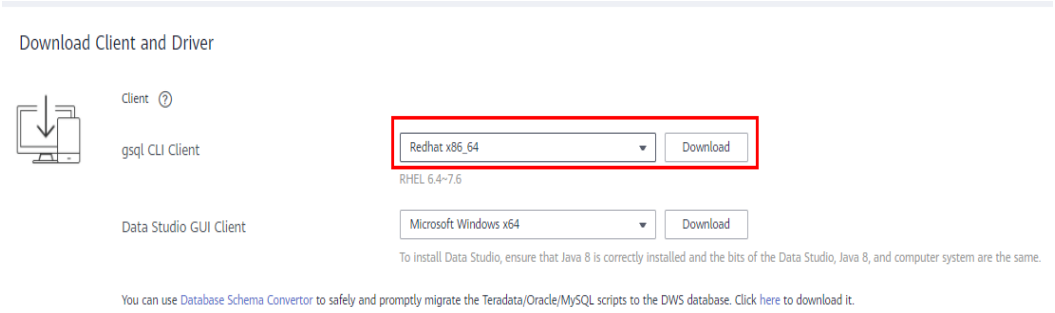
Passo 4 Confirme as informações, clique em **Next** e, em seguida, clique em **Submit**.

Passo 5 Aguarde cerca de 10 minutos. Depois que o cluster for criado, clique no nome do cluster para ir para a página **Basic Information**. Escolha **Network**, clique em um nome de grupo de segurança e verifique se uma regra de grupo de segurança foi adicionada. Neste exemplo, o endereço IP do cliente é 192.168.0.*x* (o endereço IP da rede privada do ECS onde o gsql está localizado é 192.168.0.90). Portanto, você precisa adicionar uma regra de grupo de segurança na qual o endereço IP é 192.168.0.0/24 e o número da porta é 8000.

Passo 6 Retorne à guia **Basic Information** do cluster e registre o valor de **Private Network IP Address**.



Passo 7 Retorne à página inicial do console do GaussDB(DWS). Escolha **Connections** no painel de navegação à esquerda, selecione o SO do ECS (por exemplo, selecione **Redhat x86_64** para CentOS 7.6) e clique em **Download** para salvar o pacote de ferramentas no host local. O pacote de ferramentas contém o cliente de gsql e o GDS.



Passo 8 Repita **Passo 1** a **Passo 6** para criar um segundo cluster do GaussDB(DWS) e defina seu nome como **dws-demo02**.

----Fim

Preparar dados de origem

Passo 1 Crie os três arquivos CSV a seguir no diretório especificado no PC local:

- Arquivo de dados **product_info0.csv**

```
100,XHDK-A,2017-09-01,A,2017 Shirt Women,red,M,328,2017-09-04,715,good!
205,KDKE-B,2017-09-01,A,2017 T-shirt Women,pink,L,584,2017-09-05,40,very good!
300,JODL-X,2017-09-01,A,2017 T-shirt men,red,XL,15,2017-09-03,502,Bad.
310,QQPX-R,2017-09-02,B,2017 jacket women,red,L,411,2017-09-05,436,It's nice.
150,ABEF-C,2017-09-03,B,2017 Jeans Women,blue,M,123,2017-09-06,120,good.
```

- Arquivo de dados **product_info1.csv**

```
200,BCQP-E,2017-09-04,B,2017 casual pants men,black,L,997,2017-09-10,301,good
quality.
250,EABE-D,2017-09-10,A,2017 dress women,black,S,841,2017-09-15,299,This
dress fits well.
108,CDXK-F,2017-09-11,A,2017 dress women,red,M,85,2017-09-14,22,It's really
amazing to buy.
450,MMCE-H,2017-09-11,A,2017 jacket women,white,M,114,2017-09-14,22,very good.
260,OCDA-G,2017-09-12,B,2017 woolen coat women,red,L,2004,2017-09-15,826,Very
comfortable.
```

- Arquivo de dados **product_info2.csv**

```
980,"ZKDS-J",2017-09-13,"B","2017 Women's Cotton Clothing","red","M",112,,,
98,"FKQB-I",2017-09-15,"B","2017 new shoes men","red","M",4345,2017-09-18,5473
50,"DMQY-K",2017-09-21,"A","2017 pants
men","red","37",28,2017-09-25,58,"good","good","good"
80,"GKLW-l",2017-09-22,"A","2017 Jeans Men","red","39",58,2017-09-25,72,"Very
comfortable."
30,"HWEC-L",2017-09-23,"A","2017 shoes
women","red","M",403,2017-09-26,607,"good!"
40,"IQPD-M",2017-09-24,"B","2017 new pants
Women","red","M",35,2017-09-27,52,"very good."
50,"LPEC-N",2017-09-25,"B","2017 dress Women","red","M",29,2017-09-28,47,"not
good at all."
60,"NQAB-O",2017-09-26,"B","2017 jacket
women","red","S",69,2017-09-29,70,"It's beautiful."
70,"HWNB-P",2017-09-27,"B","2017 jacket women","red","L",30,2017-09-30,55,"I
like it so much"
80,"JKHU-Q",2017-09-29,"C","2017 T-shirt","red","M",90,2017-10-02,82,"very
good."
```


Passo 2 Efetue logon no ECS criado como usuário **root** e execute o seguinte comando para criar um diretório de arquivos de origem de dados:

mkdir -p /input_data

Passo 3 Use uma ferramenta de transferência de arquivos para carregar os arquivos de dados anteriores para o diretório **/input_data** do ECS.

----Fim

Importar fontes de dados usando o GDS

Passo 1 Faça logon no ECS como usuário **root** e use uma ferramenta de transferência de arquivos para carregar o pacote de ferramentas baixado em **Passo 7** ao diretório **/opt**.

Passo 2 Descompacte o pacote de ferramentas no diretório **/opt**.

cd /opt

unzip dws_client_8.1.x_redhat_x64.zip

Passo 3 Crie um usuário do GDS e altere os proprietários da fonte de dados e dos diretórios do GDS.

groupadd gdsgrp

useradd -g gdsgrp gds_user

chown -R gds_user:gdsgrp /opt/gds

chown -R gds_user:gdsgrp /input_data

Passo 4 Mude para o usuário **gds_user**.

su - gds_user

Passo 5 Importe as variáveis de ambiente do GDS.

NOTA

Esta etapa é necessária apenas para 8.1.*x* ou posterior. Para versões anteriores, pule esta etapa.

cd /opt/gds/bin

source gds_env

Passo 6 Inicie o GDS.

/opt/gds/bin/gds -d /input_data/ -p *192.168.0.90:5000* **-H** 192.168.0.0/24 **-l /opt/gds/gds_log.txt -D**

- **-d** *dir*: diretório para armazenar arquivos de dados que contêm dados a serem importados. Esta prática usa **/input_data/** como um exemplo.
- **-p** *ip:port*: endereço IP de escuta e porta para GDS. Defina este parâmetro para o endereço IP da rede privada do ECS onde o GDS está instalado para que o GDS possa se comunicar com o GaussDB (DWS). Neste exemplo, **192.168.0.90:5000** é usado.
- **-H** *address_string*: hosts que têm permissão para se conectar e usar o GDS. O valor deve estar no formato CIDR. Neste exemplo, o segmento de rede do endereço IP da rede privada do GaussDB(DWS) é usado.
- **-l** *log_file*: diretório de log do GDS e nome do arquivo de log. Neste exemplo, **/opt/gds/gds_log.txt** é usado.

- **-D**: GDS em modo daemon.

Passo 7 Conecte-se ao primeiro cluster do GaussDB(DWS) usando gsql.

1. Execute o comando **exit** para alternar para o usuário **root**, vá para o diretório **/opt** do ECS e importe as variáveis de ambiente do gsql.

   exit

   cd /opt

   source gsql_env.sh

2. Vá para o diretório **/opt/bin** e conecte-se ao primeiro cluster do GaussDB(DWS) usando gsql.

   cd /opt/bin

   gsql -d gaussdb -h 192.168.0.8 -p 8000 -U dbadmin -W *password* -r

   - **-d**: nome do banco de dados conectado. Neste exemplo, o banco de dados padrão **gaussdb** é usado.
   - **-h**: endereço IP da rede privada do banco de dados do GaussDB(DWS) conectado consultado em **Passo 6**. Neste exemplo, **192.168.0.8** é usado.
   - **-p**: porta do GaussDB(DWS). O valor é **8000**.
   - **-U**: administrador do banco de dados. O valor padrão é **dbadmin**.
   - **-W**: senha do administrador, que é definida durante a criação do cluster em **Passo 3**. Neste exemplo, substitua *password* pela senha real.

Passo 8 Crie um usuário comum **leo** e conceda ao usuário a permissão para criar tabelas estrangeiras.

```
CREATE USER leo WITH PASSWORD 'password';
ALTER USER leo USEFT;
```

Passo 9 Mude para o usuário **leo** e crie uma tabela estrangeira do GDS.

NOTA

Defina **LOCATION** como o endereço IP de escuta do GDS e o número da porta obtidos em **Passo 6**, por exemplo, **gsfs://192.168.0.90:5000/***.

```
SET ROLE leo PASSWORD 'password';
DROP FOREIGN TABLE IF EXISTS product_info_ext;
CREATE FOREIGN TABLE product_info_ext
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
SERVER gsmpp_server
OPTIONS(
LOCATION 'gsfs://192.168.0.90:5000/*',
FORMAT 'CSV' ,
DELIMITER ',',
ENCODING 'utf8',
HEADER 'false',
FILL_MISSING_FIELDS 'true',
IGNORE_EXTRA_DATA 'true'
)
READ ONLY
```



```
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

Passo 10 Crie uma tabela local.

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
WITH (
orientation = column,
compression=middle
)
DISTRIBUTE BY hash (product_id);
```

Passo 11 Importe dados da tabela estrangeira do GDS e verifique se os dados foram importados com êxito.

```
INSERT INTO product_info SELECT * FROM product_info_ext ;
SELECT count(*) FROM product_info;
```

----Fim

Importar dados do GaussDB(DWS) remoto usando uma tabela estrangeira

Passo 1 Conecte-se ao segundo cluster no ECS fazendo referência a **Passo 7**. Altere o endereço de conexão para o endereço do segundo cluster. Neste exemplo, **192.168.0.86** é usado.

Passo 2 Crie um usuário comum **jim** e conceda ao usuário a permissão para criar tabelas e servidores estrangeiros. O valor de **FOREIGN DATA WRAPPER** é **gc_fdws**.

```
CREATE USER jim WITH PASSWORD 'password';
ALTER USER jim USEFT;
GRANT ALL ON FOREIGN DATA WRAPPER gc_fdw TO jim;
```

Passo 3 Mude para o usuário **jim** e crie um servidor.

```
SET ROLE jim PASSWORD 'password';
CREATE SERVER server_remote FOREIGN DATA WRAPPER gc_fdw OPTIONS
   (address '192.168.0.8:8000,192.168.0.158:8000' ,
  dbname 'gaussdb',
  username 'leo',
  password 'password'
);
```

- **address**: endereços IP de rede privada e número de porta do primeiro cluster obtido em **Passo 6**. Neste exemplo, **192.168.0.8:8000** e **192.168.0.158:8000** são usados.
- **dbname**: nome do banco de dados do primeiro cluster conectado. Neste exemplo, **gaussdb** é usado.
- **username**: nome do usuário do primeiro cluster conectado. Neste exemplo, **leo** é usado.
- **password**: senha do usuário

Passo 4 Crie uma tabela estrangeira.

AVISO

As colunas e as restrições da tabela estrangeira devem ser consistentes com as da tabela a ser acessada.

```
CREATE FOREIGN TABLE region
(
    product_price                integer      ,
    product_id                   char(30)     ,
    product_time                 date         ,
    product_level                char(10)     ,
    product_name                 varchar(200) ,
    product_type1                varchar(20)  ,
    product_type2                char(10)     ,
    product_monthly_sales_cnt    integer      ,
    product_comment_time         date         ,
    product_comment_num          integer      ,
    product_comment_content      varchar(200)
)
SERVER
    server_remote
OPTIONS
(
    schema_name 'leo',
    table_name 'product_info',
    encoding 'utf8'
);
```

- **SERVER**: nome do servidor criado no passo anterior. Neste exemplo, **server_remote** é usado.
- **schema_name**: nome do esquema do primeiro cluster a ser acessado. Neste exemplo, **leo** é usado.
- **table_name**: nome da tabela do primeiro cluster a ser acessado obtido em **Passo 10**. Neste exemplo, **product_info** é usado.
- **encoding**: o valor deve ser o mesmo do primeiro cluster obtido em **Passo 9**. Neste exemplo, **utf8** é usado.

Passo 5 Visualize o servidor criado e a tabela estrangeira.

```
\des+ server_remote
\d+ region
```

Passo 6 Crie uma tabela local.

AVISO

As colunas e restrições da tabela devem ser consistentes com as da tabela a ser acessada.

```
CREATE TABLE local_region
(
    product_price                integer      not null,
    product_id                   char(30)     not null,
    product_time                 date         ,
    product_level                char(10)     ,
    product_name                 varchar(200) ,
    product_type1                varchar(20)  ,
    product_type2                char(10)     ,
    product_monthly_sales_cnt    integer      ,
    product_comment_time         date         ,
    product_comment_num          integer      ,
    product_comment_content      varchar(200)
```



```
)

WITH (
orientation = column,
compression=middle
)
DISTRIBUTE BY hash (product_id);
```

Passo 7 Importe dados para a tabela local usando a tabela estrangeira.

```
INSERT INTO local_region SELECT * FROM region;
SELECT * FROM local_region;
```

Passo 8 Consulte a tabela estrangeira sem importar dados.

```
SELECT * FROM region;
```

----Fim

1.7 Tutorial: exportar dados do ORC para MRS

GaussDB(DWS) permite exportar dados do ORC para MRS usando uma tabela estrangeira de HDFS. Você pode especificar o modo de exportação e o formato de dados de exportação na tabela estrangeira. Os dados são exportados do GaussDB(DWS) em paralelo usando vários DNs e armazenados no HDFS. Desta forma, o desempenho geral das exportações é melhorado.

Preparação do ambiente

Crie um cluster de GaussDB(DWS). Verifique se os clusters do MRS e do GaussDB(DWS) estão na mesma região, AZ e sub-rede da VPC e se os clusters podem se comunicar uns com os outros.

Criar um cluster do MRS

Passo 1 Faça logon no **console da Huawei Cloud**, escolha **Analytics** > **MapReduce Service** e clique em **Buy Cluster**. Clique na guia **Custom Config**, configure os parâmetros do software e clique em **Next**.

Tabela 1-6 Configuração de software

Parâmetro	Exemplo de valor
Region	CN-Hong Kong
Cluster Name	mrs_01
Cluster Version	MRS 1.9.2 (recomendado) **NOTA** ● Para clusters da versão 8.1.1.300 e posteriores, os clusters do MRS suportam as versões *, 1.7.*, 1.8.*, 1.9.*, 2.0.*, 3.0.*, 3.1.* e posteriores (* indica um número). ● Para clusters anteriores à versão 8.1.1.300, os clusters do MRS suportam as versões 1.6.*, 1.7.*, 1.8.*, 1.9.* e 2.0.* (* indica um número).
Cluster Type	Cluster de análise

Passo 2 Configure os parâmetros de hardware e clique em **Next**.

Tabela 1-7 Configuração de hardware

Parâmetro	Exemplo de valor
Billing Mode	Pay-per-use
AZ	AZ2
VPC	vpc-01
Subnet	subnet-01
Security Group	Auto create
EIP	*10.x.x.x*
Enterprise Project	default
Master	2
Analysis Core	3
Analysis Task	0

Passo 3 Configure as configurações avançadas com base na tabela a seguir, clique em **Buy Now** e aguarde cerca de 15 minutos para que a criação do cluster seja concluída.

Tabela 1-8 Configurações avançadas

Parâmetro	Exemplo de valor
Tag	test01
Hostname Prefix	(Opcional) Prefixo para o nome de um ECS ou BMS no cluster.
Auto Scaling	mantenha o valor padrão.
Bootstrap Action	Mantenha o valor padrão. O MRS 3.x não suporta este parâmetro.
Agency	Mantenha o valor padrão.
Data Disk Encryption	Esta função está desativada por padrão. Mantenha o valor padrão.
Alarm	Mantenha o valor padrão.
Rule Name	Mantenha o valor padrão.
Topic Name	Selecione um tópico.
Kerberos Authentication	Este parâmetro é ativado por padrão.
User Name	admin

Parâmetro	Exemplo de valor
Password	Essa senha é usada para efetuar logon na página de gerenciamento de cluster.
Confirm Password	Digite a senha do usuário **admin** novamente.
Login Mode	Password
User Name	root
Password	Essa senha é usada para efetuar logon remotamente no ECS.
Confirm Password	Digite a senha do usuário **root** novamente.
Secure Communications	Selecione **Enable**.

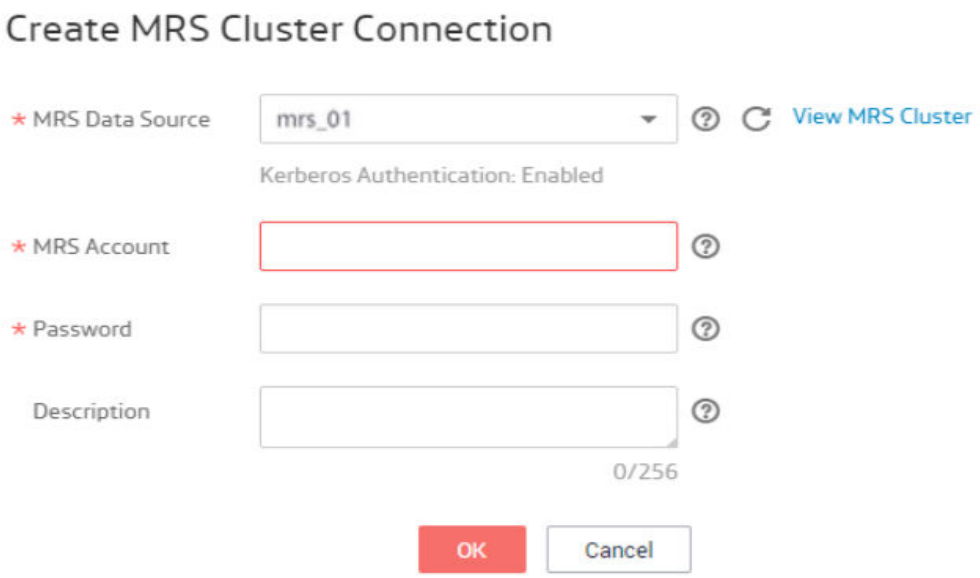
----Fim

Criar uma conexão de fonte de dados do MRS

Passo 1 Faça logon no console do GaussDB(DWS) e clique no cluster de armazém de dados criado. Verifique se os clusters do GaussDB(DWS) e MRS estão na mesma região, AZ e sub-rede da VPC.

Passo 2 Clique na guia **MRS Data Source** e clique em **Create MRS Cluster Connection**.

Passo 3 Selecione a origem de dados **mrs_01** criada na etapa anterior, insira o nome da conta do MRS **admin** e sua senha e clique em **OK**.



----Fim

Criar um servidor estrangeiro

Passo 1 Use o Data Studio para conectar-se ao cluster do GaussDB(DWS) criado.

Passo 2 Crie um usuário *dbuser* que tenha permissão para criar bancos de dados.

```
CREATE USER dbuser WITH CREATEDB PASSWORD 'password';
```

Passo 3 Mude para o usuário *dbuser*.

```
SET ROLE dbuser PASSWORD 'password';
```

Passo 4 Crie um banco de dados *mydatabase*.

```
CREATE DATABASE mydatabase;
```

Passo 5 Execute as seguintes etapas para alternar para o banco de dados *mydatabase*:

1. Na janela **Object Browser** do cliente do Data Studio, clique com o botão direito do mouse na conexão de banco de dados e selecione **Refresh** no menu de atalho. Em seguida, o novo banco de dados é exibido.
2. Clique com o botão direito do mouse no nome do banco de dados *mydatabase* e selecione **Connect to DB** no menu de atalho.
3. Clique com o botão direito do mouse no nome do banco de dados *mydatabase* e selecione **Open Terminal** no menu de atalho. A janela de comando SQL para conexão com um banco de dados é exibida. Execute os seguintes passos na janela.

Passo 6 Conceda a permissão para criar servidores externos ao usuário dbuser. Em 8.1.1 e versões posteriores, você também precisa conceder a permissão para usar o modo público.

```
GRANT ALL ON FOREIGN DATA WRAPPER hdfs_fdw TO dbuser;
In GRANT ALL ON SCHEMA public TO dbuser; //8.1.1 and later versions, common users
do not have permission on the public mode and need to grant permission. In
versions earlier than 8.1.1, you do not need to perform this operation.
```

O nome do **FOREIGN DATA WRAPPER** deve ser **hdfs_fdw**. *dbuser* indica o nome de usuário de **CREATE SERVER**.

Passo 7 Conceda ao usuário *dbuser* a permissão para usar tabelas estrangeiras.

```
ALTER USER dbuser USEFT;
```

Passo 8 Alterne para o banco de dados Postgres e consulte o servidor estrangeiro criado automaticamente pelo sistema após a criação da fonte de dados do MRS.

```
SELECT * FROM pg_foreign_server;
```

Informação semelhante à seguinte é exibida:

```
                     srvname                          | srvowner | srvfdw | srvtype |
srvversion | srvacl |
srvoptions
-----------------------------------------------------+----------+--------+---------
+------------+--------
+-----------------------------------------------------------------------------------
-------------------------------------
 gsmpp_server                                        |       10 |  13673 |
|            |        |
 gsmpp_errorinfo_server                              |       10 |  13678 |
|            |        |
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |
|            |        |
{"address=192.168.1.245:9820,192.168.1.218:9820",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(3 rows)
```

Passo 9 Mudar para a base de dados *mydatabase* e mudar para o usuário *dbuser*.

```
SET ROLE dbuser PASSWORD 'password';
```


Passo 10 Crie um servidor estrangeiro.

O nome do servidor, o endereço e o caminho de configuração devem ser os mesmos em **Passo 8**.

```
CREATE SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca FOREIGN DATA
WRAPPER HDFS_FDW
OPTIONS
(
address '192.168.1.245:9820,192.168.1.218:9820',   //The intranet IP addresses of
the active and standby master nodes on the MRS management plane, which can be
used to communicate with GaussDB(DWS).
hdfscfgpath '/MRS/8f79ada0-d998-4026-9020-80d6de2692ca',
type 'hdfs'
);
```

Passo 11 Veja o servidor estrangeiro.

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

O servidor é criado com êxito se forem apresentadas informações semelhantes às seguintes:

```
                     srvname                      | srvowner | srvfdw | srvtype |
srvversion | srvacl |
srvoptions
----------------------------------------------------+----------+--------+---------
+------------+--------
+-------------------------------------------------------------------------------
-----------------------------------
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |
|            |        |
{"address=192.168.1.245:9820,192.168.1.218:9820",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(1 row)
```

----Fim

Criar uma tabela estrangeira

Crie uma tabela estrangeira do OBS que não contenha colunas de partição. O servidor externo associado à tabela é **hdfs_server**, o formato do arquivo no HDFS correspondente à tabela é ORC e o caminho de armazenamento de dados no OBS é **/user/hive/warehouse/product_info_orc/**.

```
DROP FOREIGN TABLE IF EXISTS product_info_output_ext;
CREATE FOREIGN TABLE product_info_output_ext
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca
OPTIONS (
format 'orc',
foldername '/user/hive/warehouse/product_info_orc/',
   compression 'snappy',
    version '0.12'
) Write Only;
```


Exportação de dados

Crie uma tabela comum **product_info_output**.

```
DROP TABLE product_info_output;
CREATE TABLE product_info_output
(
    product_price                int            ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    int            ,
    product_comment_time         date           ,
    product_comment_num          int        ,
    product_comment_content      varchar(200)
)
with (orientation = column,compression=middle)
distribute by hash (product_name);
```

Exporte dados da tabela **product_info_output** para um arquivo de dados usando a tabela estrangeira **product_info_output_ext**.

```
INSERT INTO product_info_output_ext SELECT * FROM product_info_output;
```

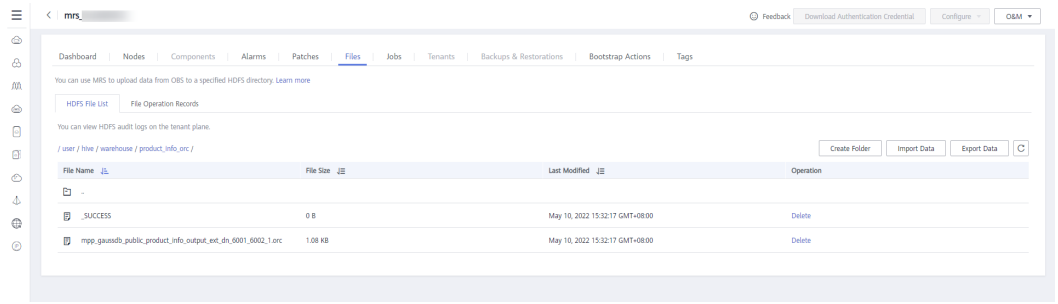
Se forem exibidas informações semelhantes às seguintes, os dados forem criados.

```
INSERT 0 10
```

Exibir o resultado da exportação

Passo 1 Vá para a lista de clusters do MRS. Clique em um nome de cluster para ir para a página de detalhes do cluster.

Passo 2 Clique em **Files** e clique em **HDFS File List**. Verifique o arquivo ORC exportado no diretório **user/hive/warehouse/product_info_orc**.



NOTA

Os dados de ORC exportados do GaussDB(DWS) estão em conformidade com as seguintes regras:

1. Dados exportados para MRS (HDFS): quando os dados são exportados de um DN, os dados são armazenados no HDFS no formato de segmento. O arquivo é nomeado no formato de **mpp**_*DatabaseName_SchemaName_TableName_NodeName_n*.**orc**.
2. É aconselhável exportar dados de diferentes clusters ou bancos de dados para diferentes caminhos. O tamanho máximo de um arquivo ORC é 128 MB e o de um arquivo de faixas é 64 MB.
3. Após a conclusão da exportação, o arquivo **_SUCCESS** é gerado.

----Fim

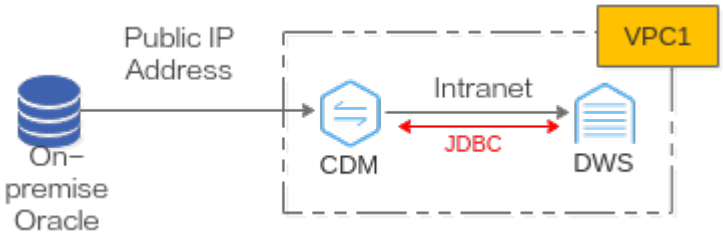
2 Migração de dados

2.1 Migração de dados do Oracle para GaussDB(DWS)

2.1.1 Progresso de migração

Este tutorial demonstra como migrar dados de tabela do Oracle para GaussDB(DWS). **Figura 2-2** e **Tabela 2-1** mostram o processo de migração.

Figura 2-1 Cenários de migração



AVISO

- Esta prática descreve como migrar dados na tabela **APEX2_DYNAMIC_ADD_REMAIN_TEST** do usuário **db_user01** no banco de dados de Oracle.
- Conexões de rede: nesta prática, o banco de dados de Oracle é implantado no local, então o CDM é usado para conectar o Oracle ao GaussDB(DWS). O CDM se conecta ao Oracle por meio de um endereço IP público. CDM e GaussDB(DWS) estão na mesma região e VPC e podem se comunicar uns com os outros. **Certifique-se de que toda a rede esteja conectada durante a migração.**
- Esta prática é apenas para referência. A migração real pode ser complexa devido a fatores como o ambiente de rede, a complexidade do serviço, a escala de nós e o volume de dados. É melhor realizar a migração sob a orientação de pessoal técnico.

Figura 2-2 Processo básico de migração de dados do Oracle para o GaussDB(DWS)

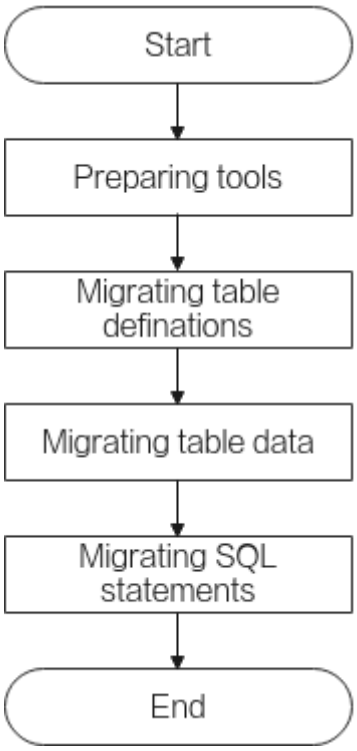


Tabela 2-1 Processo básico de migração de dados do Oracle para o GaussDB(DWS)

Processo	Descrição
Ferramentas necessárias	Ferramentas de software a serem preparadas antes da migração.
Migração de definições de tabela	Use o PL/SQL Developer para migrar definições de tabela.
Migração de dados de tabela completa	Use o Cloud Data Migration Service (CDM) da Huawei Cloud para migrar dados.
Migração de instruções SQL	Use a ferramenta de migração de sintaxe de DSC para reescrever a sintaxe para que as instruções SQL do serviço de Oracle possam ser adaptadas ao GaussDB(DWS).

2.1.2 Ferramentas necessárias

As ferramentas necessárias para a migração incluem PL/SQL Developer, Instant Client e DSC. Para obter detalhes sobre como baixar as ferramentas, consulte **Tabela 2-2**.

Tabela 2-2 Ferramentas necessárias

Ferramenta	Descrição	Endereço de download
PL/SQL Developer	Ferramenta de desenvolvimento visual do Oracle	**Endereço de baixar o PL/SQL Developer**
Oracle Instant Client	Cliente de Oracle	**Endereço de baixar o Instant Client**
DSC	Ferramenta de migração de sintaxe para GaussDB(DWS)	**Endereço de baixar o DSC**

2.1.3 Migração de definições de tabela

2.1.3.1 Instalação do PL/SQL Developer no host local

Procedimento

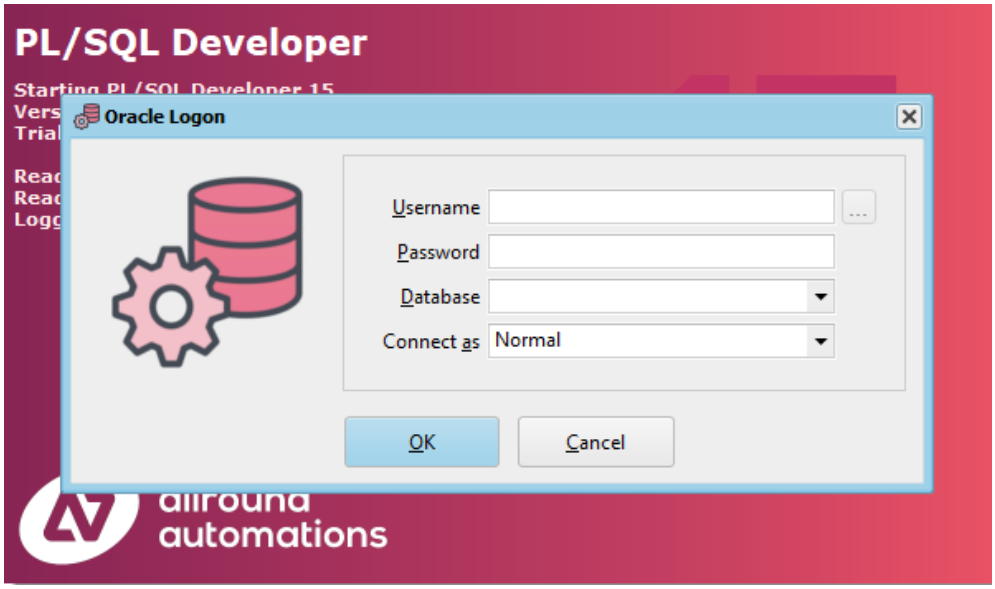
Passo 1 Descompacte os pacotes PL/SQL Developer, Instant Client e DSC.

Passo 2 Configure um Oracle home e uma biblioteca OCL para o PL/SQL Developer.

NOTA

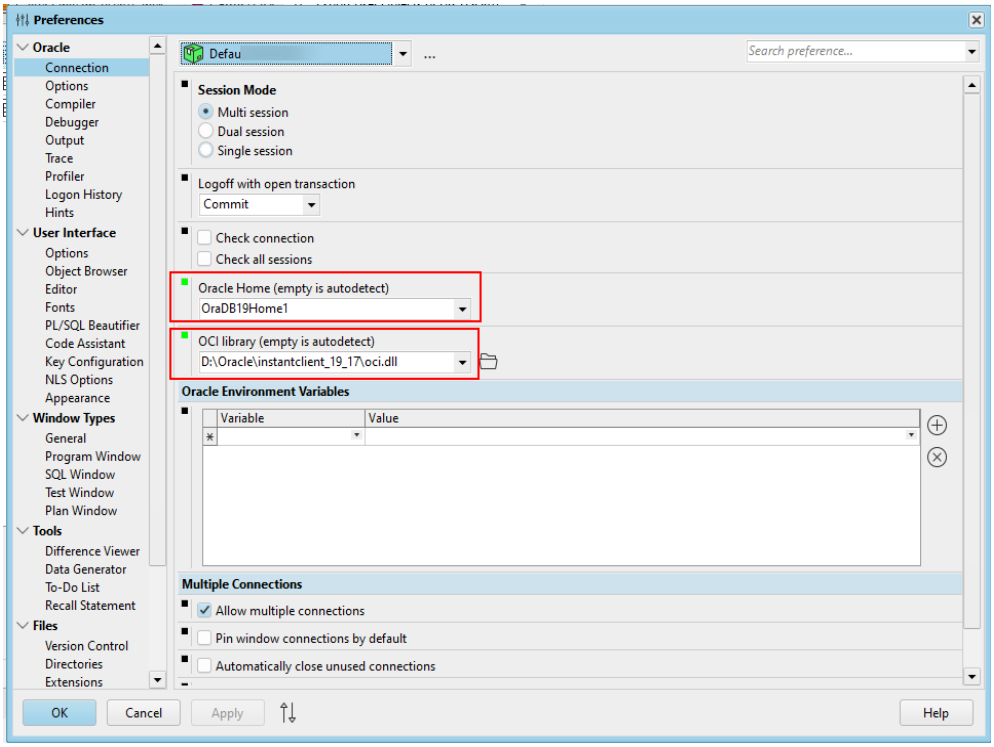
A seguir, o PL/SQL Developer Trial Version é usado como exemplo.

1. Na página de logon, clique em **Cancel**.

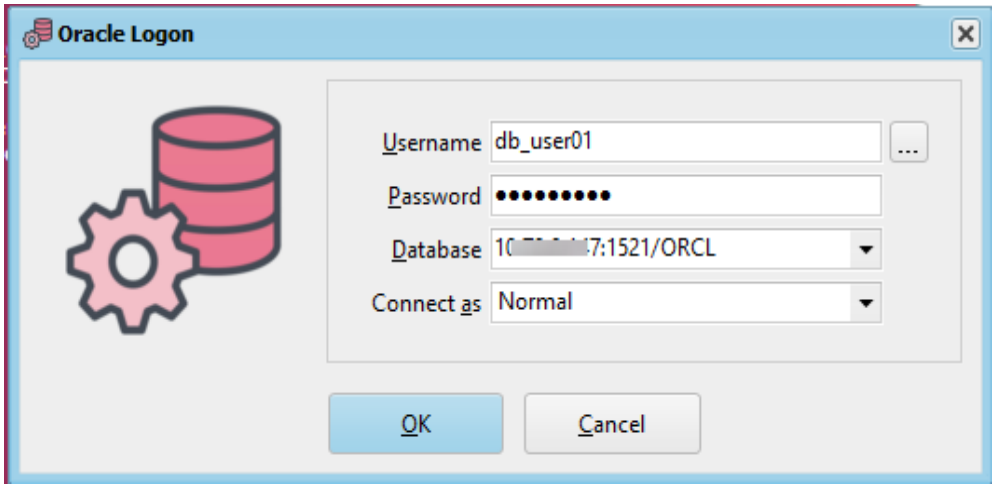


2. Escolha **Configure** > **Preferences** > **Connection** e adicione as configurações de Oracle Home e biblioteca OCl.
3. Copie o caminho do instantclient obtido de **Passo 1** (por exemplo, **D:\Oracle\instantclient_19_17\oci.dll**) para o diretório home do banco de dados do Oracle.

Copie o caminho do arquivo **oci.dll** (por exemplo, **D:\Oracle\instantclient_19_17\oci.dll**) no arquivo instantclient para a biblioteca OCI.



Passo 3 Volte para a página de logon do PL/SQL Developer. Digite o nome de usuário, a senha e o endereço do banco de dados, por exemplo, xx.xx.xx.xx:1521/ORCL.



Passo 4 Clique em **OK**. Se o banco de dados estiver conectado, isso indica que o PL/SQL Developer foi instalado com êxito.

----Fim

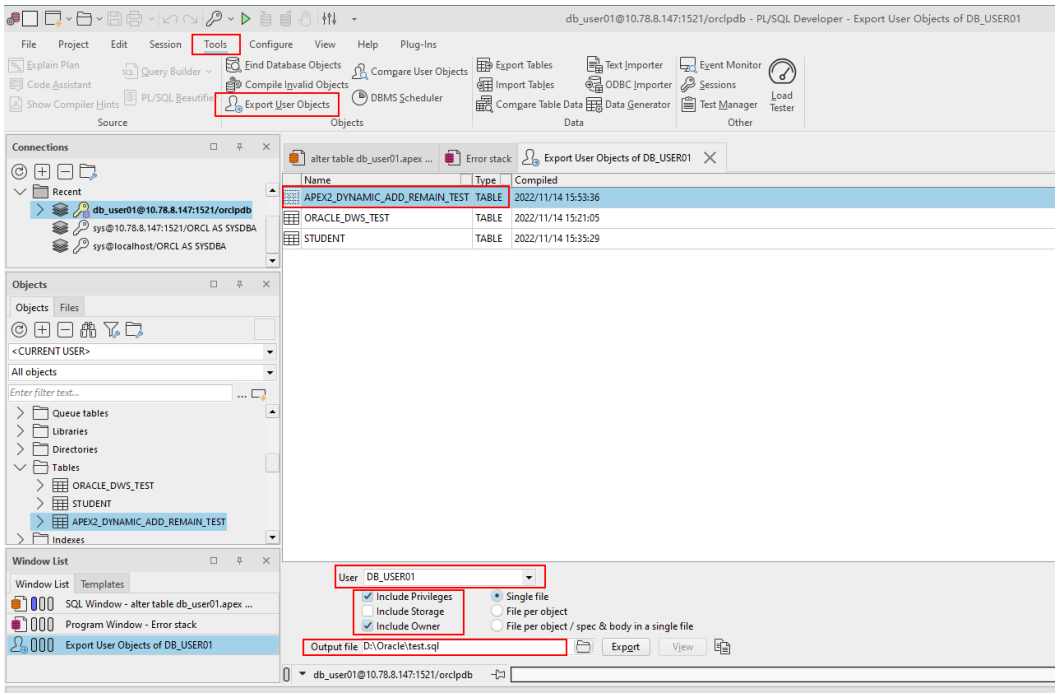
2.1.3.2 Migração de definições de tabela e sintaxe

Passo 1 Efetue logon no PL/SQL Developer usando uma conta com a permissão **sysdba**. Neste exemplo, a conta **db_user01** é usada.

> **NOTA**
>
> A seguir, o PL/SQL Developer Trial Version é usado como exemplo.

Passo 2 Na barra de menus, escolha **Tools** > **Export User Objects**.

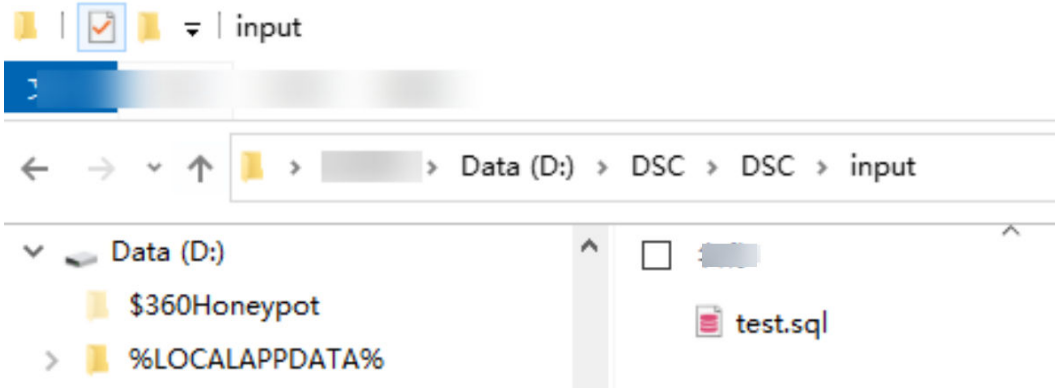
Passo 3 Selecione o usuário conectado **db_user01**, selecione o objeto da tabela **APEX2_DYNAMIC_ADD_REMAIN_TEST** do usuário, selecione o caminho para o arquivo de saída (nomeie o arquivo SQL de saída como **test**) e clique em **Export**.



O arquivo DDL exportado é o seguinte:

```
D: > DSC > DSC > output > output > test.sql
1   prompt PL
2   /
3   SQL Developer Export USER Objects FOR USER DB_USER01@10.78.8.147 :1521 / ORCLPDB \echo Created by 1[illegible] ) on 2
4    /* SET define off; */
5   /*spool test.log*/
6   \echo
7    \echo Creating table APEX2_DYNAMIC_ADD_REMAIN_TEST
8    \echo ========================================
9    \echo
10   CREATE
11       UNLOGGED TABLE
12            DB_USER01.APEX2_DYNAMIC_ADD_REMAIN_TEST (
13                id INTEGER NOT NULL
14                ,TIME DATE
15                ,add_users NUMBER
16                ,remain_users NUMBER
17                ,PRIMARY KEY (ID)
18           ) ;
19  \echo Done
20   /*spool off*/
21  SET define
22       ON ;
```

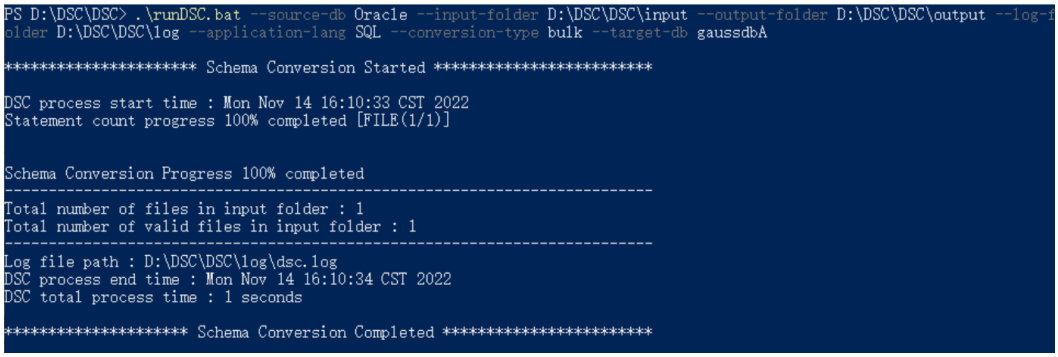
Passo 4 Coloque o arquivo DDL exportado no diretório **input** da pasta DSC descompactada.

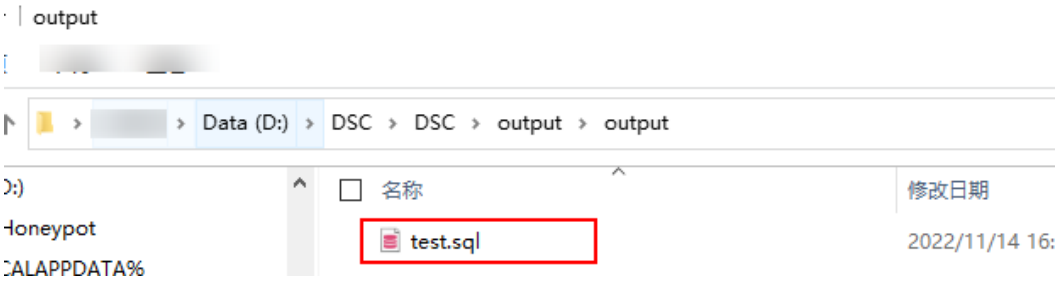


Passo 5 No diretório runDSC.bat, pressione Shift e clique com o botão direito do mouse. Escolha **Open PowerShell window here** e execute a conversão. Substitua **D:\DSC\DSC\input**, **D:\DSC\DSC\output** e **D:\DSC\DSC\log** pelos caminhos de DSC reais.

```
.\runDSC.bat --source-db Oracle --input-folder D:\DSC\DSC\input --output-folder
D:\DSC\DSC\output --log-folder D:\DSC\DSC\log --application-lang SQL --conversion-
type bulk --target-db gaussdbA
```

Passo 6 Após a conclusão da conversão, o arquivo DDL convertido é gerado automaticamente no diretório **output** de DSC.





Passo 7 A estrutura de definição de tabela do GaussDB(DWS) é diferente da do Oracle. Você precisa modificar manualmente a definição da tabela convertida.

Comentar fora **\echo** no arquivo (se você usar gsql para importar definições de tabela, você não precisa fazer isso) e altere manualmente a coluna de distribuição da tabela especificada.

- Antes da mudança:


```
D: > DSC > DSC > output > output > test.sql
prompt PL
/
SQL Developer Export USER Objects FOR USER DB_USER01@10.78.8.147 :1521 / ORCLPDB \echo Created by l[illegible] on [illegible]
 /* SET define off; */
/*spool test.log*/
\echo
 \echo Creating table APEX2_DYNAMIC_ADD_REMAIN_TEST
 \echo ========================================
 \echo
 CREATE
     UNLOGGED TABLE
          DB_USER01.APEX2_DYNAMIC_ADD_REMAIN_TEST (
               id INTEGER NOT NULL
               ,TIME DATE
               ,add_users NUMBER
               ,remain_users NUMBER
               ,PRIMARY KEY (ID)
          ) ;
\echo Done
 /*spool off*/
SET define
     ON ;
```

- Após a mudança:

```
prompt PL
/
SQL Developer Export USER Objects FOR USER DB_USER01@10.78.8.147 :1521 / ORCLPDB \echo Created by [illegible] on
 /* SET define off; */
/*spool test.log*/
 --\echo
 --\echo Creating table APEX2_DYNAMIC_ADD_REMAIN_TEST
 --\echo ========================================
 --\echo
 CREATE
     UNLOGGED TABLE
          DB_USER01.APEX2_DYNAMIC_ADD_REMAIN_TEST (
               id INTEGER NOT NULL
               ,TIME DATE
               ,add_users NUMBER
               ,remain_users NUMBER
               ,PRIMARY KEY (ID)
          ) DISTRIBUTE BY HASH (ID);
\echo Done
 /*spool off*/
SET define
     ON ;
```

NOTA

A coluna de distribuição em uma tabela de hash deve atender aos seguintes requisitos, que são classificados por prioridade em ordem decrescente:

1. Os valores da chave de distribuição devem ser discretos para que os dados possam ser distribuídos uniformemente em cada DN. Você pode selecionar a chave primária da tabela como a chave de distribuição. Por exemplo, para uma tabela de informações da pessoa, escolha a coluna do número de ID como a chave de distribuição.
2. Não selecione a coluna onde existe um filtro constante. Por exemplo, se uma restrição constante (por exemplo, zqdh= '000001') existe na coluna **zqdh** em algumas consultas na tabela **dwcjk**, não é aconselhável usar **zqdh** como a chave de distribuição.
3. Selecione a condição de junção como a coluna de distribuição, para que as tarefas de junção possam ser enviadas para os DNs para serem executadas, reduzindo a quantidade de dados transferidos entre os DNs.

Passo 8 Crie um cluster de GaussDB(DWS). Para obter detalhes, consulte **Criação de um cluster**.

Passo 9 Conecte-se ao cluster do GaussDB(DWS) como o administrador do sistema **dbadmin**. Para obter detalhes, consulte **Uso do cliente da GUI do Data Studio para se conectar a um cluster**. Por padrão, a primeira conexão é com o banco de dados padrão **gaussdb**.

Passo 10 Crie um novo banco de dados de destino **test** e, em seguida, alterne para ele.

```
CREATE DATABASE test WITH ENCODING 'UTF-8' DBCOMPATIBILITY 'ORA' TEMPLATE
template0;
```


Passo 11 Crie um esquema e alterne para ele. O nome do esquema deve ser igual ao nome do usuário de Oracle (**db_user01** neste exemplo).

```
CREATE SCHEMA db_user01;
SET CURRENT_SCHEMA = db_user01;
```

Passo 12 Copie as instruções DDL convertidas em **Passo 7** para Data Studio para execução.

Passo 13 Se a tabela **APEX2_DYNAMIC_ADD_REMAIN_TEST** puder ser encontrada no esquema no banco de dados **test** do cluster GaussDB(DWS), a definição da tabela será migrada.

```
SELECT COUNT(*) FORM db_user01.APEX2_DYNAMIC_ADD_REMAIN_TEST;
```

----Fim

2.1.4 Migração de dados de tabela completa

2.1.4.1 Configuração de uma conexão de fonte de dados do DWS

Passo 1 Crie um cluster e vincule um EIP ao cluster. Para obter detalhes.

AVISO

Certifique-se de que o cluster do CDM e o cluster do GaussDB(DWS) estejam na mesma região e VPC para garantir a conectividade de rede.

Passo 2 Na página **Cluster Management**, clique em **Job Management** na coluna **Operation** do cluster e escolha **Links** > **Create Link**.

Passo 3 Selecione **Data Warehouse Service** e clique em **Next**.

Passo 4 Configure a conexão GaussDB(DWS), clique em **Test**. Se a conexão for bem-sucedida, clique em **Save**.

Tabela 2-3 Informações de conexão de GaussDB(DWS)

Parâmetro	Valor
Name	dws
Database Server	Clique em **Select** e selecione o cluster de GaussDB(DWS) a ser conectado na lista de clusters. **NOTA** O sistema exibe automaticamente os clusters de GaussDB(DWS) na mesma região e VPC. Se nenhum cluster de GaussDB(DWS) estiver disponível, insira manualmente o endereço IP do cluster de GaussDB(DWS) que foi conectado à rede.
Host Port	8000
Database Name	test
User Name	dbadmin
Password	Senha do usuário **dbadmin**

Parâmetro	Valor
Use Agent	No

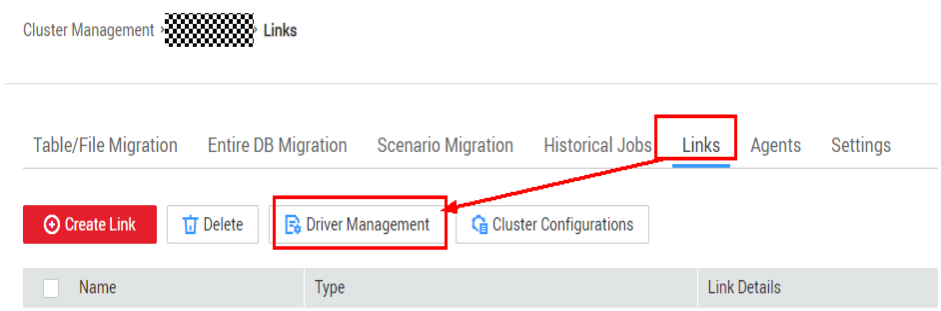
----Fim

2.1.4.2 Configuração de uma conexão de fonte de dados de Oracle

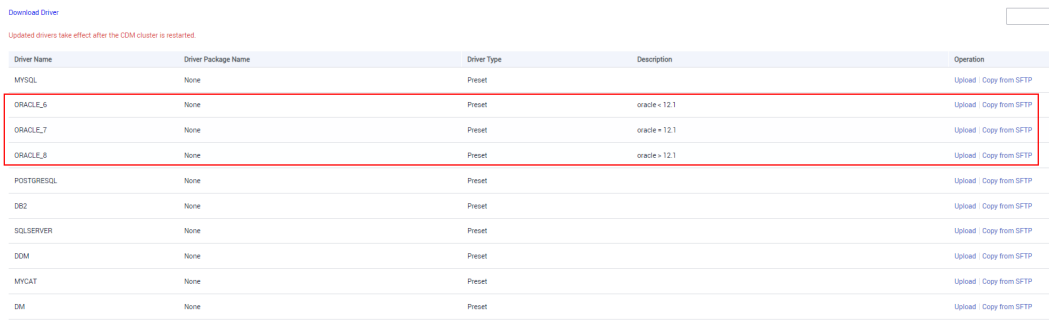
Para migrar dados do Oracle para o GaussDB(DWS), primeiro é necessário configurar uma conexão de fonte de dados Oracle.

Procedimento

Passo 1 Na página **Cluster Management**, clique em **Job Management** na coluna **Operation** do cluster e escolha **Links** > **Driver Management**.



Passo 2 Clique em **Upload** à direita de ORACLE, selecione um pacote de driver de Oracle (se nenhum pacote de driver estiver disponível no PC local, faça o download consultando **Gerenciamento de drivers**) e clique em **Upload**.



Passo 3 Na página **Cluster Management**, clique em **Job Management** na coluna **Operation** do cluster e escolha **Links** > **Create Link**.

Passo 4 Selecione Oracle como o conector e clique em **Next**.

Passo 5 Configure a conexão de Oracle, clique em **Test**. Se a conexão for bem-sucedida, clique em **Save**.

Tabela 2-4 Informações de conexão de Oracle

Parâmetro	Valor
Name	oracle
Database Server	192.168.1.100 (Este é um exemplo. Insira o endereço IP público real do banco de dados de Oracle.)
Host Port	1521
Connection Type	Nome do serviço
Database Name	orcl
User Name	db_user01
Password	-
Use Local API	Não
Use Agent	Não
Oracle Version	Mais tarde do que 12.1

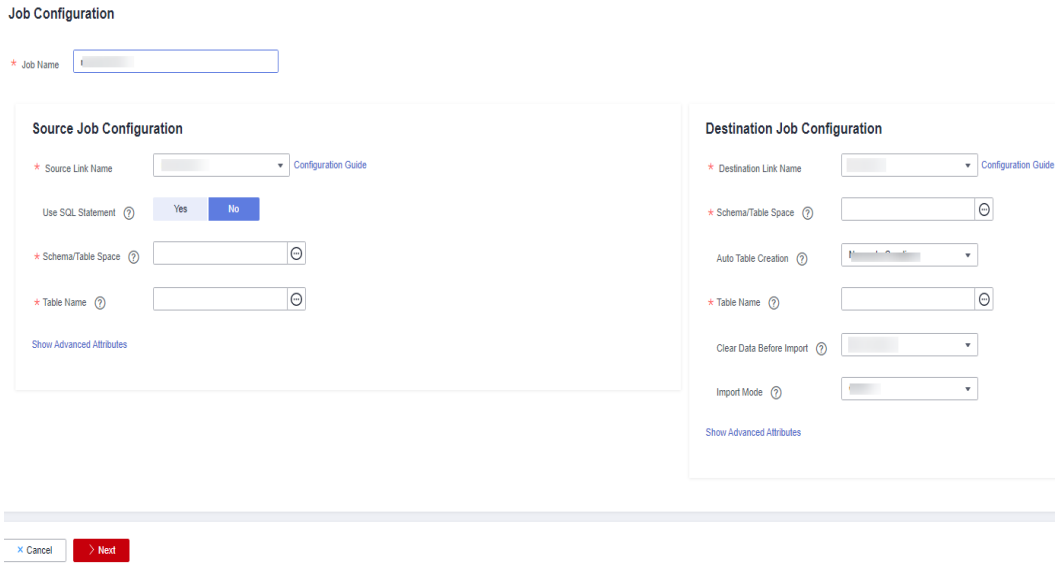
----Fim

2.1.4.3 Migração de tabelas

Procedimento

Passo 1 Na página **Cluster Management**, clique em **Job Management** na coluna **Operation** do cluster e escolha **Table/File Migration** > **Create Job**.

Passo 2 Configure trabalhos na extremidade de origem e na extremidade de destino.



Passo 3 Configure os parâmetros do trabalho de origem com base no tipo do banco de dados de origem.

Tabela 2-5 Parâmetros do trabalho de origem

Parâmetro	Exemplo de valor
Schema/Table Space	db_user01
Use SQL Statement	No
Table Name	APEX2_DYNAMIC_ADD_REMAIN_TEST
WHERE Clause	-
Null in Partition Column	Yes

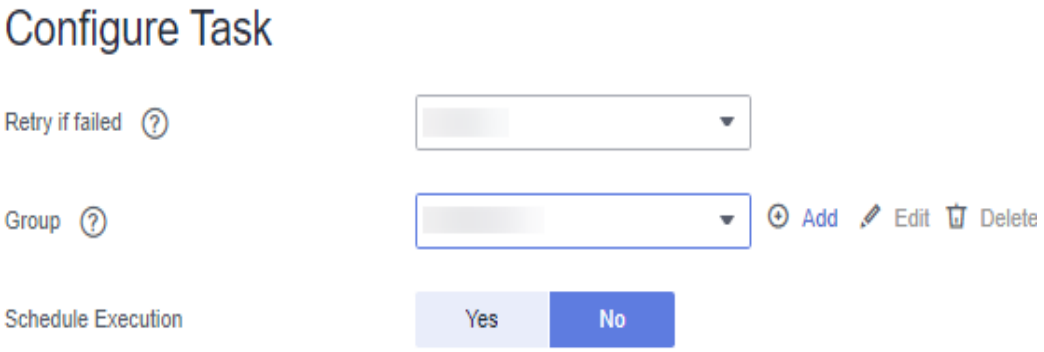
Passo 4 Configure os parâmetros do trabalho de destino com base no serviço de nuvem de destino.

Tabela 2-6 Parâmetros do trabalho de destino

1. Parâmetro	Exemplo de valor
Schema/Table Space	db_user01
Auto Table Creation	Non-auto creation
Table Name	apex2_dynamic_add_remain_test
Clear Data Before Import	Clear all data
Import Mode	COPY
Import to Staging Table	No
Prepare for Data Import	-
Complete Statement After Data Import	analyze db_user01. apex2_dynamic_add_remain_test;

Passo 5 Mapeamento entre campos de origem e campos de destino.

Passo 6 Se a tarefa não for configurada, tente novamente três vezes, salve a configuração e execute a tarefa.



Passo 7 A tarefa é executada e a migração de dados é concluída.

----Fim

2.1.4.4 Verificação

Passo 1 No banco de dados de GaussDB(DWS) **test**, execute a seguinte instrução SQL para consultar o número de linhas na tabela **apex2_dynamic_add_remain_test**. Se o número de linhas for igual ao da tabela de origem, os dados serão consistentes.

```
SELECT COUNT(*) FROM  db_user01.apex2_dynamic_add_remain_test;
```

Passo 2 Execute a seguinte instrução para verificar a assimetria de dados:

Se a assimetria dos dados estiver dentro de 10%, a distribuição dos dados é normal. A migração de dados foi concluída.

```
SELECT TABLE_SKEWNESS('db_user01.apex2_dynamic_add_remain_test');
```

	table_skewness
1	("dn_6001_6002 ",97,32.119%)
2	("dn_6003_6004 ",105,34.768%)
3	("dn_6005_6006 ",100,33.113%)

----Fim

2.1.5 Migração de instruções SQL

2.1.5.1 Migração de sintaxe

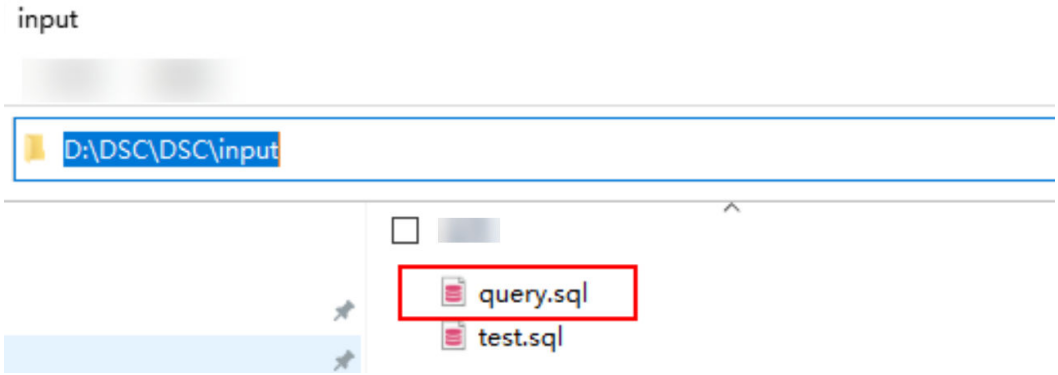
Passo 1 Salve as seguintes instruções SQL em um banco de dados Oracle como um arquivo query.sql.

```
-- Generally, the HAVING clause must appear after the GROUP BY clause, but Oracle
allows HAVING to appear before or after the GROUP BY clause. Therefore, you need
to move the HAVING clause after the GROUP BY clause in the target database.
SELECT
id,
count(*),
sum(remain_users)
FROM LYC.APEX2_DYNAMIC_ADD_REMAIN_TEST
HAVING id <= 5
GROUP BY id;

UNIQUE keywords are migrated as DISTINCT keywords.
SELECT UNIQUE add_users FROM LYC.APEX2_DYNAMIC_ADD_REMAIN_TEST;
```



```
-- In NVL2(expression,value1,value2), if the expression is not Null, NVL2 returns
Value1. If the expression is Null, NVL2 returns Value2.
SELECT NVL2(add_users, 1, 2) FROM LYC.APEX2_DYNAMIC_ADD_REMAIN_TEST SHERE rownum
<= 2;
```

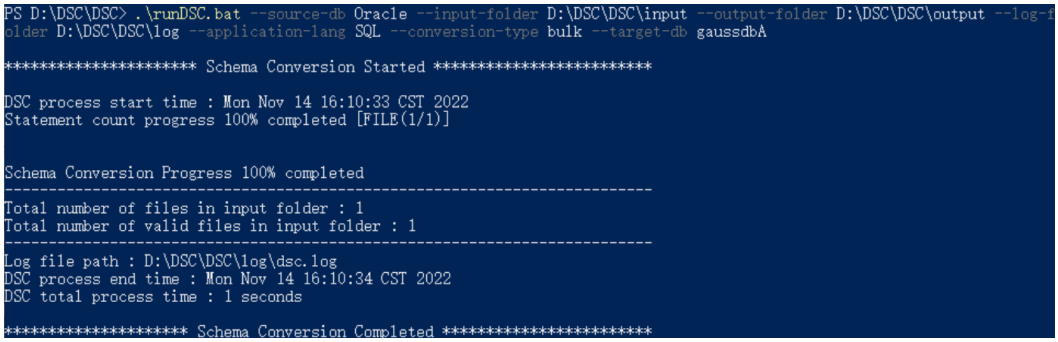
Passo 2 Coloque o arquivo query.sql obtido em **Passo 1** no diretório **input** da pasta de DSC descompactada.



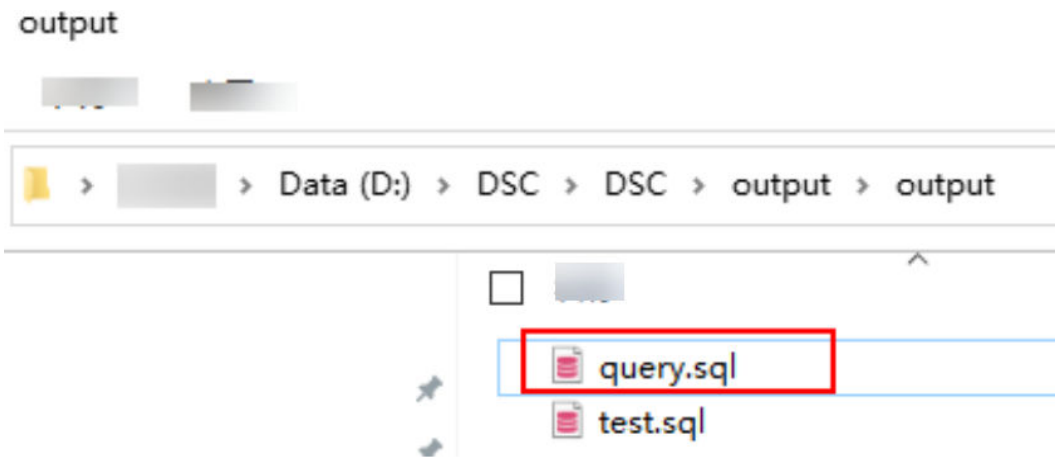
Passo 3 No diretório runDSC.bat, pressione Shift e clique com o botão direito do mouse. Escolha **Open PowerShell window here** e execute a conversão.

Substitua **D:\DSC\DSC\input**, **D:\DSC\DSC\output** e **D:\DSC\DSC\log** pelos caminhos de DSC reais.

```
.\runDSC.bat --source-db Oracle --input-folder D:\DSC\DSC\input --output-folder
D:\DSC\DSC\output --log-folder D:\DSC\DSC\log --application-lang SQL --conversion-
type bulk --target-db gaussdbA
```



Passo 4 Após a conclusão da conversão, um arquivo DML é gerado no diretório de saída.



----Fim

2.1.5.2 Verificação

Passo 1 Execute as instruções SQL no banco de dados do Oracle antes da migração.

Passo 2 Execute as instruções SQL migradas no Data Studio.

Passo 3 Compare os resultados da execução. Se eles forem os mesmos, a migração do SQL está completa.

----Fim

2.2 Sincronização de dados da tabela de MySQL para GaussDB(DWS) em tempo real

Esta prática demonstra como usar o Data Replication Service (DRS) para sincronizar dados do MySQL com o GaussDB(DWS) em tempo real. Para obter detalhes sobre DRS, consulte **O que é o DRS?**

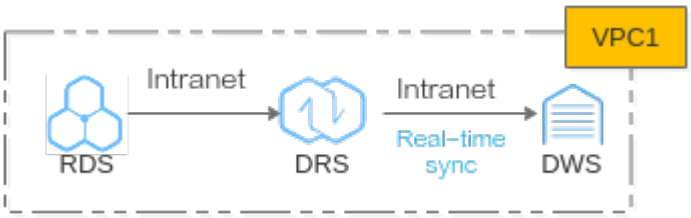
Essa prática leva cerca de 60 minutos. O procedimento é os seguintes:

1. **Preparativos**
2. **Passo 1: preparar uma tabela de origem MySQL**
3. **Passo 2: criar um cluster do GaussDB(DWS).**
4. **Passo 3: criar uma tarefa de sincronização do DRS**
5. **Passo 4: verificar sincronização de dados**

Descrição do cenário

Em cenários de análise de Big Data, o MySQL serve como um banco de dados OLTP. Depois que o MySQL é conectado ao armazém de dados do GaussDB(DWS) para análise OLAP, os dados escritos pelo MySQL em tempo real precisam ser sincronizados com o armazém de dados do GaussDB(DWS) em tempo real. DRS é usado para executar a sincronização.

Figura 2-3 Sincronização em tempo real do DRS



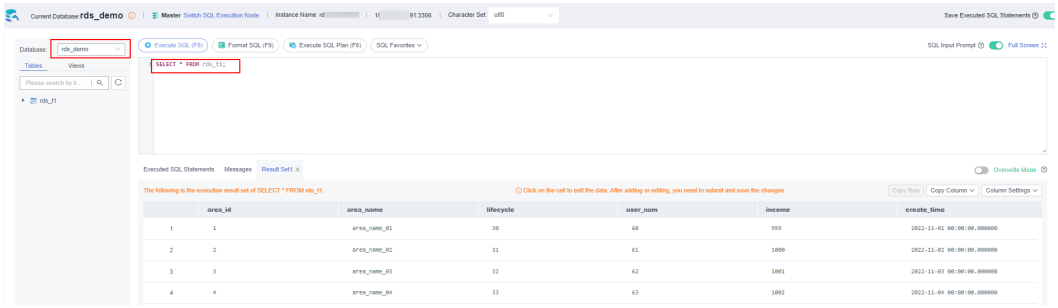
Preparativos

- Você registrou uma conta da Huawei e ativou os serviços da Huawei Cloud.. Antes de usar o GaussDB(DWS), verifique o status da conta. A conta não pode estar em atraso ou congelada.
- A tabela de origem do MySQL a ser migrada foi preparada. Nessa prática, um banco de dados MySQL do RDS da Huawei Cloud é usado como dados de origem. Se o banco de dados MySQL estiver off-line, verifique se a conexão de rede está normal.

Passo 1: preparar uma tabela de origem MySQL

Passo 1 Você adquiriu um mecanismo de banco de dados MySQL do RDS (nesta prática, use o MySQL 8.0.x como exemplo). Para obter detalhes, consulte **Compra de uma instância de BD**.

Passo 2 O banco de dados de origem **rds_demo** com o conjunto de caracteres **utf8mb4** foi criado e há a tabela **rds_t1** com dados no banco de dados.



----Fim

Passo 2: criar um cluster do GaussDB(DWS).

Passo 1 **Criação de um cluster**. Para garantir a conectividade de rede, o cluster do GaussDB(DWS) e o RDS devem estar na mesma região.

Passo 2 Na página **Clusters** do console do GaussDB(DWS), localize a linha que contém o cluster de destino e clique em **Login** na coluna **Operation**.

NOTA

Esta prática usa a versão 8.1.3.x como exemplo. 8.1.2 e versões anteriores não suportam este modo de logon. Você pode usar o Data Studio para se conectar a um cluster. Para obter detalhes, consulte **Uso do Data Studio para se conectar a um cluster**.

Passo 3 Após efetuar logon no banco de dados do GaussDB(DWS), crie o banco de dados **rds_demo** para sincronização.


```
CREATE DATABASE rds_demo WITH ENCODING 'UTF-8' DBCOMPATIBILITY 'mysql' TEMPLATE
template0;
```

Passo 4 Alterne para o banco de dados **rds_demo** e crie um esquema chamado **rds_demo**.

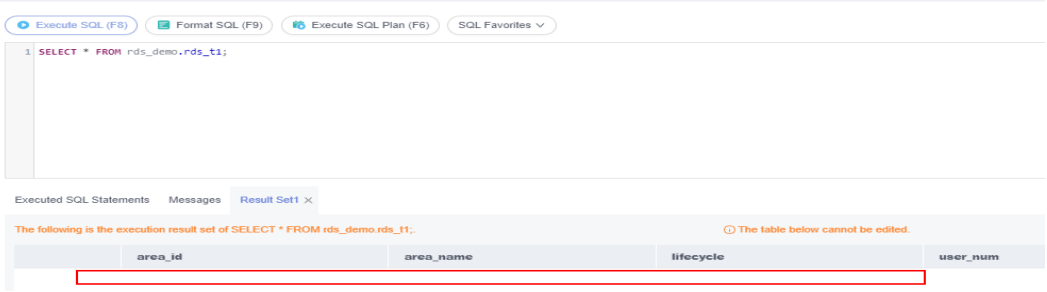
```
CREATE SCHEMA rds_demo;
```

Passo 5 Crie uma tabela chamada **rds_t1** no esquema **rds_demo**.

```
CREATE TABLE rds_demo.rds_t1 (
  area_id varchar(256) NOT NULL,
  area_name varchar(256) DEFAULT NULL,
  lifecycle varchar(256) DEFAULT NULL,
  user_num int DEFAULT NULL,
  income  bigint DEFAULT NULL,
  create_time timestamp DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (area_id)
)distribute by hash(area_id);
COMMENT on column rds_demo.rds_t1.area_id is 'Region Code';
COMMENT on column rds_demo.rds_t1.area_name is 'Region Name';
COMMENT on column rds_demo.rds_t1.lifecycle is 'Life Cycle';
 COMMENT on column rds_demo.rds_t1.user_num is 'Subscribers in Each Life Cycle';
 COMMENT on column rds_demo.rds_t1.income is 'Region Income';
COMMENT on column rds_demo.rds_t1.create_time is 'Creation Time';
```

Passo 6 Consulte dados da tabela. Atualmente, a tabela está vazia.

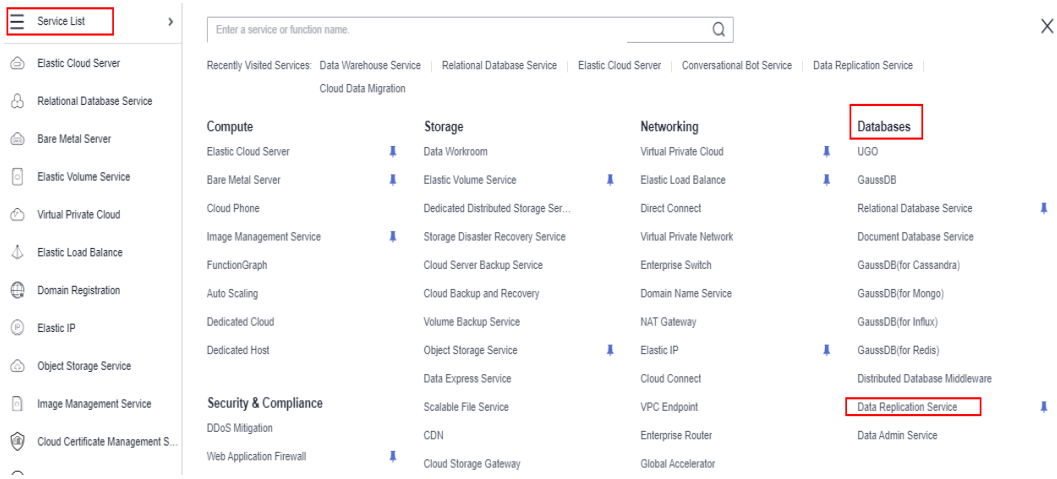
```
SELECT * FROM rds_demo.rds_t1;
```



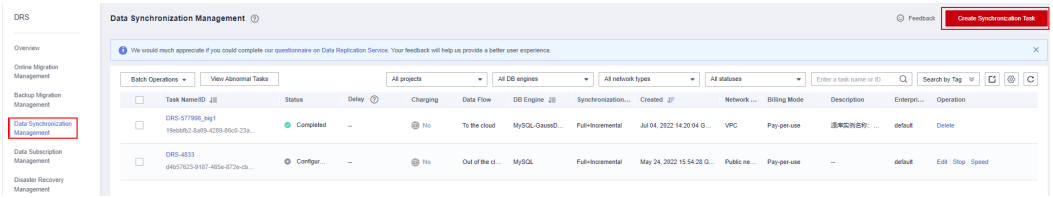
----Fim

Passo 3: criar uma tarefa de sincronização do DRS

Passo 1 Escolha **Service List** > **Databases** > **Data Replication Service** para alternar para o console do DRS.



Passo 2 Escolha **Data Synchronization Management** à esquerda e clique em **Create Synchronization Task** no canto superior direito.



Passo 3 Configure parâmetros básicos. Para mais detalhes, consulte **Tabela 2-7**.

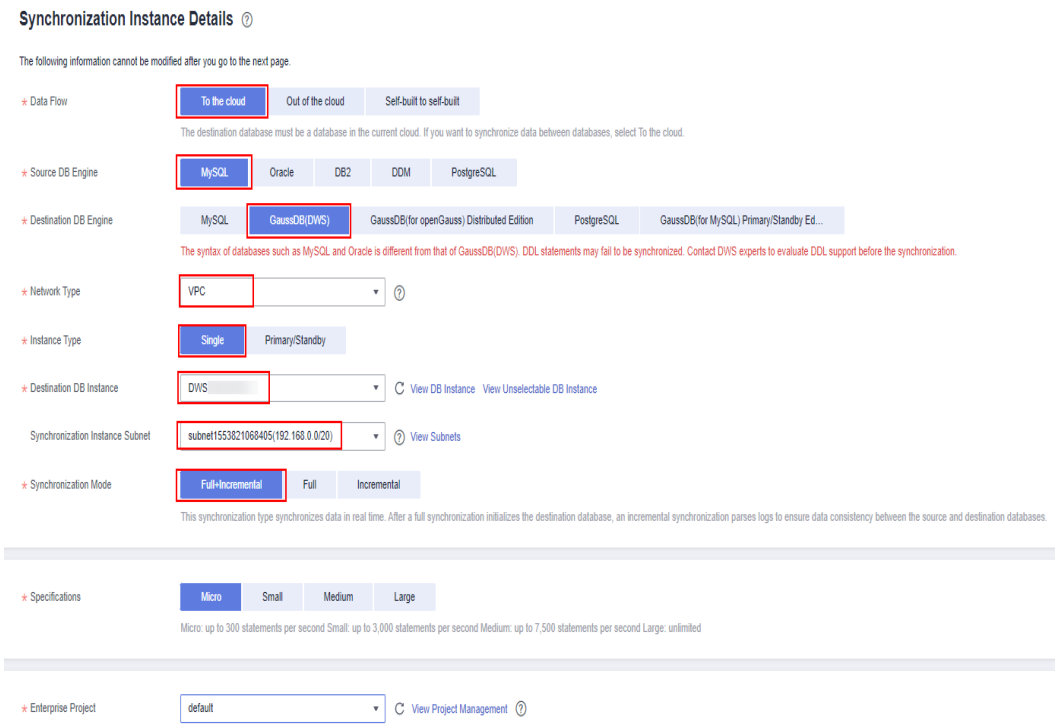
Tabela 2-7 Parâmetros básicos

Parâmetro	Valor
Billing Mode	Pay-per-use
Region	CN-Hong Kong. Verifique se o RDS e o GaussDB(DWS) estão na mesma região.
Project	CN-Hong Kong
Task Name	DRS-DWS
Description	-

Passo 4 Configure os seguintes parâmetros. Para mais detalhes, consulte **Tabela 2-8**.

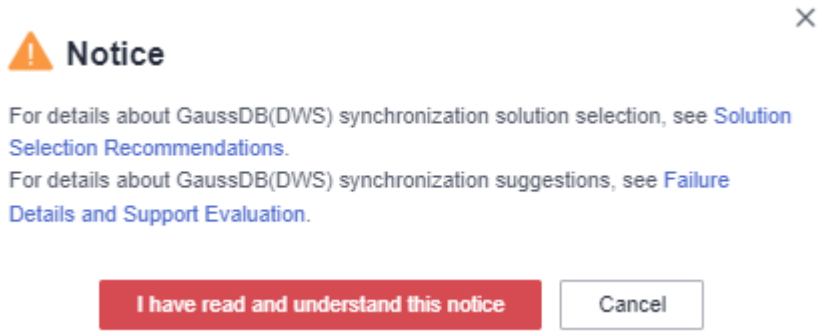
Tabela 2-8 Parâmetros de instância sincronizados

Parâmetro	Valor
Data Flow	To the cloud
Source DB Engine	MySQL
Destination DB engine	GaussDB(DWS)
Network Type	Nesta prática, selecione **VPC**. Se o banco de dados MySQL estiver off-line, selecione **Public Network**.
Instance Type	Single
Destination DB Instance	Selecione o cluster criado em **Passo 2: criar um cluster do GaussDB(DWS).**.
Synchronization Instance Subnet	Selecione a sub-rede em que o cluster do GaussDB(DWS) reside. Nessa prática, o RDS e o GaussDB(DWS) estão na mesma VPC e sub-rede.
Synchronous Mode	Full+Incremental
Specifications	Nesta prática, selecione **Micro**. Essa opção é selecionada com base no volume de dados e na taxa de sincronização.



Passo 5 Clique em **Next** e clique em **I have read and understand this notice**.

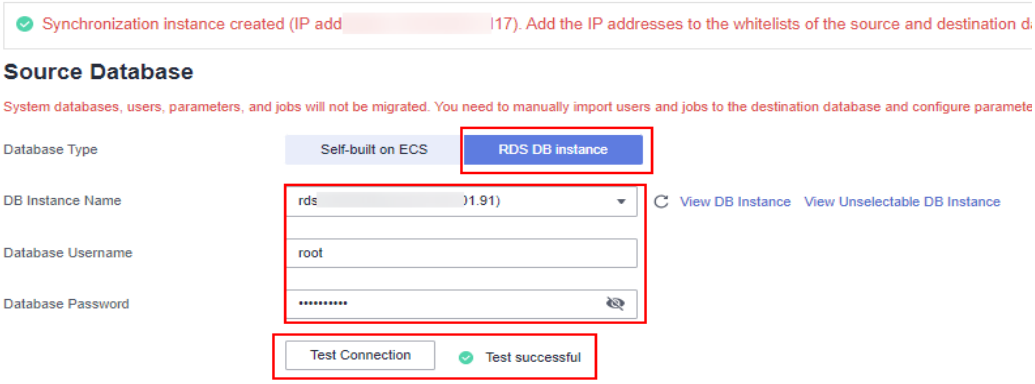
Aguarde cerca de 5 a 10 minutos para que a sincronização seja concluída.



Passo 6 Depois que a sincronização for bem-sucedida, insira as informações do banco de dados de origem e clique em **Test Connection**.

Tabela 2-9 Informações do banco de dados de origem

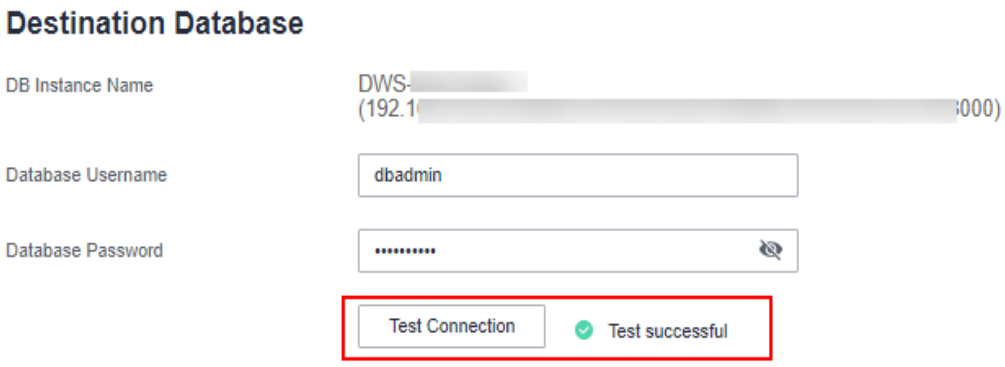
Parâmetro	Valor
Tipo de banco de dados	RDS DB Instance
DB Instance Name	Selecione a instância de banco de dados do RDS criada.
Database Username	root
Database Password	****



Passo 7 Insira as informações do banco de dados de destino e clique em **Test Connection**. O teste de conexão foi bem-sucedido.

Tabela 2-10 Informações do banco de dados de destino

Parâmetro	Valor
Database Username	dbadmin
Database Password	****



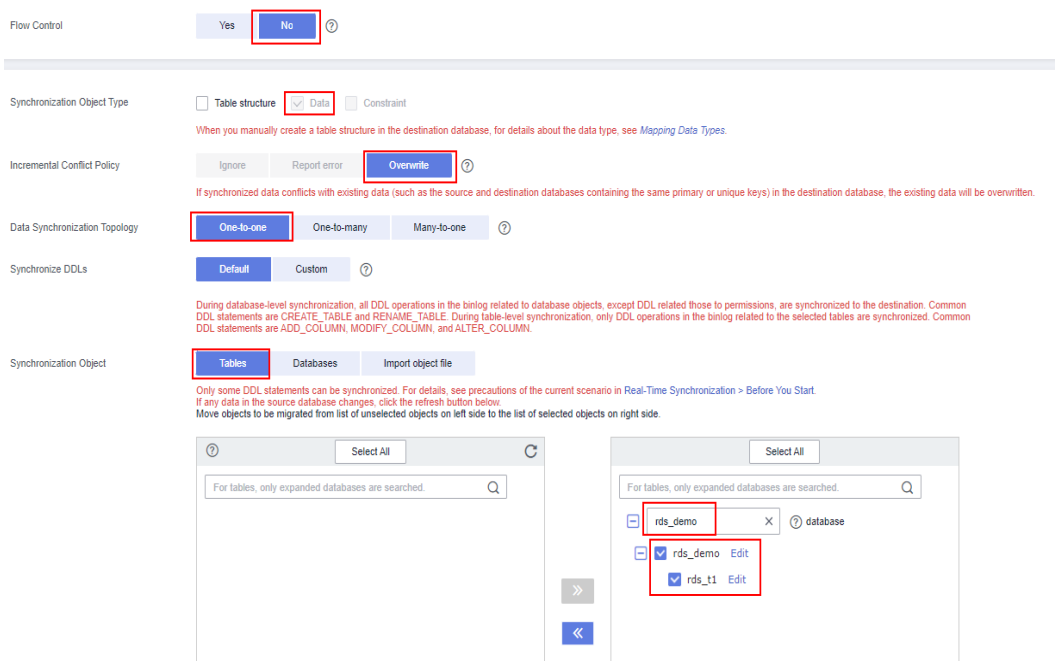
Passo 8 Clique em **Next** e, em seguida, clique em **Agree**.

Passo 9 Defina a política de sincronização. Para mais detalhes, consulte **Tabela 2-11**.

Tabela 2-11 Política de sincronização

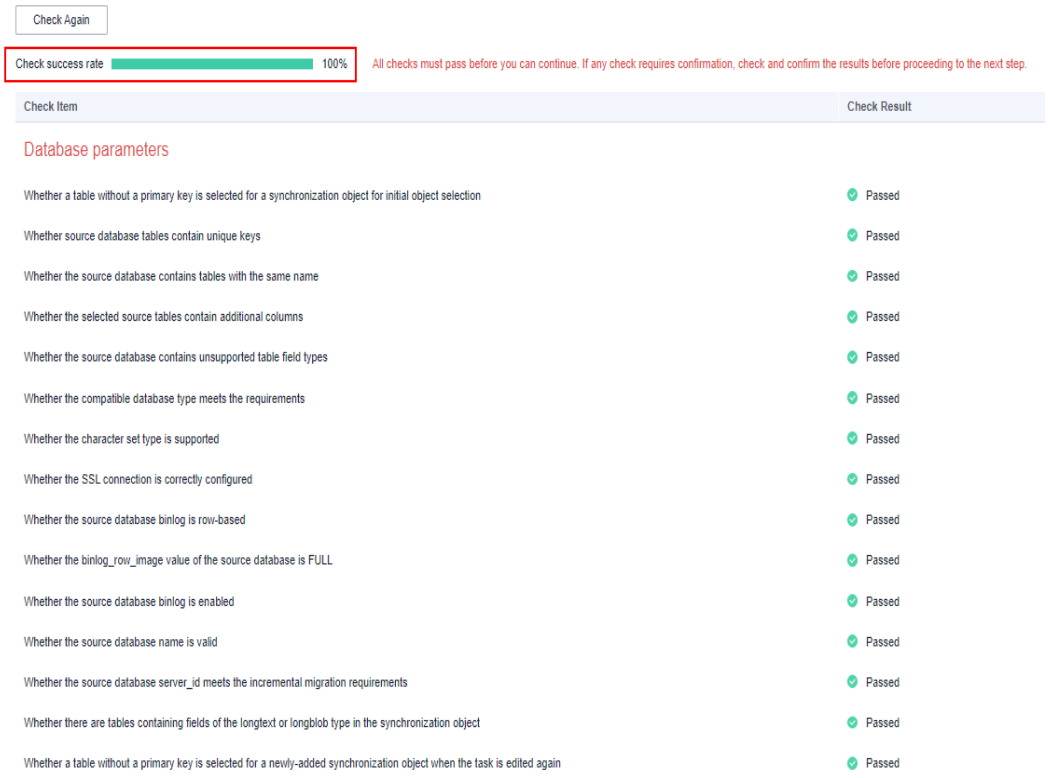
Parâmetro	Valor
Flow Control	No

Parâmetro	Valor
Synchronization Object Type	Data
Incremental Conflict Policy	Overwrite
Data Synchronization Topology	Um-para-um
Synchronize DDLs	Default
Synchronization Object	Tables Selecione a tabela a ser sincronizada do banco de dados de origem. Nesta prática, selecione **rds_t1** em **rds_demo**. Digite o nome do banco de dados do GaussDB(DWS) para o qual os dados são sincronizados: **rds_demo**

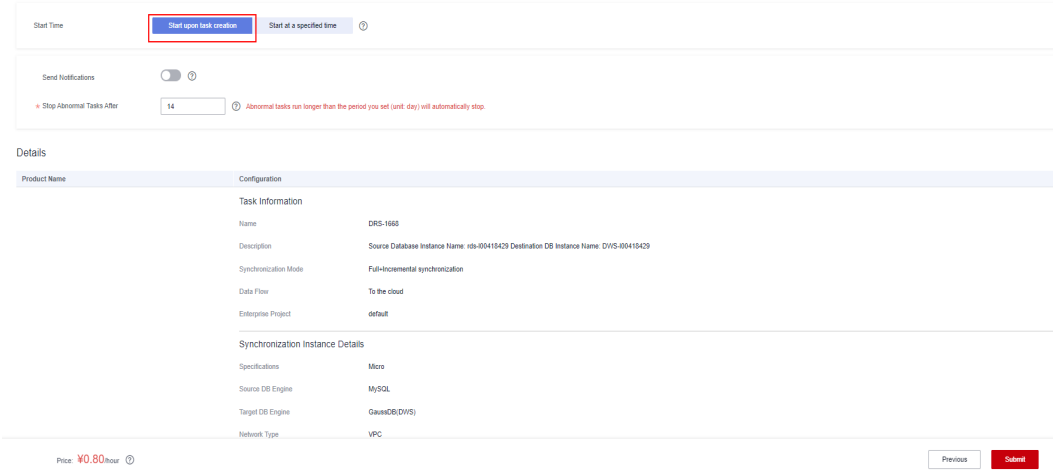


Passo 10 Clique em **Next**, confirme as informações e clique em **Next**.

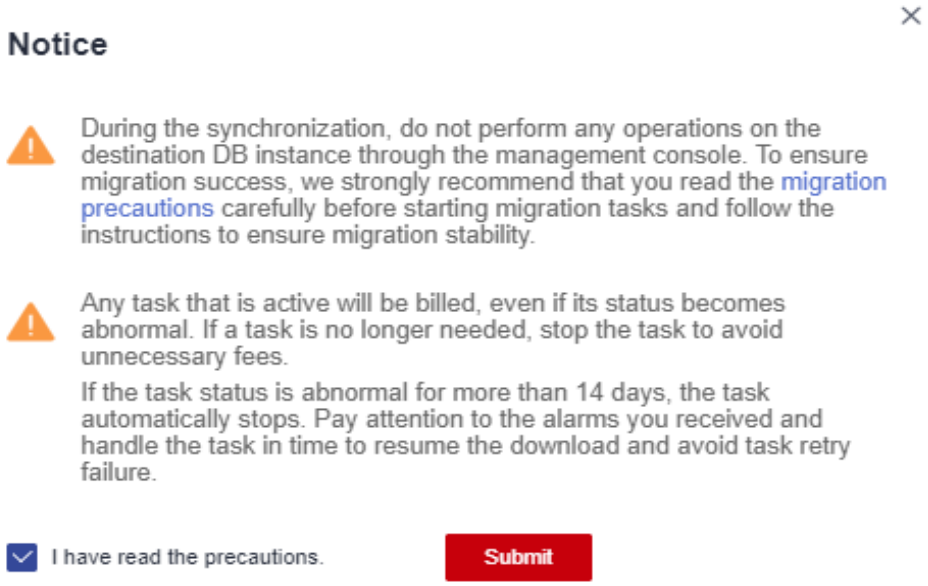
Aguarde até que a verificação do parâmetro do banco de dados seja bem-sucedida. Se a verificação falhar, clique em **Check Again**.



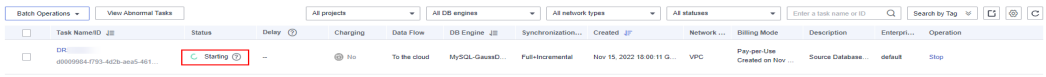
Passo 11 Clique em **Next**, selecione **Start upon task creation**, verifique outras informações e clique em **Submit** no canto inferior direito.



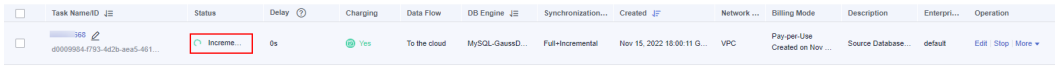
Passo 12 Na caixa de diálogo exibida, confirme as informações, selecione **I have read and understand this notice** e clique em **Start Task**.



Volte para a página **Data Synchronization Management** e aguarde cerca de 5 a 10 minutos. A sincronização foi iniciada com êxito.



Aguarde cerca de 5 minutos e continue **Passo 4: verificar sincronização de dados**.

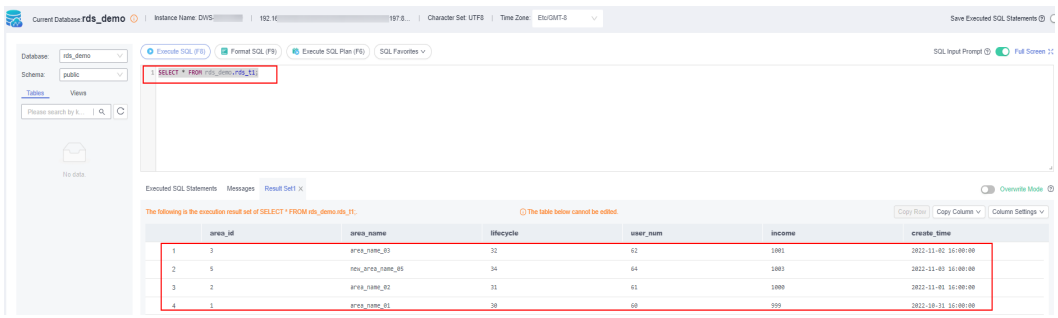


----Fim

Passo 4: verificar sincronização de dados

Passo 1 Efetue logon no console do GaussDB(DWS) novamente e execute a instrução a seguir para consultar os dados da tabela novamente. Se o resultado for mostrado da seguinte forma, a sincronização completa de dados é bem-sucedida.

```
SELECT * FROM rds_demo.rds_t1;
```



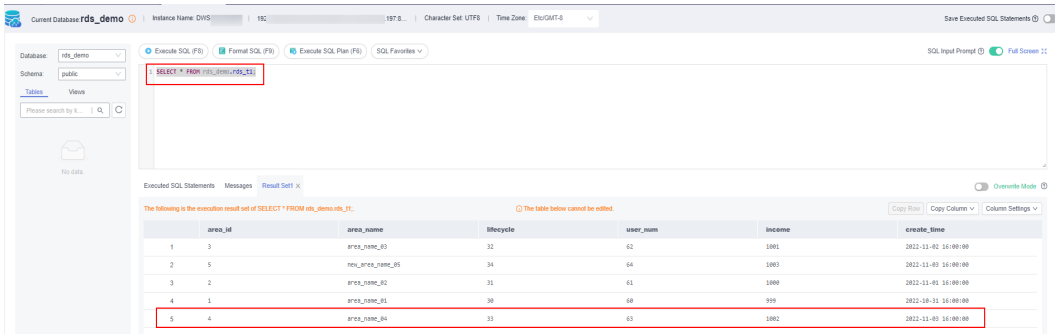
Passo 2 Alterne para o console do RDS, efetue logon no banco de dados do RDS e insira novos dados na tabela **rds_t1**.

```
INSERT INTO rds_t1 VALUES ('5','new_area_name_05',34,64,1003,'2022-11-04');
```


Passo 3 Alterne de volta para o banco de dados do GaussDB(DWS) e execute a seguinte instrução para consultar dados da tabela:

Uma linha de dados é adicionada ao resultado da consulta, indicando que os dados no banco de dados MySQL foram sincronizados com o GaussDB(DWS) em tempo real.

```
SELECT * FROM rds_demo.rds_t1;
```



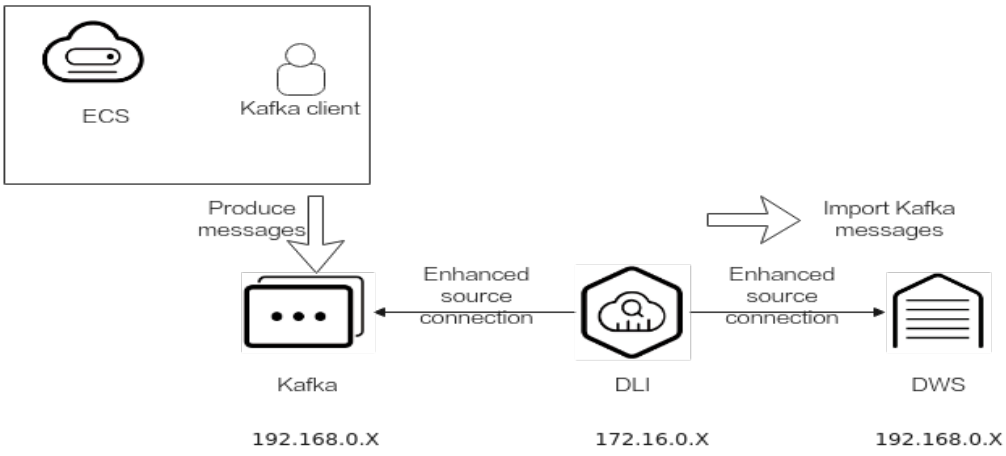
----Fim

2.3 Uso de trabalhos de Flink de DLI para gravar dados do Kafka para GaussDB(DWS) em tempo real

Esta prática demonstra como usar trabalhos do Flink de DLI para sincronizar dados de consumo do Kafka para o GaussDB(DWS) em tempo real. O processo de demonstração inclui gravar e atualizar os dados existentes em tempo real.

- Para obter detalhes, consulte **O que é o Data Lake Insight?**
- Para obter detalhes sobre o Kafka, consulte **O que é o DMS for Kafka?**

Figura 2-4 Importação de dados do Kafka para o GaussDB(DWS) em tempo real



Essa prática leva cerca de 90 minutos. Os serviços de nuvem usados nessa prática incluem **Virtual Private Cloud (VPC) e sub-redes**, **Elastic Load Balance (ELB)**, **Elastic Cloud Server (ECS)**, **Object Storage Service (OBS)**, **Distributed Message Service (DMS) for Kafka**, **Data Lake Insight (DLI)** e **Data Warehouse Service (DWS)**. O processo básico é o seguinte:

1. **Preparações**
2. **Passo 1: criar uma instância do Kafka**
3. **Passo 2: criar um cluster do GaussDB(DWS) e uma tabela de destino**
4. **Passo 3: criar uma fila de DLI**
5. **Passo 4: criar uma conexão de origem de dados avançada para Kafka e GaussDB(DWS)**
6. **Passo 5: preparar a ferramenta dws-connector-flink para interconectar o GaussDB(DWS) com o Flink**
7. **Passo 6: criar e editar um trabalho do Flink de DLI**
8. **Passo 7: criar e modificar mensagens no cliente do Kafka**

Descrição do cenário

Suponha que os dados de exemplo da fonte de dados de Kafka é uma tabela de informações do usuário, como mostrado em **Tabela 2-12**, que contém os campos **id**, **name** e **age**. O campo **id** é único e fixo, que é compartilhado por vários sistemas de serviço. Geralmente, o campo **id** não precisa ser modificado. Somente os campos **name** e **age** precisam ser modificados.

Use o Kafka para gerar os três grupos de dados a seguir e use os trabalhos de Flink de DLI para sincronizar os dados com o GaussDB(DWS): Altere os usuários cujos IDs são **2** e **3** para **jim** e **tom** e use os trabalhos de Flink de DLI para atualizar dados e sincronizar os dados com o GaussDB(DWS).

Tabela 2-12 Dados de amostra

id	name	age
1	lily	16
2	lucy > jim	17
3	lilei > tom	15

Restrições

- Certifique-se de que VPC, ECS, OBS, Kafka, DLI e GaussDB(DWS) estejam na mesma região, por exemplo, China-Hong Kong.
- Certifique-se de que Kafka, DLI e GaussDB(DWS) possam se comunicar uns com os outros. Nesta prática, Kafka e GaussDB(DWS) são criados na mesma região e VPC, e os grupos de segurança de Kafka e GaussDB(DWS) permitem o segmento de rede das filas de DLI.
- Para garantir que a ligação entre DLI e DWS é estável, vincule o serviço ELB para o cluster de armazém de dados criado.

Preparativos

- Você registrou uma conta da Huawei e ativou os serviços da Huawei Cloud.. Antes de usar o GaussDB(DWS), verifique o status da conta. A conta não pode estar em atraso ou congelada.

- Você criou uma VPC e uma sub-rede. Para obter detalhes, consulte **Criação de uma VPC**.

Passo 1: criar uma instância do Kafka

Passo 1 Faça logon no console de gerenciamento do Huawei Cloud e escolha **Middleware** > **Distributed Message Service (for Kafka)** na lista de serviços. O console de gerenciamento do Kafka é exibido.

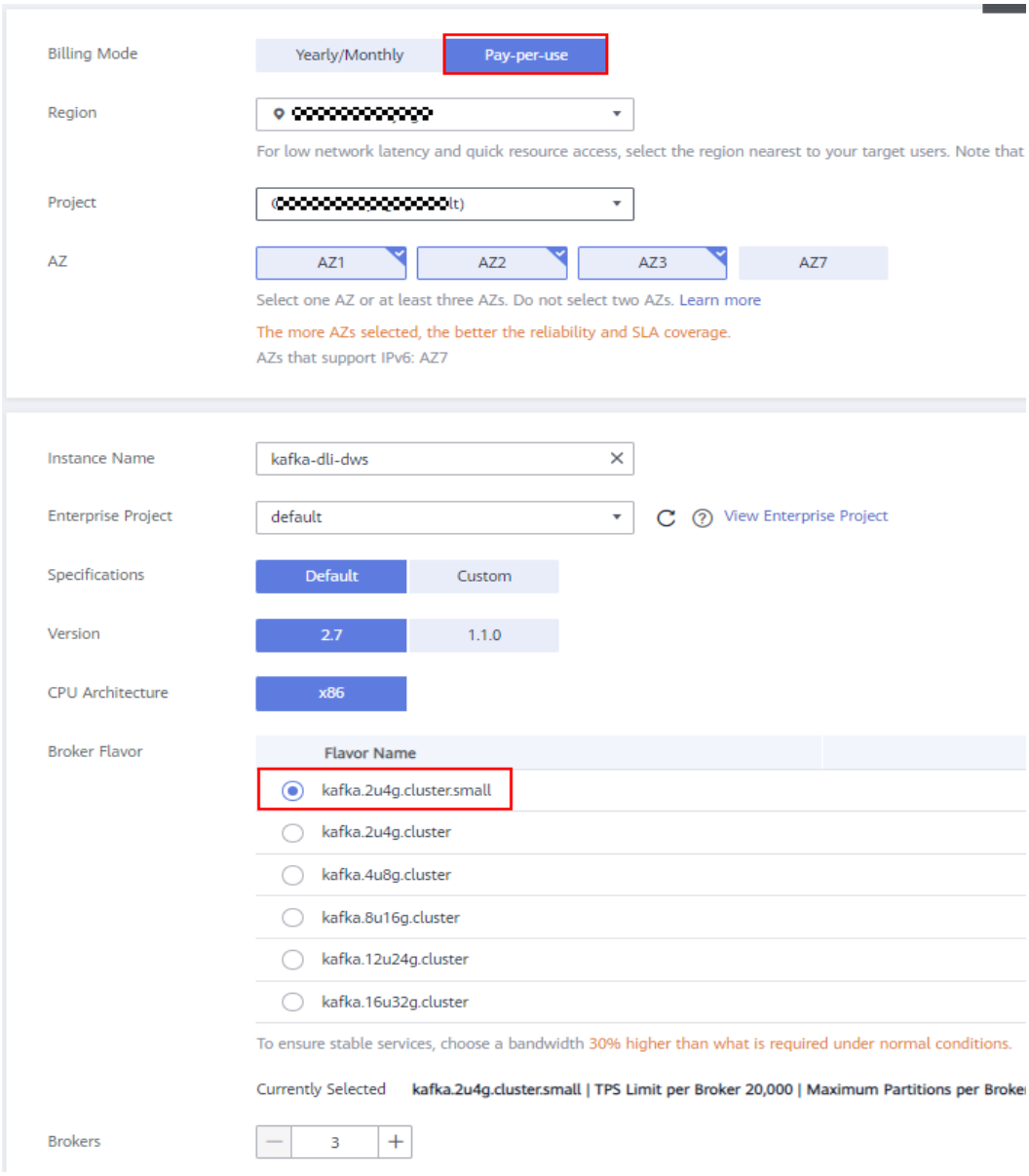
Passo 2 Clique em **DMS for Kafka** à esquerda e clique em **Buy Instance** no canto superior direito.

Passo 3 Defina os seguintes parâmetros. Retém os valores padrão para outros parâmetros que não estão descritos na tabela.

Tabela 2-13 Parâmetros de instância do Kafka

Parâmetro	Valor
Billing Mode	Pay-per-use
Region	CN-Hong Kong
Project	Default
AZ	AZ 1 (Se não estiver disponível, selecione outra AZ.)
Instance Name	kafka-dli-dws
Enterprise Project	default
Specifications	Default
Version	2.7
CPU Architecture	x86
Broker Flavor	kafka.2u4g.cluster.small (apenas para referência. Selecione o menor flavor.)
Brokers	3
VPC	Selecione uma VPC criada. Se nenhuma VPC estiver disponível, crie uma.
Security Group	Selecione um grupo de segurança criado. Se nenhum grupo de segurança estiver disponível, crie um.
Other parameters	Mantenha o valor padrão.

Figura 2-5 Criar uma instância do Kafka



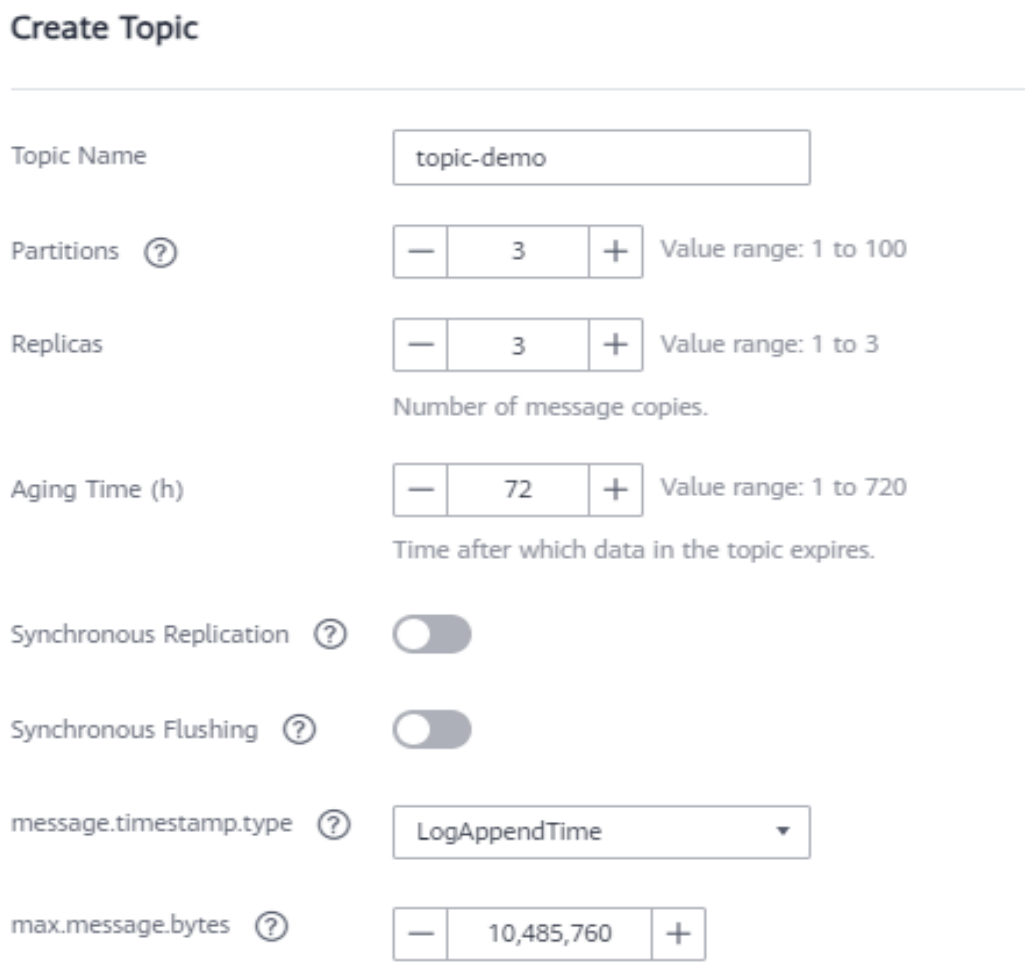
Passo 4 Clique em **Buy** e conclua o pagamento. Espere até que a criação seja bem sucedida.

Passo 5 Na lista de instâncias do Kafka, clique no nome da instância criada do Kafka. A página **Basic Information** é exibida.

Passo 6 Escolha **Topics** à esquerda e clique em **Create Topic**.

Defina **Topic Name** como **topic-demo** e mantenha os valores padrão para outros parâmetros.

Figura 2-6 Criação de um tópico



Passo 7 Clique em **OK**. Na lista de tópicos, você pode ver que o topic-demo foi criado com sucesso.

Passo 8 Escolha **Consumer Groups** à esquerda e clique em **Create Consumer Group**.

Passo 9 Insira **kafka01** para **Consumer Group Name** e clique em **OK**.

----Fim

Passo 2: criar um cluster do GaussDB(DWS) e uma tabela de destino

Passo 1 **Crie um balanceador de carga dedicado**, defina **Network Type** como **IPv4 private network**. Defina Region e VPC com os mesmos valores da instância do Kafka. Neste exemplo, defina Region como China-Hong Kong.

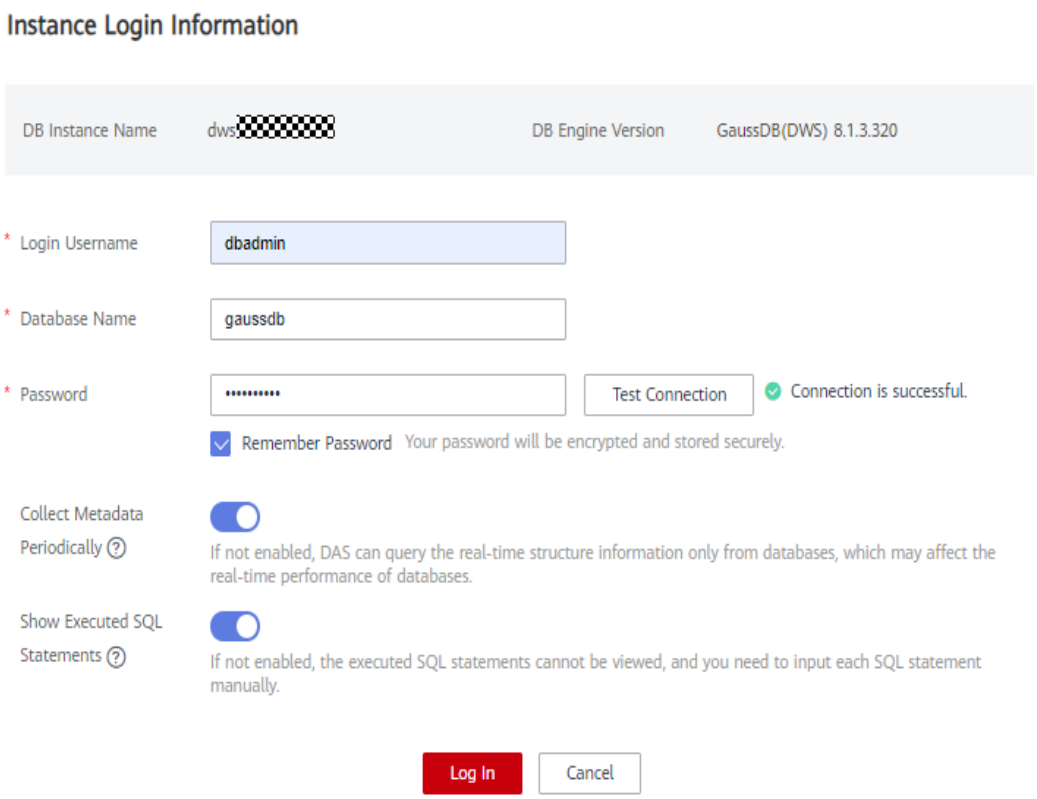
Passo 2 **Criação de um cluster**. Para garantir a conectividade de rede, a região e a VPC do cluster de GaussDB(DWS) devem ser as mesmas da instância do Kafka. Nesta prática, a região e a VPC são China-Hong Kong.. A VPC deve ser a mesma que a criada para o Kafka.

Passo 3 Na página **Clusters** do console do GaussDB(DWS), localize a linha que contém o cluster de destino e clique em **Login** na coluna **Operation**.

> **NOTA**
>
> Esta prática usa a versão 8.1.3.x como exemplo. 8.1.2 e versões anteriores não suportam este modo de logon. Você pode usar o Data Studio para se conectar a um cluster. Para obter detalhes, consulte **Uso do Data Studio para se conectar a um cluster**.

Passo 4 O nome de usuário de logon é **dbadmin**, o nome do banco de dados é **gaussdb** e a senha é a senha do usuário **dbadmin** definida durante a criação do cluster do armazém de dados. Selecione **Remember Password**, ative **Collect Metadata Periodically** e **Show Executed SQL Statements** e clique em **Log In**.

Figura 2-7 Fazer login no GaussDB(DWS)



Passo 5 Clique no nome do banco de dados **gaussdb** e clique em **SQL Window** no canto superior direito para acessar o editor SQL.

Passo 6 Copie a seguinte instrução SQL. Na janela SQL, clique em Execute SQL para criar a tabela de destino **user_dws**.

```
CREATE TABLE user_dws (
id int,
name varchar(50),
age int,
PRIMARY KEY (id)
);
```

----Fim

Passo 3: criar uma fila de DLI

Passo 1 Faça logon no console de gerenciamento da Huawei Cloud e escolha **Analytics** > **Data Lake Insight** na lista de serviços. O console de gerenciamento do DLI é exibido.

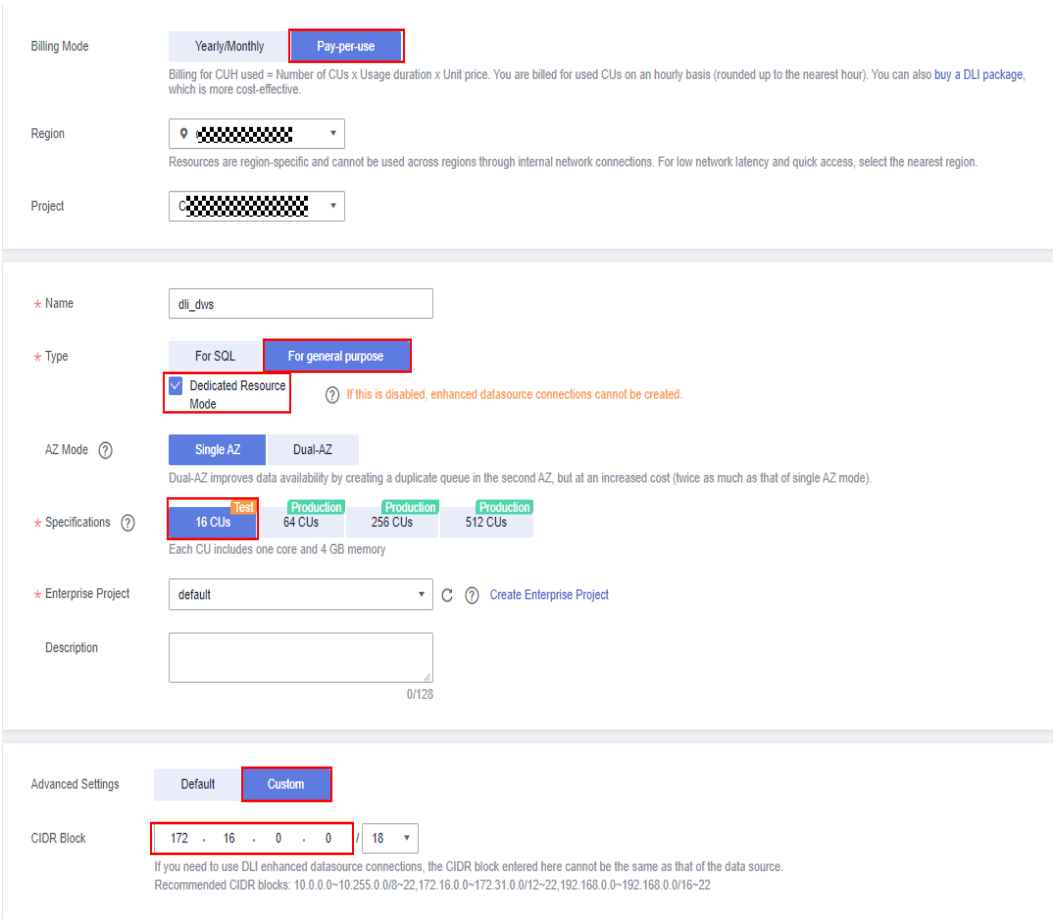
Passo 2 No painel de navegação à esquerda, escolha Resource Management > Queue Manager.

Passo 3 Clique em **Buy Queue** no canto superior direito, defina os seguintes parâmetros e mantenha os valores padrão para outros parâmetros que não estão descritos na tabela.

Tabela 2-14 Parâmetros da fila de DLI

Parâmetro	Valor
Billing Mode	Pay-per-use
Region	CN-Hong Kong
Project	Default
Name	dli_dws
Type	Para uma fila geral, selecione **Dedicated Resource Mode**.
AZ Mode	Single-AZ deployment
Specifications	16 CUs
Enterprise Project	default
Advanced Settings	Custom
CIDR Block	172.16.0.0/18. Ele deve estar em um segmento de rede diferente do Kafka e do GaussDB(DWS). Por exemplo, se Kafka e GaussDB(DWS) estiverem no segmento de rede 192.168.x.x, selecione 172.16.x.x para DLI.

Figura 2-8 Criar uma fila do DLI



Passo 4 Clique em **Buy**.

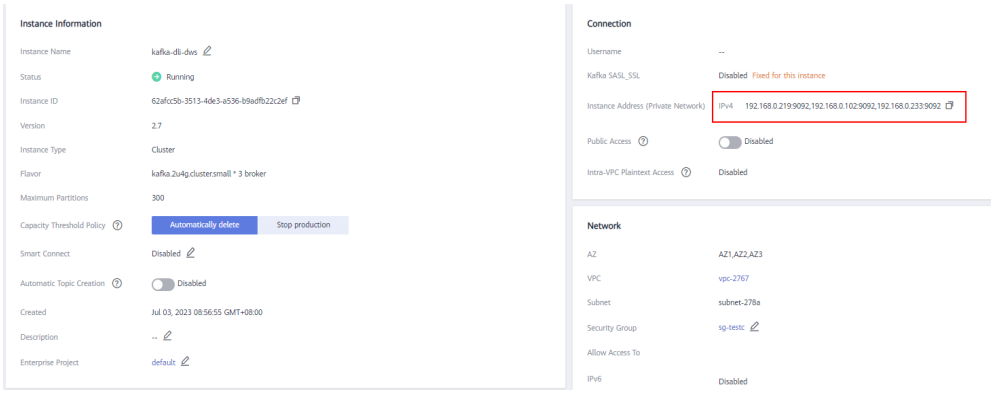
----Fim

Passo 4: criar uma conexão de origem de dados avançada para Kafka e GaussDB(DWS)

Passo 1 No grupo de segurança do Kafka, permita o segmento de rede onde a fila do DLI está localizada.

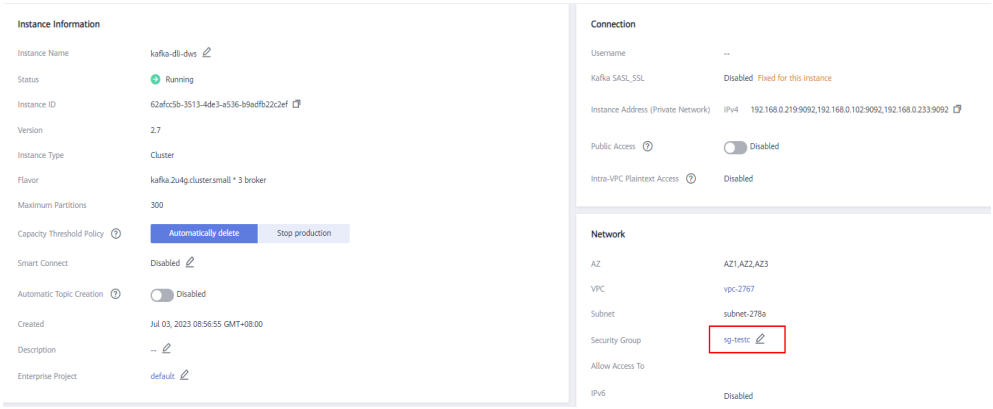
1. Retorne ao console do Kafka e clique no nome da instância do Kafka para acessar a página **Basic Information**. Visualize o valor de **Instance Address (Private Network)** nas informações de conexão e registre o endereço para uso futuro.

Figura 2-9 Endereço de rede privada do Kafka



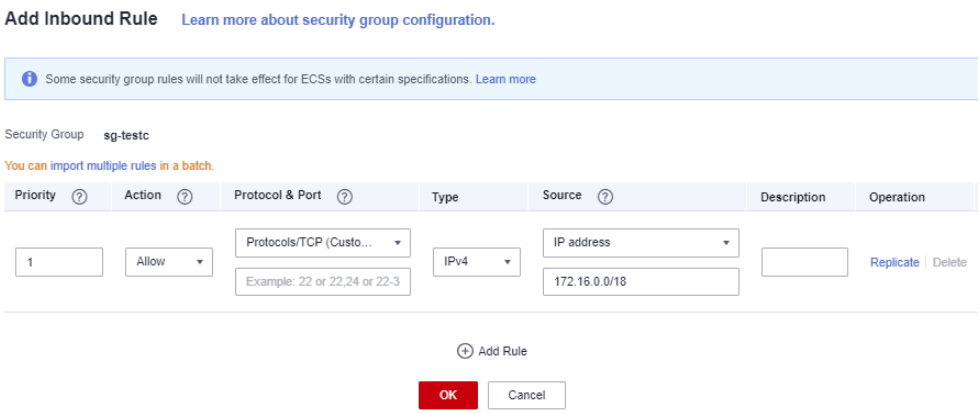
2. Clique no nome do grupo de segurança.

Figura 2-10 Grupo de segurança do Kafka



3. Escolha **Inbound Rules** > **Add Rule**, conforme mostrado na figura a seguir. Adicione o segmento de rede da fila do DLI. Neste exemplo, o segmento de rede é **172.16.0.0/18**. Assegure-se de que o segmento de rede seja o mesmo que aquele entrado durante **Passo 3: criar uma fila de DLI**.

Figura 2-11 Adicionar regras ao grupo de segurança do Kafka



4. Clique em **OK**.

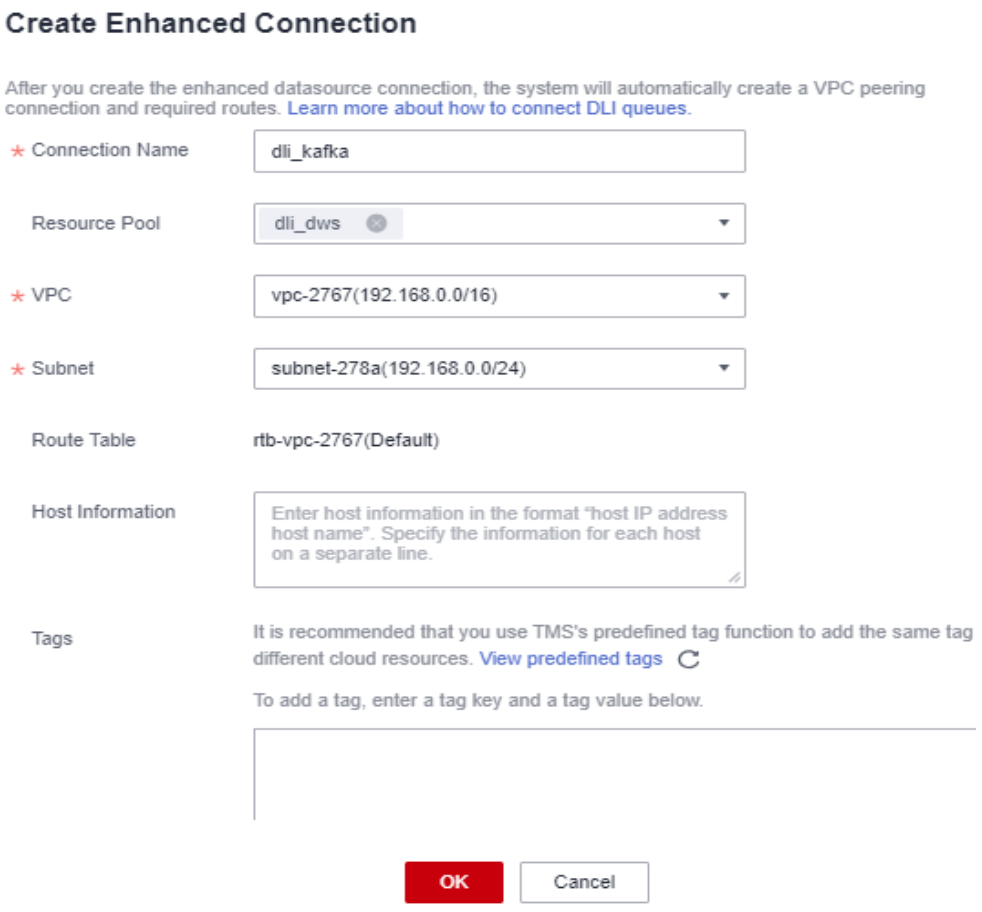
Passo 2 Retorne ao console de gerenciamento do DLI, clique em **Datasource Connections** à esquerda, selecione **Enhanced** e clique em **Create**.

Passo 3 Defina os seguintes parâmetros. Retém os valores padrão para outros parâmetros que não estão descritos na tabela.

Tabela 2-15 Conexão de DLI para Kafka

Parâmetro	Valor
Connection Name	dli_kafka
Resource Pool	Selecione a fila do DLI criada **dli_dws**.
VPC	Selecione a VPC do Kafka.
Subnet	Selecione a sub-rede onde o Kafka está localizado.
Other parameters	Mantenha o valor padrão.

Figura 2-12 Criar uma conexão aprimorada

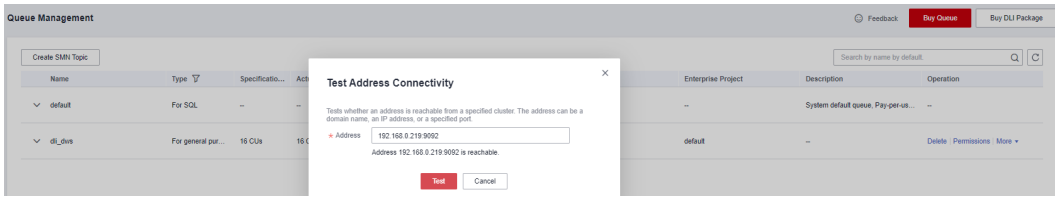


Passo 4 Clique em **OK**. Aguarde até que a conexão do Kafka seja criada com êxito.

Passo 5 Escolha **Resources** > **Queue Management** à esquerda e escolha **More** > **Test Address Connectivity** à direita de **dli_dws**.

Passo 6 Na caixa endereço, digite o endereço IP privado e o número da porta da instância do Kafka obtida em **Passo 1.1**. (Há três endereços de Kafka. Digite apenas um deles.)

Figura 2-13 Testar a conectividade do Kafka

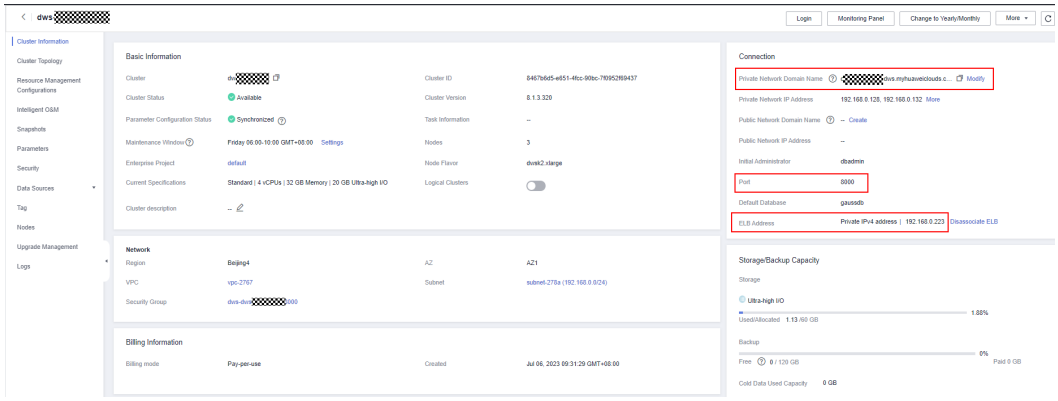


Passo 7 Clique em **Test** para verificar se o DLI está conectado com êxito ao Kafka.

Passo 8 Faça logon no console de gerenciamento do GaussDB(DWS), escolha **Clusters** à esquerda e clique no nome do cluster para ir para a página de detalhes.

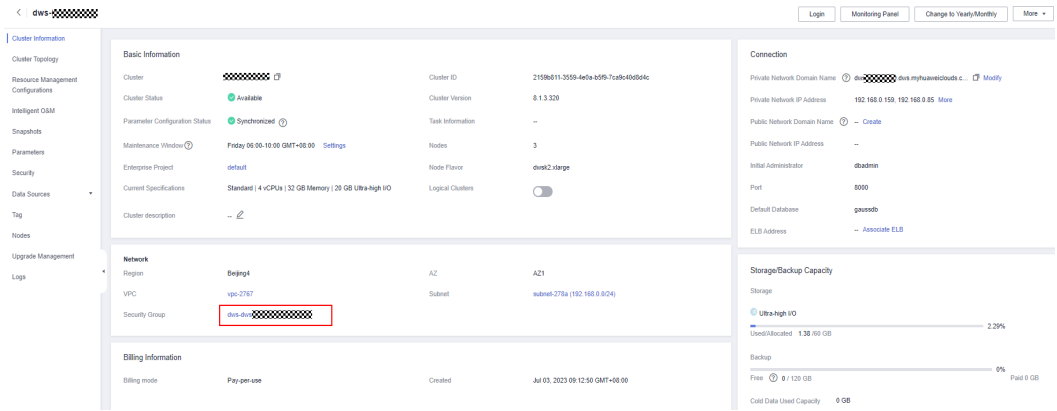
Passo 9 Registre o nome do domínio da rede privada, o número da porta e o endereço do Elastic Load Balance do cluster do armazém de dados para uso futuro.

Figura 2-14 Nome de domínio privado e endereço do ELB



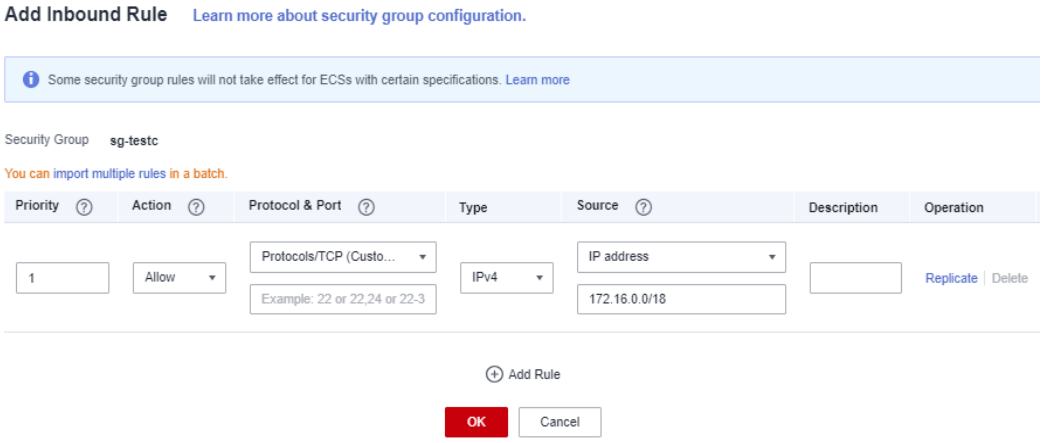
Passo 10 Clique no nome do grupo de segurança.

Figura 2-15 Grupo de segurança do GaussDB(DWS)



Passo 11 Escolha **Inbound Rules** > **Add Rule**, conforme mostrado na figura a seguir. Adicione o segmento de rede da fila do DLI. Neste exemplo, o segmento de rede é **172.16.0.0/18**. Assegure-se de que o segmento de rede seja o mesmo que aquele entrado durante **Passo 3: criar uma fila de DLI**.

Figura 2-16 Adicionar uma regra ao grupo de segurança do GaussDB(DWS)

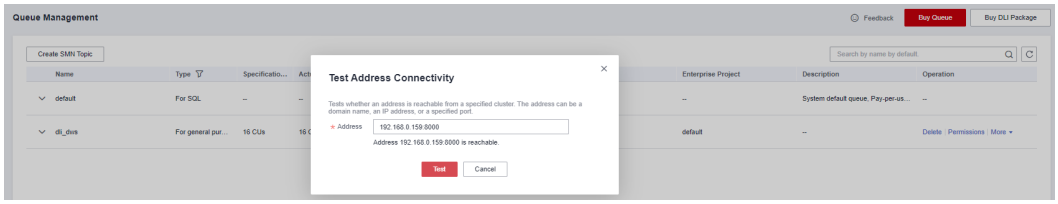


Passo 12 Clique em **OK**.

Passo 13 Alterne ao console do DLI, escolha **Resources** > **Queue Management** à esquerda e clique **More** > **Test Address Connectivity** à direita de **dli_dws**.

Passo 14 Na caixa endereço, digite o endereço IP do Elastic Load Balance e o número da porta do cluster do GaussDB(DWS) obtido em **Passo 9**.

Figura 2-17 Testar a conectividade do GaussDB(DWS)



Passo 15 Clique em **Test** para verificar se o DLI está conectado com êxito ao GaussDB(DWS).

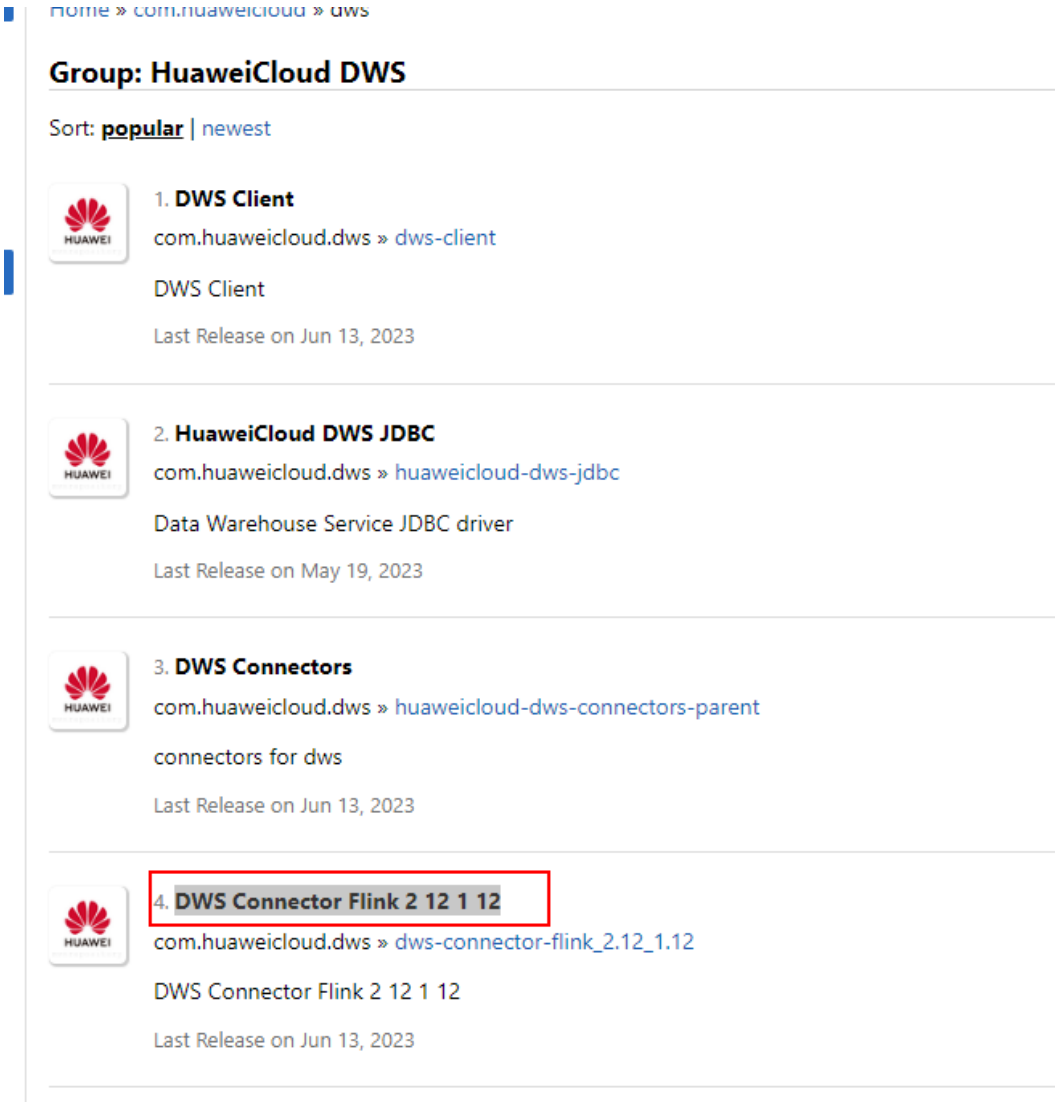
----Fim

Passo 5: preparar a ferramenta dws-connector-flink para interconectar o GaussDB(DWS) com o Flink

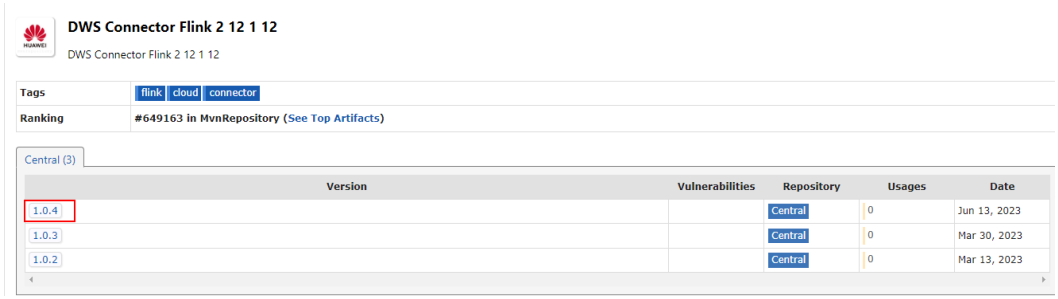
dws-connector-flink é uma ferramenta para interconexão com Flink baseada em APIs de JDBC do DWS. Durante a configuração do trabalho do DLI, essa ferramenta e suas dependências são armazenadas no diretório de carregamento da classe do Flink para melhorar a capacidade de importar trabalhos do Flink para GaussDB(DWS).

Passo 1 Acesse **https://mvnrepository.com/artifact/com.huaweicloud.dws** em um navegador.

Passo 2 Na lista de softwares, selecione a versão mais recente do GaussDB(DWS) Connectors Flink. Nesta prática, selecione **DWS Connector Flink 2 12 1 12**.

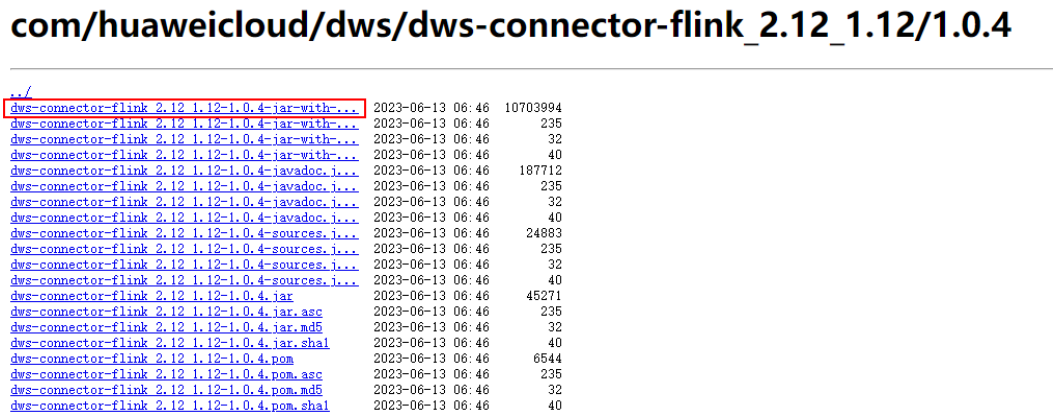


Passo 3 Clique na ramificação 1.0.4. (Clique na ramificação mais recente em cenários reais).



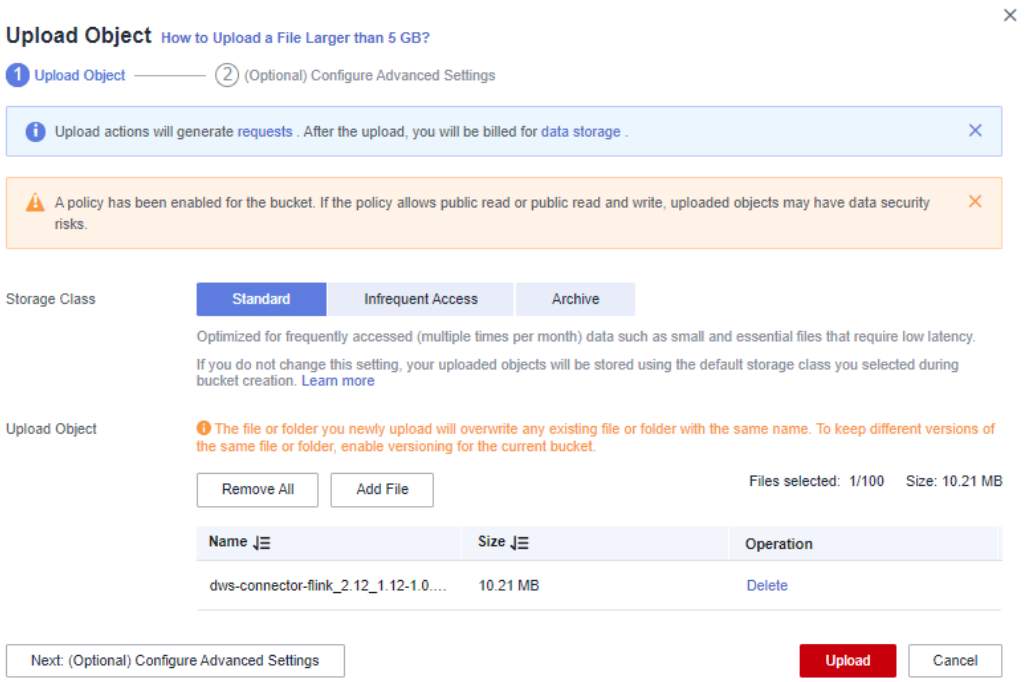
Passo 4 Clique em **View All**.

Passo 5 Clique em **dws-connector-flink_2.12_1.12-1.0.4-jar-with-dependencies.jar** para fazer o download para o host local.



Passo 6 Crie um bucket do OBS. Nesta prática, defina o nome do bucket como **obs-flink-dws** e faça upload do arquivo para o bucket do OBS. Certifique-se de que o bucket esteja na mesma região que o DLI, que nesta prática é China-Hong Kong.

Figura 2-18 Fazer upload do pacote JAR para o bucket do OBS



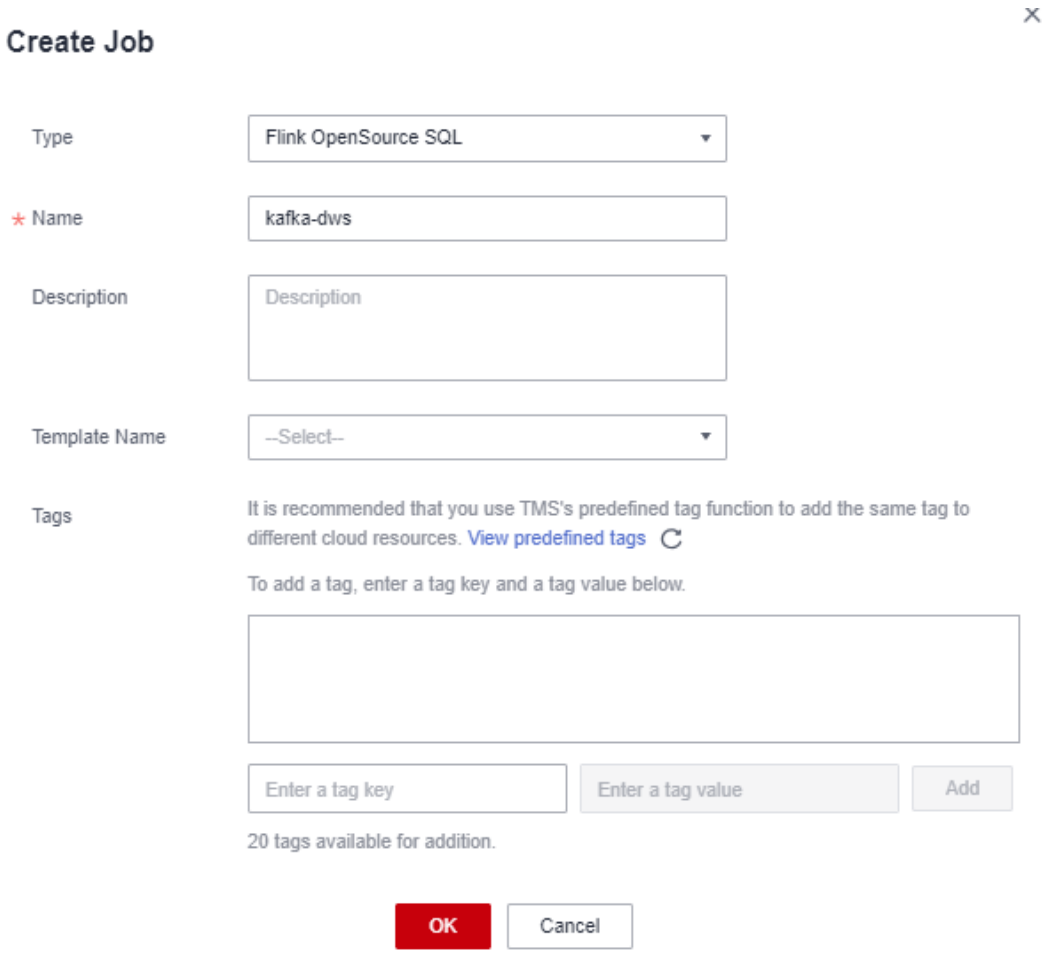
----Fim

Passo 6: criar e editar um trabalho do Flink de DLI

Passo 1 Retorne ao console de gerenciamento do DLI, escolha **Job Management** > **Flink Jobs** à esquerda e clique em **Create Job** no canto superior direito.

Passo 2 Defina **Type** para **Flink OpenSource SQL** e **Name** para **kafka-dws**.

Figura 2-19 Criare um trabalho



Passo 3 Clique em **OK**. A página para edição do trabalho é exibida.

Passo 4 Defina os seguintes parâmetros à direita da página. Retém os valores padrão para outros parâmetros que não estão descritos na tabela.

Tabela 2-16 Parâmetros do trabalho do Flink

Parâmetro	Valor
Queue	dli_dws
Flink Version	1.12

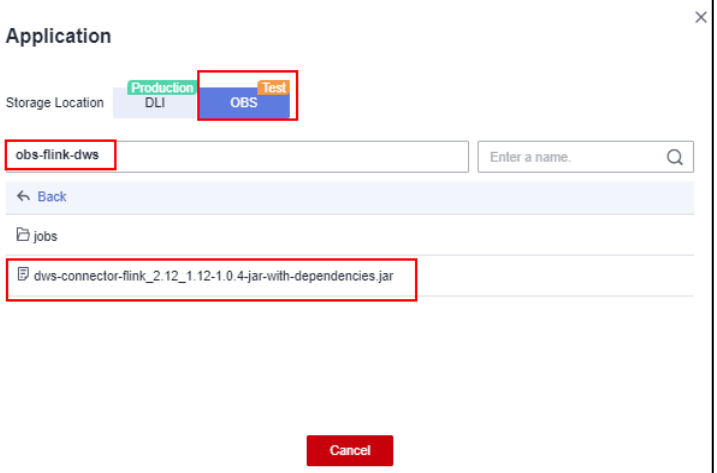
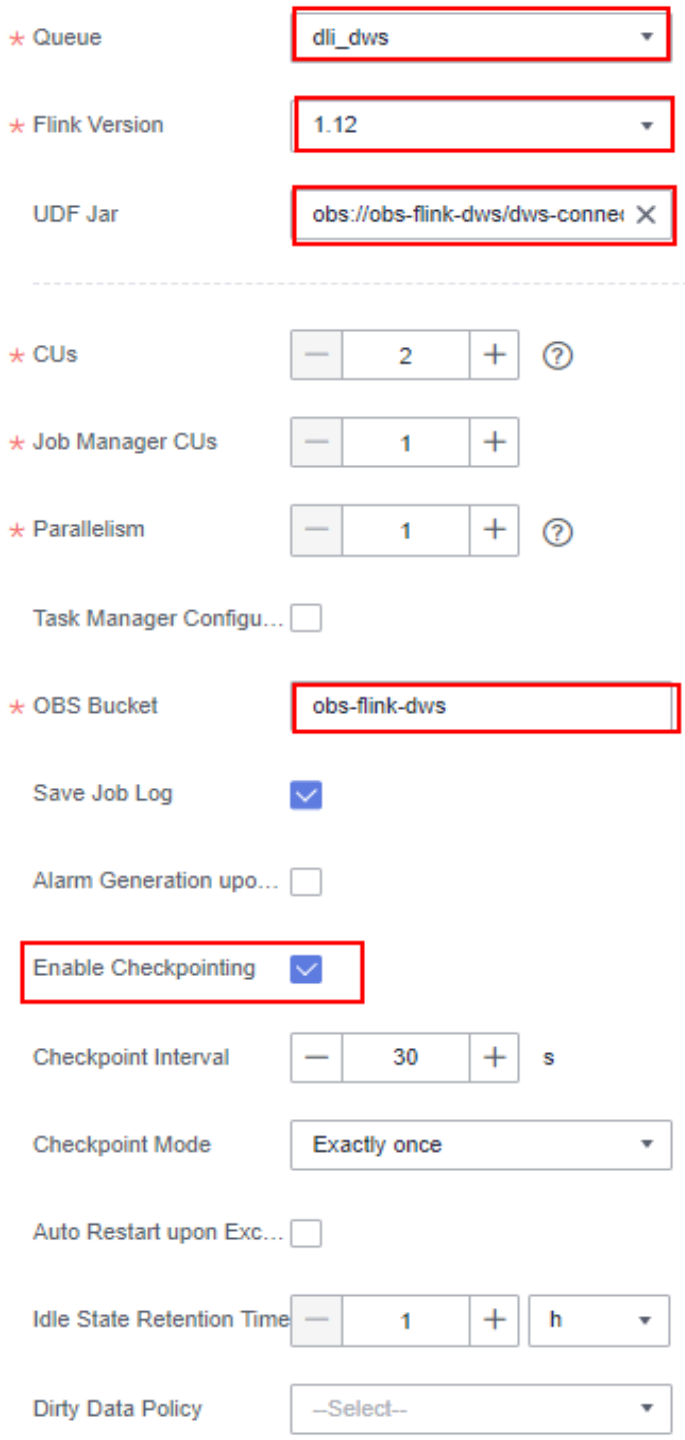
Parâmetro	Valor
UDF Jar	Selecione o arquivo JAR no bucket do OBS criado em **Passo 5: preparar a ferramenta dws-connector-flink para interconectar o GaussDB(DWS) com o Flink**. 
OBS Bucket	Selecione o bucket criado em **Passo 5: preparar a ferramenta dws-connector-flink para interconectar o GaussDB(DWS) com o Flink**.
Enable Checkpointing	Verifique a caixa.
Other parameters	Mantenha o valor padrão.

Figura 2-20 Editar um trabalho



Passo 5 Copie o seguinte código SQL para a janela de código SQL à esquerda.

Obtenha o endereço IP privado e o número da porta da instância do Kafka de **Passo 1.1** e obtenha o nome de domínio privado de **Passo 9**.

```
CREATE TABLE user_kafka (
  id string,
  name string,
  age int
) WITH (
```

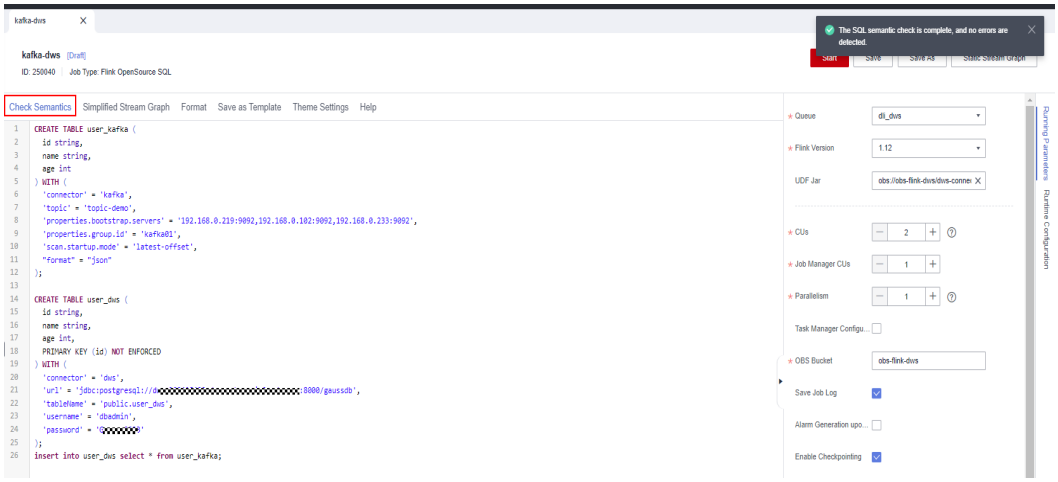


```
  'connector' = 'kafka',
  'topic' = 'topic-demo',
'properties.bootstrap.servers' ='Private IP address and port number of the Kafka
instance',
  'properties.group.id' = 'kafka01',
  'scan.startup.mode' = 'latest-offset',
  "format" = "json"
);

CREATE TABLE user_dws (
  id string,
  name string,
  age int,
  PRIMARY KEY (id) NOT ENFORCED
) WITH (
  'connector' = 'dws',
'url'='jdbc:postgresql://GaussDB(DWS) private network domain name:8000/gaussdb',
  'tableName' = 'public.user_dws',
  'username' = 'dbadmin',
'password' ='Password of database user dbdamin'
);

insert into user_dws select * from user_kafka;
```

Passo 6 Clique em **Check Semantics** e aguarde até que a verificação seja bem-sucedida.

Se a verificação falhar, verifique se a entrada SQL tem erros de sintaxe.

Figura 2-21 Instrução SQL de um trabalho



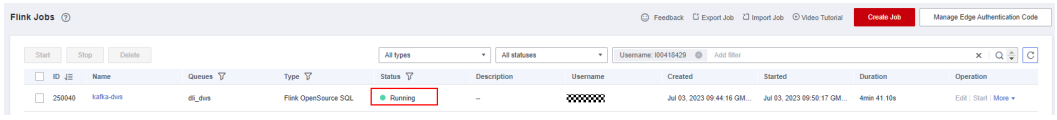
Passo 7 Clique em **Save**.

Passo 8 Volte para a home page do console do DLI e escolha **Job Management** > **Flink Jobs** à esquerda.

Passo 9 Clique em **Start** à direita do nome do trabalho **kafka-dws** e clique em **Start Now**.

Aguarde cerca de 1 minuto e atualize a página. Se o status for **Running**, o trabalho será executado com êxito.

Figura 2-22 Status de execução do trabalho



----Fim

Passo 7: criar e modificar mensagens no cliente do Kafka

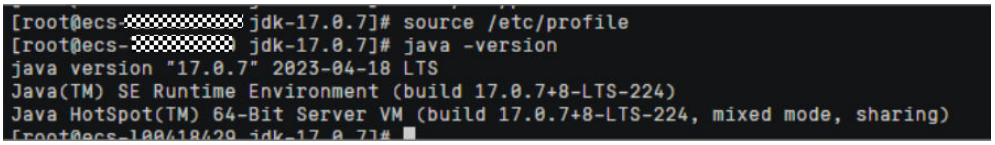
Passo 1 Crie um ECS consultando o documento do ECS. Certifique-se de que a região e o VPC do ECS sejam iguais aos do Kafka.

Passo 2 Instale o JDK.

1. Faça logon no ECS, vá para o diretório **/usr/local** e faça download do pacote JDK.
   ```
   cd /usr/local
   wget https://download.oracle.com/java/17/latest/jdk-17_linux-x64_bin.tar.gz
   ```
2. Descompacte o pacote JDK baixado.
   ```
   tar -zxvf jdk-17_linux-x64_bin.tar.gz
   ```
3. Execute o seguinte comando para abrir o arquivo **/etc/profile**:
   ```
   vim /etc/profile
   ```
4. Pressione **i** para entrar no modo de edição e adicione o seguinte conteúdo ao final do arquivo **/etc/profile**:
   ```
   export JAVA_HOME=/usr/local/jdk-17.0.7 #JDK installation directory
   export JRE_HOME=${JAVA_HOME}/jre
   export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib:${JAVA_HOME}/test:$
   {JAVA_HOME}/lib/gsjdbc4.jar:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib/
   tools.jar:$CLASSPATH
   export JAVA_PATH=${JAVA_HOME}/bin:${JRE_HOME}/bin
   export PATH=$PATH:${JAVA_PATH}
   ```

5. Pressione **Esc** e insira **:wq!** para salvar a configuração e sair.
6. Execute o seguinte comando para que as variáveis de ambiente entrem em vigor:
   ```
   source /etc/profile
   ```
7. Execute o seguinte comando. Se as seguintes informações forem exibidas, o JDK será instalado com sucesso:
   ```
   java -version
   ```

Passo 3 Instale o cliente do Kafka.

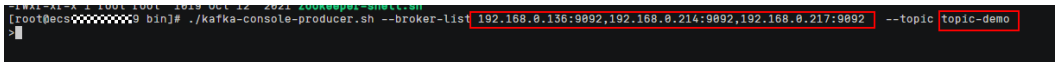
1. Vá para o diretório **/opt** e execute o seguinte comando para obter o pacote de software cliente do Kafka.
   ```
   cd /opt
   wget https://archive.apache.org/dist/kafka/2.7.2/kafka_2.12-2.7.2.tgz
   ```
2. Descompacte o pacote de software baixado.
   ```
   tar -zxf kafka_2.12-2.7.2.tgz
   ```
3. Vá para o diretório do cliente do Kafka.
   ```
   cd /opt/kafka_2.12-2.7.2/bin
   ```

Passo 4 Execute o seguinte comando para se conectar ao Kafka: {*Connection address*} indica o endereço de conexão de rede interna do Kafka. Para obter detalhes sobre como obter o endereço, consulte **Passo 1.1**. *topic* indica o nome do tópico do Kafka criado em **Passo 6**.

```
./kafka-console-producer.sh --broker-list {connection address} --topic {Topic
name}
```

O seguinte é um exemplo:

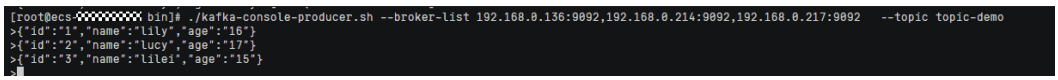
./kafka-console-producer.sh --broker-list
192.168.0.136:9092,192.168.0.214:9092,192.168.0.217:9092 --topic topic-demo



Se > for exibido e nenhuma outra mensagem de erro for exibida, a conexão foi bem-sucedida.

Passo 5 Na janela do cliente do Kafka conectado, copie o seguinte conteúdo (uma linha por vez) com base nos dados planejados em **Descrição do cenário** e pressione **Enter** para produzir mensagens:

```
{"id":"1","name":"lily","age":"16"}
{"id":"2","name":"lucy","age":"17"}
{"id":"3","name":"lilei","age":"15"}
```



Passo 6 Retorne ao console do GaussDB(DWS), escolha **Clusters** à esquerda e clique em **Log In** à direita do cluster do GaussDB(DWS). A página SQL é exibida.

Passo 7 Execute a seguinte instrução SQL. Você pode descobrir que os dados são salvos com sucesso no banco de dados em tempo real.

```
SELECT * FROM user_dws ORDER BY id;
```

	id	name	age
1	1	lily	16
2	2	lucy	17
3	3	lilei	15

Passo 8 Volte para a janela do cliente para se conectar ao Kafka no ECS, copie o conteúdo a seguir (uma linha por vez) e pressione **Enter** para produzir mensagens.

```
{"id":"2","name":"jim","age":"17"}
{"id":"3","name":"tom","age":"15"}
```

Passo 9 Volte para a janela SQL aberta do GaussDB(DWS) e execute a seguinte instrução SQL. Verificou-se que os nomes cujos IDs são **2** e **3** foram alterados para **jim** e **tom**.

A descrição do cenário é como esperado. Fim dessa prática.

```
SELECT * FROM user_dws ORDER BY id;
```

	id	name	age
1	1	lily	16
2	2	jim	17
3	3	tom	15

----Fim

2.4 Prática de interconexão de dados entre dois clusters do DWS baseados em GDS

Essa prática demonstra como migrar 15 milhões de linhas de dados entre dois clusters do armazém de dados em minutos com base na alta simultaneidade de importação e exportação de GDS.

NOTA

- Esta função é suportada apenas por clusters da versão 8.1.2 ou posterior.
- O GDS é uma ferramenta de importação e exportação de alta concorrência desenvolvida pelo GaussDB(DWS). Para obter mais informações, visite **Descrição de uso do GDS**.
- Esta seção descreve apenas a prática de operação. Para obter detalhes sobre a interconexão do GDS e a descrição da sintaxe, consulte **Interconexão entre clusters baseada em GDS**.

Essa prática leva cerca de 90 minutos. Os recursos de serviço de nuvem usados nessa prática são Data Warehouse Service (DWS), Elastic Cloud Server (ECS) e Virtual Private Cloud (VPC). O processo básico é o seguinte:

1. **Preparativos**
2. **Passo 1: criar dois clusters do DWS**
3. **Passo 2: preparar dados de origem**
4. **Passo 3: instalar e iniciai o servidor do GDS**
5. **Passo 4: implementar interconexão de dados em clusters do DWS**

Regiões suportadas

Tabela 2-17 descreve as regiões onde os dados do OBS foram carregados.

Tabela 2-17 Regiões e nomes de bucket do OBS

Região	Bucket de OBS
CN North-Beijing1	dws-demo-cn-north-1
CN North-Beijing2	dws-demo-cn-north-2
CN North-Beijing4	dws-demo-cn-north-4
CN North-Ulanqab1	dws-demo-cn-north-9
CN East-Shanghai1	dws-demo-cn-east-3
CN East-Shanghai2	dws-demo-cn-east-2
CN South-Guangzhou	dws-demo-cn-south-1
CN South-Guangzhou-InvitationOnly	dws-demo-cn-south-4
CN-Hong Kong	dws-demo-ap-southeast-1
AP-Singapore	dws-demo-ap-southeast-3
AP-Bangkok	dws-demo-ap-southeast-2
LA-Santiago	dws-demo-la-south-2
AF-Johannesburg	dws-demo-af-south-1
LA-Mexico City1	dws-demo-na-mexico-1
LA-Mexico City2	dws-demo-la-north-2

Região	Bucket de OBS
RU-Moscow2	dws-demo-ru-northwest-2
LA-Sao Paulo1	dws-demo-sa-brazil-1

Restrições

Nessa prática, dois conjuntos de serviços DWS e ECS são implantados na mesma região e VPC para garantir a conectividade de rede.

Preparativos

- Você obteve o AK e SK da conta.
- Você criou uma VPC e uma sub-rede. Para obter detalhes, consulte **Criação de uma VPC**.

Passo 1: criar dois clusters do DWS

Crie dois clusters do GaussDB(DWS) na região China-Hong Kong. Para obter detalhes, consulte **Criação de um cluster**. Os dois clusters são denominados dws-demo01 e dws-demo02.

Passo 2: preparar dados de origem

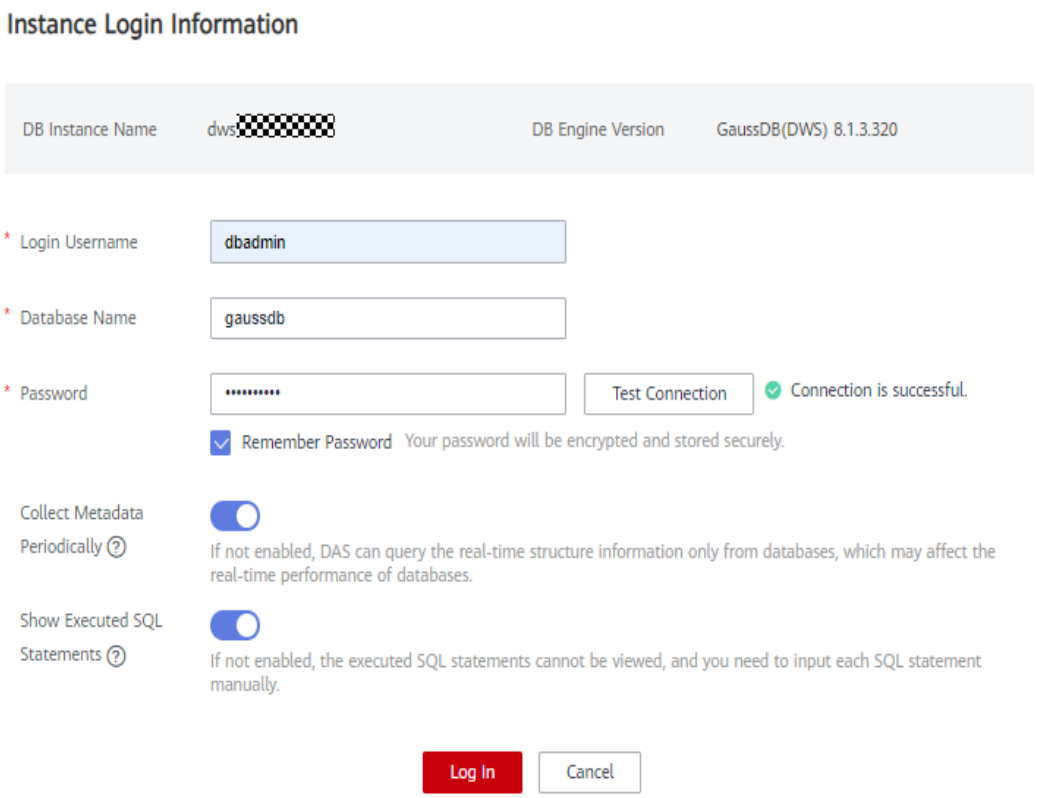
Passo 1 Na página Cluster Management do console do GaussDB(DWS), clique em Login na coluna Operation do cluster de origem dws-demo01.

NOTA

Esta prática usa a versão 8.1.3.x como exemplo. 8.1.2 e versões anteriores não suportam este modo de logon. Você pode usar o Data Studio para se conectar a um cluster. Para obter detalhes, consulte **Uso do Data Studio para se conectar a um cluster**.

Passo 2 O nome de usuário de logon é **dbadmin**, o nome do banco de dados é **gaussdb** e a senha é a senha do usuário **dbadmin** definida durante a criação do cluster do armazém de dados. Selecione **Remember Password**, ative **Collect Metadata Periodically** e **Show Executed SQL Statements** e clique em **Log In**.

Figura 2-23 Fazer logon no GaussDB(DWS)



Passo 3 Clique no nome do banco de dados **gaussdb** e clique em **SQL Window** no canto superior direito para acessar o editor SQL.

Passo 4 Copie a seguinte instrução SQL para a janela SQL e clique em Execute SQL para criar a tabela TPC-H de teste ORDERS.

```
CREATE TABLE ORDERS
 (
O_ORDERKEY BIGINT NOT NULL ,
O_CUSTKEY BIGINT NOT NULL ,
O_ORDERSTATUS CHAR(1) NOT NULL ,
O_TOTALPRICE DECIMAL(15,2) NOT NULL ,
O_ORDERDATE DATE NOT NULL ,
O_ORDERPRIORITY CHAR(15) NOT NULL ,
O_CLERK CHAR(15) NOT NULL ,
O_SHIPPRIORITY BIGINT NOT NULL ,
O_COMMENT VARCHAR(79) NOT NULL)
with (orientation = column)
distribute by hash(O_ORDERKEY)
PARTITION BY RANGE(O_ORDERDATE)
(
PARTITION O_ORDERDATE_1 VALUES LESS THAN('1993-01-01 00:00:00'),
PARTITION O_ORDERDATE_2 VALUES LESS THAN('1994-01-01 00:00:00'),
PARTITION O_ORDERDATE_3 VALUES LESS THAN('1995-01-01 00:00:00'),
PARTITION O_ORDERDATE_4 VALUES LESS THAN('1996-01-01 00:00:00'),
PARTITION O_ORDERDATE_5 VALUES LESS THAN('1997-01-01 00:00:00'),
PARTITION O_ORDERDATE_6 VALUES LESS THAN('1998-01-01 00:00:00'),
PARTITION O_ORDERDATE_7 VALUES LESS THAN('1999-01-01 00:00:00')
);
```

Passo 5 Execute a seguinte instrução SQL para criar uma tabela estrangeira do OBS:

Substitua AK e SK pelos AK e SK reais da conta. <obs_bucket_name> é obtido de **Regiões suportadas**.

> **NOTA**
>
> // AK e SK codificados rigidamente ou em texto não criptografado são arriscados. Para fins de segurança, criptografe seu AK e SK e armazene-os no arquivo de configuração ou nas variáveis de ambiente.

```
CREATE FOREIGN TABLE ORDERS01
 (
LIKE orders
 )
 SERVER gsmpp_server
 OPTIONS (
 ENCODING 'utf8',
 LOCATION obs://<obs_bucket_name>/tpch/orders.tbl',
 FORMAT 'text',
 DELIMITER '|',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
 CHUNKSIZE '64',
 IGNORE_EXTRA_DATA 'on'
 );
```

Passo 6 Execute a seguinte instrução SQL para importar dados da tabela estrangeira do OBS para o cluster de armazém de dados de origem: A importação leva cerca de 2 minutos. Por favor, aguarde.

> **NOTA**
>
> Se ocorrer um erro de importação, os valores de AK e SK da tabela estrangeira estão incorretos. Neste caso, execute o comando DROP FOREIGN TABLE order01; para excluir a tabela estrangeira, criar uma tabela estrangeira novamente e execute a seguinte instrução para importar dados novamente:

```
INSERT INTO orders SELECT * FROM orders01;
```

Passo 7 Repita as etapas anteriores para efetuar logon no cluster de destino dws-demo02 e execute a seguinte instrução SQL para criar as ordens da tabela de destino:

```
CREATE TABLE ORDERS
 (
 O_ORDERKEY BIGINT NOT NULL ,
 O_CUSTKEY BIGINT NOT NULL ,
 O_ORDERSTATUS CHAR(1) NOT NULL ,
 O_TOTALPRICE DECIMAL(15,2) NOT NULL ,
 O_ORDERDATE DATE NOT NULL ,
 O_ORDERPRIORITY CHAR(15) NOT NULL ,
 O_CLERK CHAR(15) NOT NULL ,
 O_SHIPPRIORITY BIGINT NOT NULL ,
 O_COMMENT VARCHAR(79) NOT NULL)
 with (orientation = column)
 distribute by hash(O_ORDERKEY)
 PARTITION BY RANGE(O_ORDERDATE)
 (
 PARTITION O_ORDERDATE_1 VALUES LESS THAN('1993-01-01 00:00:00'),
 PARTITION O_ORDERDATE_2 VALUES LESS THAN('1994-01-01 00:00:00'),
 PARTITION O_ORDERDATE_3 VALUES LESS THAN('1995-01-01 00:00:00'),
 PARTITION O_ORDERDATE_4 VALUES LESS THAN('1996-01-01 00:00:00'),
 PARTITION O_ORDERDATE_5 VALUES LESS THAN('1997-01-01 00:00:00'),
 PARTITION O_ORDERDATE_6 VALUES LESS THAN('1998-01-01 00:00:00'),
 PARTITION O_ORDERDATE_7 VALUES LESS THAN('1999-01-01 00:00:00')
 );
```

----Fim

Passo 3: instalar e iniciai o servidor do GDS

Passo 1 Crie um ECS consultando **Compra de um ECS**. Observe que as instâncias do ECS e GaussDB(DWS) devem ser criadas na mesma região e VPC. Neste exemplo, a versão do CentOS 7.6 é selecionada como a imagem do ECS.

Passo 2 Baixar o pacote do GDS

1. Efetue logon no console do GaussDB(DWS).
2. Na árvore de navegação à esquerda, clique em **Connections**.
3. Selecione o cliente do GDS da versão correspondente na lista suspensa de **CLI Client**.

   Selecione uma versão com base na versão do cluster e no SO em que o cliente está instalado.
4. Clique em **Download**.

Passo 3 Use a ferramenta SFTP para fazer upload do cliente baixado (por exemplo, **dws_client_8.2.x_redhat_x64.zip**) para o diretório /opt do ECS.

Passo 4 Efetue logon no ECS como o usuário root e execute os seguintes comandos para ir para o diretório /opt e descompactar o pacote do cliente:

```
cd /opt
unzip dws_client_8.2.x_redhat_x64.zip
```

Passo 5 Crie um usuário do GDS e o grupo de usuários ao qual o usuário pertence. Este usuário é usado para iniciar o GDS e ler os dados de origem.

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

Passo 6 Altere o proprietário do diretório do pacote do GDS e do diretório do arquivo de dados de origem para o usuário do GDS.

```
chown -R gds_user:gdsgrp /opt/gds/bin
chown -R gds_user:gdsgrp /opt
```

Passo 7 Alterne para o usuário gds.

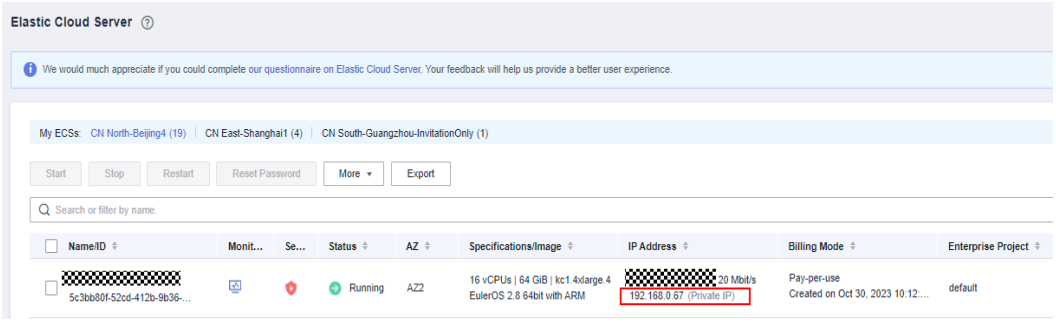
```
su - gds_user
```

Passo 8 Execute os seguintes comandos para ir para o diretório gds e executar variáveis de ambiente:

```
cd /opt/gds/bin
source gds_env
```

Passo 9 Execute o seguinte comando para iniciar o GDS. Você pode exibir o endereço IP interno do ECS no console do ECS.

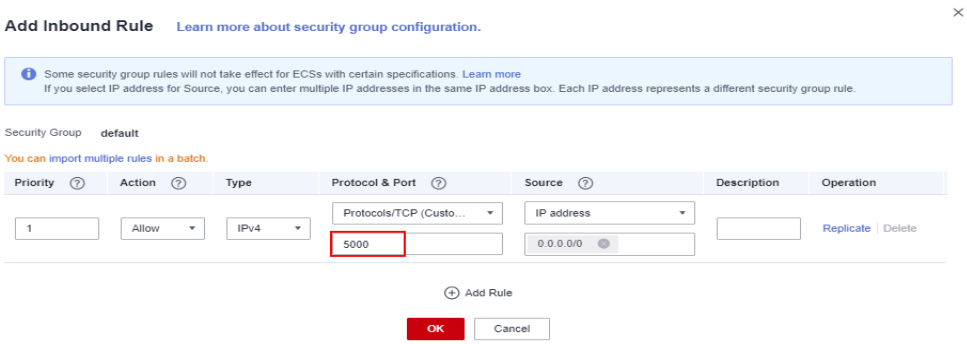
```
/opt/gds/bin/gds -d /opt -p  ECS Intranet IP:5000 -H 0.0.0.0/0 -l /opt/gds/bin/
gds_log.txt -D -t 2
```



Passo 10 Ative a porta de rede entre o ECS e o DWS.

O servidor do GDS (ECS neste experimento) precisa se comunicar com o DWS. O grupo de segurança padrão do ECS não permite tráfego de entrada da porta do GDS 5000 e da porta DWS 8000. Execute as seguintes etapas:

1. Retorne ao console do ECS e clique no nome do ECS para acessar a página de detalhes do ECS.
2. Alterne para a guia Security Groups e clique em Configure Rule.
3. Selecione Inbound Rules, clique em Add Rule, defina Priority como 1, defina Protocol Port como 5000 e clique em OK.



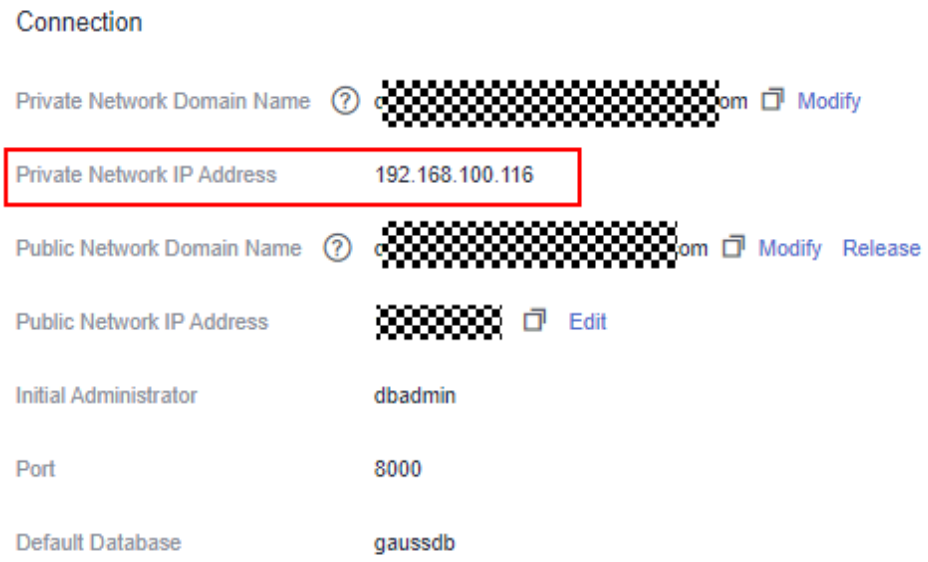
4. Repita as etapas anteriores para adicionar uma regra de entrada de 8000.

----Fim

Passo 4: implementar interconexão de dados em clusters do DWS

Passo 1 Crie um servidor.

1. Obtenha o endereço IP privado do cluster de armazém de dados de origem: Alterne para o console do DWS, escolha Cluster Management à esquerda e clique no nome de cluster de origem dws-demo01.
2. Vá para a página de detalhes do cluster e registre o endereço IP interno do DWS.



3. Volte para o console do DWS e clique em Log In na coluna Operation do destino dws-demo02. A janela SQL é exibida,

Execute o seguinte comando para criar um servidor:

O endereço IP privado do cluster de armazém de dados de origem é obtido na etapa anterior. O endereço IP privado do servidor do ECS é obtido do console do ECS. A senha de logon do usuário dbadmin é definida quando o cluster do armazém de dados é criado.

```
CREATE SERVER server_remote FOREIGN DATA WRAPPER GC_FDW OPTIONS
 (
address'Private network IP address of the source DWS cluster :8000',
 dbname 'gaussdb',
 username 'dbadmin',
password'Password of user dbadmin',
syncsrv'gsfs://Internal IP address of the ECS server:5000'
 )
 ;
```

Passo 2 Crie uma tabela estrangeira para interconexão.

Na janela SQL do cluster de destino dws-demo02, execute o seguinte comando para criar uma tabela estrangeira para interconexão:

```
CREATE FOREIGN TABLE ft_orders
 (
 O_ORDERKEY BIGINT ,
 O_CUSTKEY BIGINT ,
 O_ORDERSTATUS CHAR(1) ,
 O_TOTALPRICE DECIMAL(15,2) ,
 O_ORDERDATE DATE ,
 O_ORDERPRIORITY CHAR(15) ,
 O_CLERK CHAR(15) ,
 O_SHIPPRIORITY BIGINT ,
 O_COMMENT VARCHAR(79)

 )
 SERVER server_remote
 OPTIONS
 (
 schema_name 'public',
 table_name 'orders',
 encoding 'SQL_ASCII'
 );
```

Passo 3 Importe todos os dados da tabela.

Na janela SQL, execute a seguinte instrução SQL para importar dados completos da tabela estrangeira ft_orders: Aguarde cerca de 1 minuto.

```
INSERT INTO orders SELECT * FROM ft_orders;
```

Execute a seguinte instrução SQL. Constatou-se que 15 milhões de linhas de dados são importadas com sucesso.

```
SELECT count(*) FROM orders;
```

Passo 4 Importe dados com base em critérios de filtro.

Execute as seguintes instruções SQL para importar dados com base nos critérios de filtro:

```
INSERT INTO orders SELECT * FROM ft_orders WHERE o_orderkey < '10000000';
```

----Fim

3 Práticas da otimização de tabela

3.1 Projeto da estrutura da tabela

Antes de otimizar uma tabela, você precisa entender a estrutura da tabela. Durante o design do banco de dados, alguns fatores-chave sobre o design da tabela afetarão muito o desempenho de consulta subsequente do banco de dados. O design da tabela também afeta o armazenamento de dados. O design da tabela científica reduz as operações de I/O e minimiza o uso de memória, melhorando o desempenho da consulta.

Esta seção descreve como otimizar o desempenho da tabela no GaussDB(DWS) projetando corretamente a estrutura da tabela (por exemplo, configurando o modo de armazenamento da tabela, o nível de compactação, o modo de distribuição, a coluna de distribuição, as tabelas particionadas e o agrupamento local).

Selecionar um tipo de armazenamento

Selecionar um modelo para armazenamento de tabela é o primeiro passo da definição de tabela. Selecione um modelo de armazenamento adequado para o seu serviço com base na tabela abaixo.

Geralmente, se uma tabela contém muitas colunas (chamada de tabela ampla) e sua consulta envolve apenas algumas colunas, o armazenamento de colunas é recomendado. Se uma tabela contiver apenas algumas colunas e uma consulta que envolve a maioria das colunas, recomenda-se o armazenamento de linhas.

Modelo de armazenamento	**Cenário de aplicação**
Armazenamento de linha	Consulta de ponto (consulta baseada em índice simples que retorna apenas alguns registros). Consulta envolvendo muitas operações **INSERT**, **UPDATE** e **DELETE**.
Armazenamento de coluna	Consulta de análise de estatísticas, na qual operações, como group e join, são executadas muitas vezes.

O armazenamento de linha/coluna de uma tabela é especificado pelo atributo **orientation** na definição da tabela. O valor **row** indica uma tabela de armazenamento de linha e **column** indica uma tabela de armazenamento de coluna. O valor padrão é **row**.

Compressão de tabela

A compactação de tabela pode ser ativada quando uma tabela é criada. A compactação de tabela permite que os dados da tabela sejam armazenados em formato compactado para reduzir o uso de memória.

Em cenários em que a I/O é grande (muitos dados são lidos e gravados) e a CPU é suficiente (poucos dados são computados), selecione uma alta taxa de compactação. Em cenários em que a I/O é pequena e a CPU é insuficiente, selecione uma taxa de compactação baixa. Com base neste princípio, é aconselhável selecionar diferentes taxas de compressão e testar e comparar os resultados para selecionar a taxa de compressão ideal, conforme necessário. Especifique uma taxa de compressão usando o parâmetro **COMPRESSION**. Os valores suportados são os seguintes:

- O valor válido das tabelas de armazenamento de colunas é **YES**, **NO**, **LOW**, **MIDDLE** ou **HIGH**, e o valor padrão é **LOW**.
- Os valores válidos de tabelas de armazenamento de linha são **YES** e **NO**, e o padrão é **NO**. (A função de compactação de tabela de armazenamento de linha não é colocada em uso comercial. Para usar essa função, entre em contato com o suporte técnico.)

Os cenários de serviço aplicáveis a cada nível de compactação são descritos na tabela a seguir.

Nível de compressão	Cenário de aplicação
LOW	O uso da CPU do sistema é alto e o espaço de armazenamento em disco é suficiente.
MIDDLE	O uso da CPU do sistema é moderado e o espaço de armazenamento em disco é insuficiente.
HIGH	O uso da CPU do sistema é baixo e o espaço de armazenamento em disco é insuficiente.

Selecionar um modo de distribuição

GaussDB(DWS) suporta os seguintes modos de distribuição: replication, hash e Round-robin.

NOTA

Round-robin é suportado no cluster 8.1.2 e posterior.

Política	Descrição	Cenário de aplicação	Vantagens/desvantagens
Replication	Os dados completos em uma tabela são armazenados em cada DN no cluster.	Pequenas tabelas e tabelas de dimensões	● A vantagem da replicação é que cada DN tem dados completos da tabela. Durante a operação de junção, os dados não precisam ser redistribuídos, reduzindo as sobrecargas de rede e reduzindo os segmentos do plano (cada segmento do plano inicia um thread correspondente). ● A desvantagem da replicação é que cada DN retém os dados completos da tabela, resultando em redundância de dados. Geralmente, a replicação é usada apenas para tabelas de pequenas dimensões.
Hash	Os dados da tabela são distribuídos em todos os DNs no cluster.	Tabelas de fatos contendo uma grande quantidade de dados	● Os recursos de I/O de cada nó podem ser usados durante a leitura/gravação de dados, melhorando consideravelmente a velocidade de leitura/gravação de uma tabela. ● Geralmente, uma tabela grande (contendo mais de 1 milhão de registros) é definida como uma tabela hash.

Política	Descrição	Cenário de aplicação	Vantagens/desvantagens
Polling (Round-robin)	Cada linha na tabela é enviada a cada DN por sua vez. Os dados podem ser distribuídos uniformemente em cada DN.	Tabelas de fatos que contêm uma grande quantidade de dados e não conseguem encontrar uma chave de distribuição adequada no modo hash	● Round-robin pode evitar distorção de dados, melhorando a utilização do espaço do cluster. ● Round-robin não suporta otimização de DN local como uma tabela de hash faz, e o desempenho de consulta Round-robin é geralmente menor do que o de uma tabela hash. ● Se uma chave de distribuição adequada puder ser encontrada para uma tabela grande, use o modo de distribuição de hash com melhor desempenho. Caso contrário, defina a tabela como uma tabela round-robin.

Selecionar uma chave de distribuição

Se o modo de distribuição hash for usado, uma chave de distribuição deve ser especificada para a tabela de usuário. Se um registro for inserido, o sistema executará o cálculo de hash com base nos valores na coluna de distribuição e, em seguida, armazenará os dados no DN relacionado.

Selecione uma chave de distribuição de hash com base nos seguintes princípios:

1. **Os valores da chave de distribuição devem ser discretos para que os dados possam ser distribuídos uniformemente em cada DN.** Você pode selecionar a chave primária da tabela como a chave de distribuição. Por exemplo, para uma tabela de informações da pessoa, escolha a coluna do número de ID como a chave de distribuição.
2. **Não selecione a coluna onde existe um filtro constante.** Por exemplo, se uma restrição constante (por exemplo, zqdh= '000001') existe na coluna **zqdh** em algumas consultas na tabela **dwcjk**, não é aconselhável usar **zqdh** como a chave de distribuição.
3. **Com os princípios acima atendidos, você pode selecionar condições de junção como chaves de distribuição**, para que as tarefas de junção possam ser enviadas para DNs para execução, reduzindo a quantidade de dados transferidos entre os DNs.

   Para uma tabela hash, uma chave de distribuição imprópria pode causar distorção de dados ou desempenho ruim de I/O em determinados DNs. Portanto, você precisa verificar a tabela para garantir que os dados sejam distribuídos uniformemente em cada DN. Você pode executar as seguintes instruções SQL para verificar a distorção de dados:

   ```
   SELECT
   xc_node_id, count(1)
   FROM tablename
   ```



```
GROUP BY xc_node_id
ORDER BY xc_node_id desc;
```

xc_node_id corresponde a um DN. Geralmente, **mais de 5% de diferença entre a quantidade de dados em diferentes DNs é considerada como distorção de dados. Se a diferença for superior a 10%, escolha outra chave de distribuição.**

4. Não é aconselhável adicionar uma coluna como uma chave de distribuição, especialmente adicionar uma nova coluna e usar o valor de SEQUENCE para preencher a coluna. (Sequências podem causar gargalos de desempenho e custos de manutenção desnecessários.)

Usar tabelas particionadas

O particionamento refere-se a dividir o que é logicamente uma grande tabela em pedaços físicos menores com base em esquemas específicos. A tabela baseada na lógica é chamada de tabela particionada, e uma parte física é chamada de partição. Os dados são armazenados nessas partes físicas menores, ou seja, partições, em vez da tabela particionada lógica maior. Uma tabela particionada tem as seguintes vantagens sobre uma tabela comum:

1. Alto desempenho de consulta: o sistema consulta apenas as partições em questão, em vez de toda a tabela, melhorando a eficiência da consulta.
2. Alta disponibilidade: se uma partição estiver com defeito, os dados nas outras partições ainda estarão disponíveis.
3. Manutenção fácil: você só precisa corrigir a partição defeituosa.

As tabelas particionadas suportadas pelo GaussDB(DWS) incluem tabelas particionadas por intervalo e tabelas particionadas por lista. (As tabelas particionadas por lista são suportadas apenas no cluster 8.1.3).

Usar clustering parcial

Chave de cluster parcial é a tecnologia baseada em coluna. Ela pode minimizar ou maximizar índices esparsos para filtrar rapidamente tabelas base. Chave de cluster parcial pode especificar várias colunas, mas é aconselhável especificar não mais do que duas colunas. Use os seguintes princípios para especificar colunas:

1. As colunas selecionadas devem ser restritas por expressões simples em tabelas base. Tais restrições são geralmente representadas por Col, Op e Const. Col especifica o nome da coluna, Op especifica operadores, (incluindo =, >, >=, <= e <) Const especifica constantes.
2. Selecione colunas que são frequentemente selecionadas (para filtrar muito mais dados indesejados) em expressões simples.
3. Liste as colunas selecionadas com menos frequência na parte superior.
4. Liste as colunas do tipo enumerado na parte superior.

Selecionar um tipo de dados

Você pode usar tipos de dados com os seguintes recursos para melhorar a eficiência:

1. **Tipos de dados que aumentam a eficiência da execução**

   Geralmente, o cálculo de inteiros (incluindo cálculos de comparação comuns, como o =, >, <, ≥, ≤ e ≠ e **GROUP BY**) é mais eficiente do que o de cadeias e números de ponto flutuante. Por exemplo, se você precisar executar uma consulta de ponto em uma tabela de armazenamento de colunas cuja coluna **NUMERIC** é usada como critério de filtro, a

consulta levará mais de 10 segundos. Se você alterar o tipo de dados de **NUMERIC** para **INT**, a consulta leva apenas cerca de 1,8 segundos.

2. **Selecionar tipos de dados com um comprimento curto**

   Tipos de dados com comprimento curto reduzem tanto o tamanho do arquivo de dados quanto a memória usada para computação, melhorando o desempenho de I/O e computação. Por exemplo, use **SMALLINT** em vez de **INT** e **INT** em vez de**BIGINT**.

3. **Mesmo tipo de dados para uma junção**

   É aconselhável usar o mesmo tipo de dados para uma junção. Para unir colunas com diferentes tipos de dados, o banco de dados precisa convertê-las para o mesmo tipo, o que leva a sobrecargas de desempenho adicionais.

Uso do índice

- O objetivo da criação de índices é acelerar as consultas. Portanto, certifique-se de que os índices possam ser usados em algumas consultas. Se um índice não for usado por nenhuma instrução de consulta, o índice não terá sentido. Exclua o índice.
- Não crie índices secundários desnecessários. Índices secundários úteis podem acelerar a consulta. No entanto, o espaço ocupado pelos índices aumenta com o número de índices. Cada vez que um índice é adicionado, um par chave-valor adicional precisa ser adicionado quando um dado é inserido. Portanto, quanto mais índices, mais lenta a velocidade de gravação e maior o uso de espaço. Além disso, muitos índices afetam o tempo de execução do otimizador e índices inadequados enganam o otimizador. Portanto, quanto mais índices, melhor.
- Crie índices adequados com base nas características do serviço. Em princípio, os índices precisam ser criados para colunas necessárias em uma consulta para melhorar o desempenho. Os índices podem ser criados nos seguintes cenários:
  - Para colunas com alta diferenciação, os índices podem reduzir significativamente o número de linhas após a filtragem. Por exemplo, é aconselhável criar um índice na coluna número do cartão de identificação, mas não na coluna do gênero.
  - Se houver várias condições de consulta, você poderá selecionar um índice de combinação. Observe que a coluna da condição equivalente deve ser colocada antes do índice de combinação. Por exemplo, se a consulta comum for SELECT * FROM t onde c1 = 10 e c2 = 100 e c3 > 10, você pode criar o índice de combinação Index cidx (c1, c2, c3). Dessa forma, você pode usar as condições de consulta para construir um prefixo de índice para varredura.
- Quando uma coluna de índice é usada como uma condição de consulta, não execute cálculo, função ou conversão de tipo na coluna de índice. Caso contrário, o otimizador não poderá usar o índice.
- Certifique-se de que a coluna de índice contém a coluna de consulta. Não execute sempre a instrução SELECT * para consultar todas as colunas.
- A condição de consulta é usada. =. Quando NOT IN é usado, índices não podem ser usados.
- Quando LIKE é usado, se a condição começar com o caractere curinga %, o índice não poderá ser usado.
- Se vários índices estiverem disponíveis para uma condição de consulta, mas você souber qual índice é o ideal, é aconselhável usar a dica do otimizador para forçar o otimizador a usar o índice. Isso impede que o otimizador selecione um índice incorreto devido a estatísticas imprecisas ou outros problemas.

- Quando a expressão IN é usada como a condição de consulta, o número de condições correspondentes não deve ser muito grande. Caso contrário, a eficiência de execução é baixa.

3.2 Visão geral da otimização de tabelas

Nesta prática, você aprenderá como otimizar o design de suas tabelas. Você começará criando tabelas sem especificar seu modo de armazenamento, chave de distribuição, modo de distribuição ou modo de compactação. Carregue dados de teste nessas tabelas e teste o desempenho do sistema. Em seguida, siga excelentes práticas para criar as tabelas novamente usando novos modos de armazenamento, chaves de distribuição, modos de distribuição e modos de compactação. Carregue os dados de teste e teste o desempenho novamente. Compare os dois resultados do teste para descobrir como o design da tabela afeta o espaço de armazenamento e o desempenho de carregamento e consulta das tabelas.

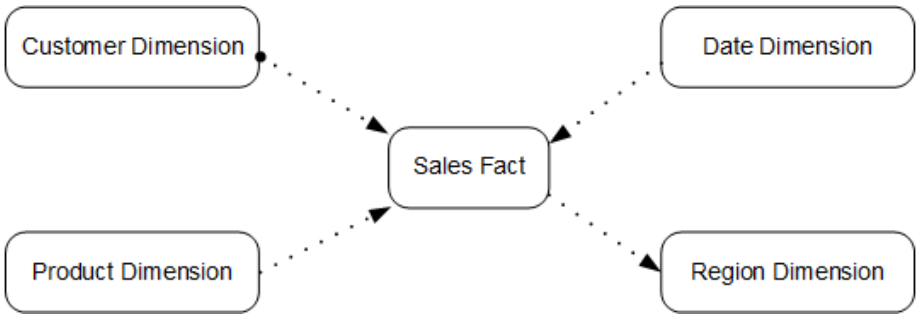
Tempo estimado: 60 minutos

3.3 Seleção de um modelo de tabela

Os tipos mais comuns de esquemas de armazém de dados são os esquemas de estrela e de floco de neve. Considere os requisitos de serviço e desempenho ao escolher um esquema para suas tabelas.

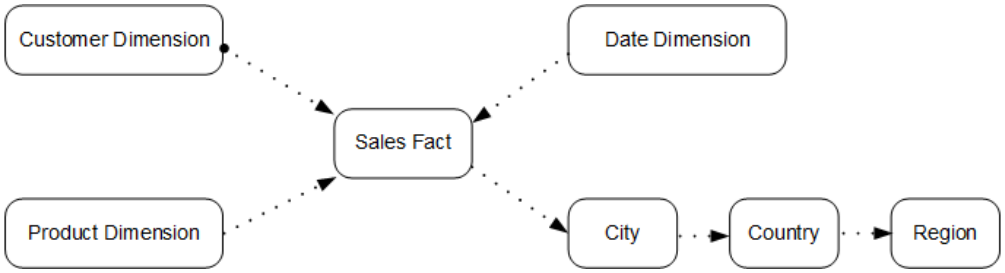
- No esquema estrela, uma tabela de fatos central contém os dados principais do banco de dados e várias tabelas de dimensão fornecem informações de atributos descritivos para a tabela de fatos. A chave primária de uma tabela de dimensão associa uma chave estrangeira em uma tabela de fatos, conforme mostrado na **Figura 3-1**.
  - Todos os fatos devem ter a mesma granularidade.
  - Dimensões diferentes não estão associadas.

Figura 3-1 Esquema de estrela



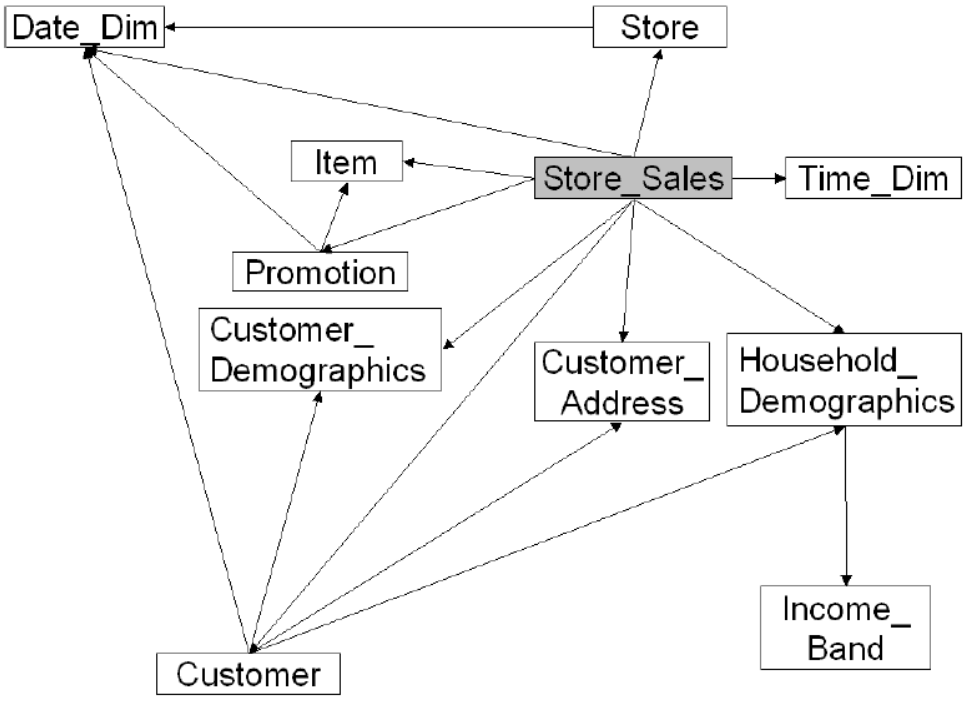
- O esquema de floco de neve é desenvolvido com base no esquema de estrela. Nesse esquema, cada dimensão pode ser associada a várias dimensões e dividida em tabelas de diferentes granularidades com base no nível da dimensão, conforme mostrado em **Figura 3-2**.
  - As tabelas de dimensão podem ser associadas conforme necessário, e os dados armazenados nelas são reduzidos.
  - Esse esquema tem mais tabelas de dimensão para manter do que o esquema de estrela.

Figura 3-2 Esquema de estrela



Esta prática verifica o desempenho usando o modelo de Vendas da loja (SS) de TPC-DS. O modelo usa o esquema de floco de neve. **Figura 3-3** ilustra a sua estrutura.

Figura 3-3 Diagrama ER de Vendas da loja de TPC-DS



Para obter detalhes sobre a tabela de fato **store_sales** e tabelas de dimensões no modelo, consulte o documento oficial do TPC-DS em **http://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp**.

3.4 Passo 1: criar uma tabela inicial e carregar dados de amostra

Regiões suportadas

Tabela 3-1 descreve as regiões onde os dados do OBS foram carregados.

Tabela 3-1 Regiões e nomes de bucket do OBS

Região	Bucket de OBS
CN North-Beijing1	dws-demo-cn-north-1
CN North-Beijing2	dws-demo-cn-north-2
CN North-Beijing4	dws-demo-cn-north-4
CN North-Ulanqab1	dws-demo-cn-north-9
CN East-Shanghai1	dws-demo-cn-east-3
CN East-Shanghai2	dws-demo-cn-east-2
CN South-Guangzhou	dws-demo-cn-south-1
CN South-Guangzhou-InvitationOnly	dws-demo-cn-south-4
CN-Hong Kong	dws-demo-ap-southeast-1
AP-Singapore	dws-demo-ap-southeast-3
AP-Bangkok	dws-demo-ap-southeast-2
LA-Santiago	dws-demo-la-south-2
AF-Johannesburg	dws-demo-af-south-1
LA-Mexico City1	dws-demo-na-mexico-1
LA-Mexico City2	dws-demo-la-north-2
RU-Moscow2	dws-demo-ru-northwest-2
LA-Sao Paulo1	dws-demo-sa-brazil-1

Crie um grupo de tabelas sem especificar seus modos de armazenamento, chaves de distribuição, modos de distribuição ou modos de compactação. Carregue dados de amostra nessas tabelas.

Passo 1 (Opcional) Crie um cluster.

Se um cluster estiver disponível, ignore esta etapa. Para obter detalhes sobre como criar um cluster, consulte **Criação de um cluster do GaussDB(DWS) 2.0**.

Conecte-se ao cluster e teste a conexão. Para obter detalhes, consulte **Métodos de conexão a um cluster**.

Esta prática usa um cluster de 8 nós como exemplo. Você também pode usar um cluster de quatro nós para executar o teste.

Passo 2 Crie uma tabela de teste SS **store_sales**.

> **NOTA**
>
> Antes de criar esta tabela, exclua as tabelas SS existentes primeiro (se houver) usando o comando **DROP TABLE**. Por exemplo, para excluir a tabela **store_sales**, execute o seguinte comando:
>
> ```
> DROP TABLE store_sales;
> ```

Não configure o modo de armazenamento, a chave de distribuição, o modo de distribuição ou o modo de compactação ao criar esta tabela.

Execute o comando **CREATE TABLE** para criar as 11 tabelas no **Figura 3-3**. Esta seção fornece apenas a sintaxe para criar a tabela **store_sales**. Para criar todas as tabelas, copie a sintaxe em **Criação de uma tabela inicial**.

```
CREATE TABLE store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer               not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          bigint                not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
    ss_ext_discount_amt       decimal(7,2)                  ,
    ss_ext_sales_price        decimal(7,2)                  ,
    ss_ext_wholesale_cost     decimal(7,2)                  ,
    ss_ext_list_price         decimal(7,2)                  ,
    ss_ext_tax                decimal(7,2)                  ,
    ss_coupon_amt             decimal(7,2)                  ,
    ss_net_paid               decimal(7,2)                  ,
    ss_net_paid_inc_tax       decimal(7,2)                  ,
    ss_net_profit             decimal(7,2)
) ;
```

Passo 3 Carregue dados de amostra nessas tabelas.

Um bucket do OBS fornece dados de exemplo usados para essa prática. O bucket pode ser lido por todos os usuários da nuvem autenticados. Execute as seguintes operações para carregar os dados de amostra:

1. Crie uma tabela estrangeira para cada tabela.

   GaussDB(DWS) usa os wrappers de dados estrangeiros (FDWs) fornecidos pelo PostgreSQL para importar dados em paralelo. Para usar FDWs, crie tabelas de FDW primeiro (também chamadas de tabelas estrangeiras). Esta seção fornece apenas a sintaxe para criar a tabela estrangeira **obs_from_store_sales_001** correspondente à tabela **store_sales**. Para criar todas as tabelas estrangeiras, copie a sintaxe em **Criação de uma tabela estrangeira**.

NOTA

- Observe que <*obs_bucket_name*> na instrução a seguir indica o nome do bucket do OBS. Apenas algumas regiões são suportadas. Para obter detalhes sobre as regiões suportadas e os nomes dos bucket do OBS, consulte **Tabela 3-1**. Os clusters do GaussDB(DWS) não oferecem suporte ao acesso entre regiões aos dados do bucket do OBS.
- As colunas da tabela estrangeira devem ser as mesmas da tabela ordinária correspondente. Neste exemplo, **store_sales** e **obs_from_store_sales_001** devem ter as mesmas colunas.
- A sintaxe da tabela estrangeira obtém os dados de exemplo usados para esta prática do bucket do OBS. Para carregar outros dados de amostra, modifique **SERVER gsmpp_server OPTIONS** conforme necessário. Para obter detalhes, consulte **Sobre a importação paralela de dados do OBS**.
- // AK e SK codificados rigidamente ou em texto não criptografado são arriscados. Para fins de segurança, criptografe seu AK e SK e armazene-os no arquivo de configuração ou nas variáveis de ambiente.

```
CREATE FOREIGN TABLE obs_from_store_sales_001
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer                not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          bigint                 not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
    ss_ext_discount_amt       decimal(7,2)                  ,
    ss_ext_sales_price        decimal(7,2)                  ,
    ss_ext_wholesale_cost     decimal(7,2)                  ,
    ss_ext_list_price         decimal(7,2)                  ,
    ss_ext_tax                decimal(7,2)                  ,
    ss_coupon_amt             decimal(7,2)                  ,
    ss_net_paid               decimal(7,2)                  ,
    ss_net_paid_inc_tax       decimal(7,2)                  ,
    ss_net_profit             decimal(7,2)
)
-- Configure OBS server information and data format details.
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/store_sales',
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
-- If create foreign table failed,record error message
WITH err_obs_from_store_sales_001;
```

2. Defina os parâmetros **ACCESS_KEY** e **SECRET_ACCESS_KEY** conforme necessário na instrução de criação de tabela estrangeira e execute essa instrução em uma ferramenta cliente para criar uma tabela estrangeira.

   Para obter os valores de **ACCESS_KEY** e **SECRET_ACCESS_KEY**, consulte **Criação de chaves de acesso (AK e SK)**.

3. Importe dados.

Crie o script **insert.sql** contendo as seguintes instruções e execute-o:

```
\timing on
\parallel on 4
INSERT INTO store_sales SELECT * FROM obs_from_store_sales_001;
INSERT INTO date_dim SELECT * FROM obs_from_date_dim_001;
INSERT INTO store SELECT * FROM obs_from_store_001;
INSERT INTO item SELECT * FROM obs_from_item_001;
INSERT INTO time_dim SELECT * FROM obs_from_time_dim_001;
INSERT INTO promotion SELECT * FROM obs_from_promotion_001;
INSERT INTO customer_demographics SELECT * from
obs_from_customer_demographics_001 ;
INSERT INTO customer_address SELECT * FROM obs_from_customer_address_001 ;
INSERT INTO household_demographics SELECT * FROM
obs_from_household_demographics_001;
INSERT INTO customer SELECT * FROM obs_from_customer_001;
INSERT INTO income_band SELECT * FROM obs_from_income_band_001;
\parallel off
```

Informação semelhante à seguinte é exibida:

```
SET
Timing is on.
SET
Time: 2.831 ms
Parallel is on with scale 4.
Parallel is off.
INSERT 0 402
Time: 1820.909 ms
INSERT 0 73049
Time: 2715.275 ms
INSERT 0 86400
Time: 2377.056 ms
INSERT 0 1000
Time: 4037.155 ms
INSERT 0 204000
Time: 7124.190 ms
INSERT 0 7200
Time: 2227.776 ms
INSERT 0 1920800
Time: 8672.647 ms
INSERT 0 20
Time: 2273.501 ms
INSERT 0 1000000
Time: 11430.991 ms
INSERT 0 1981703
Time: 20270.750 ms
INSERT 0 287997024
Time: 341395.680 ms
total time: 341584  ms
```

4. Calcule o tempo total gasto na criação das 11 tabelas. O resultado será registrado como o tempo de carregamento na tabela de referência em **Passo 1** na próxima seção.
5. Execute o comando a seguir para verificar se cada tabela é carregada corretamente e registra linhas na tabela:

```
SELECT COUNT(*) FROM store_sales;
SELECT COUNT(*) FROM date_dim;
SELECT COUNT(*) FROM store;
SELECT COUNT(*) FROM item;
SELECT COUNT(*) FROM time_dim;
SELECT COUNT(*) FROM promotion;
SELECT COUNT(*) FROM customer_demographics;
SELECT COUNT(*) FROM customer_address;
SELECT COUNT(*) FROM household_demographics;
SELECT COUNT(*) FROM customer;
SELECT COUNT(*) FROM income_band;
```

O número de linhas em cada tabela SS é o seguinte:

Nome da tabela	Número de linhas
Store_Sales	287997024
Date_Dim	73049
Store	402
Item	204000
Time_Dim	86400
Promotion	1000
Customer_Demographics	1920800
Customer_Address	1000000
Household_Demographics	7200
Customer	1981703
Income_Band	20

Passo 4 Execute o comando **ANALYZE** para atualizar as estatísticas.

```
ANALYZE;
```

Se **ANALYZE** for retornado, a execução é bem-sucedida.

```
ANALYZE
```

A instrução **ANALYZE** coleta estatísticas sobre o conteúdo da tabela em bancos de dados, que serão armazenadas no catálogo do sistema **PG_STATISTIC**. Em seguida, o otimizador de consulta usa as estatísticas para elaborar o plano de execução mais eficiente.

Depois de executar inserções e exclusões em lote, é aconselhável executar a instrução **ANALYZE** na tabela ou na biblioteca inteira para atualizar as estatísticas.

----Fim

3.5 Passo 2: testar o desempenho do sistema da tabela inicial e estabelecer uma linha de base

Antes e depois de ajustar as estruturas da tabela, teste e registre as seguintes informações para comparar as diferenças no desempenho do sistema:

- Tempo de carregamento
- Espaço de armazenamento ocupado por tabelas
- Desempenho da consulta

Os exemplos nesta prática são baseados em um cluster dws.d2.xlarge composto por oito nós. Como o desempenho do sistema é afetado por muitos fatores, clusters do mesmo sabor podem ter resultados diferentes.

Modelo	dws.d2.xlarge VM
CPU	4*CPU E5-2680 v2 @ 2.80GHZ
Memória	32 GB
Rede	1 GB
Disco	1,63 TB
Número de nós	8

Registre os resultados usando a seguinte tabela de referência.

Referência	Antes de	Depois
Tempo de carregamento (11 tabelas)	341584 ms	-
Espaço de armazenamento ocupado		
Store_Sales	-	-
Date_Dim	-	-
Store	-	-
Item	-	-
Time_Dim	-	-
Promotion	-	-
Customer_Demographics	-	-
Customer_Address	-	-
Household_Demographics	-	-
Customer	-	-
Income_Band	-	-
Espaço total de armazenamento	-	-
Tempo de execução da consulta		
Consulta 1	-	-
Consulta 2	-	-
Consulta 3	-	-
Tempo total de execução	-	-

Execute as seguintes etapas para testar o desempenho do sistema antes de ajustar para estabelecer uma referência:

Passo 1 Insira o tempo de carregamento cumulativo para todas as 11 tabelas na tabela de referências na coluna **Before**.

Passo 2 Registre o uso do espaço de armazenamento de cada tabela.

Determine quanto espaço em disco é usado para cada tabela usando a função **pg_size_pretty** e registre os resultados em tabelas básicas.

```
SELECT T_NAME, PG_SIZE_PRETTY(PG_RELATION_SIZE(t_name)) FROM
(VALUES('store_sales'),('date_dim'),('store'),('item'),('time_dim'),('promotion'),
('customer_demographics'),('customer_address'),('household_demographics'),
('customer'),('income_band')) AS names1(t_name);
```

As seguintes informações são exibidas:

```
         t_name         | pg_size_pretty
------------------------+----------------
 store_sales            | 42 GB
 date_dim               | 11 MB
 store                  | 232 kB
 item                   | 110 MB
 time_dim               | 11 MB
 promotion              | 256 kB
 customer_demographics  | 171 MB
 customer_address       | 170 MB
 household_demographics | 504 kB
 customer               | 441 MB
 income_band            | 88 kB
(11 rows)
```

Passo 3 Teste o desempenho da consulta.

Execute as seguintes consultas e registre o tempo gasto em cada consulta. As durações de execução da mesma consulta podem ser diferentes, dependendo do cache do sistema operacional durante a execução. É aconselhável realizar várias rodadas de testes e selecionar um grupo com valores médios.

```
\timing on
SELECT * FROM (SELECT  COUNT(*)
FROM store_sales
    ,household_demographics
    ,time_dim, store
WHERE ss_sold_time_sk = time_dim.t_time_sk
    AND ss_hdemo_sk = household_demographics.hd_demo_sk
    AND ss_store_sk = s_store_sk
    AND time_dim.t_hour = 8
    AND time_dim.t_minute >= 30
    AND household_demographics.hd_dep_count = 5
    AND store.s_store_name = 'ese'
ORDER BY COUNT(*)
 ) LIMIT 100;

SELECT * FROM (SELECT  i_brand_id brand_id, i_brand brand, i_manufact_id,
i_manufact,
SUM(ss_ext_sales_price) ext_price
 FROM date_dim, store_sales, item,customer,customer_address,store
 WHERE d_date_sk = ss_sold_date_sk
   AND ss_item_sk = i_item_sk
   AND i_manager_id=8
   AND d_moy=11
   AND d_year=1999
   AND ss_customer_sk = c_customer_sk
   AND c_current_addr_sk = ca_address_sk
   AND substr(ca_zip,1,5) <> substr(s_zip,1,5)
```



```
   AND ss_store_sk = s_store_sk
 GROUP BY i_brand
      ,i_brand_id
      ,i_manufact_id
      ,i_manufact
 ORDER BY ext_price desc
         ,i_brand
         ,i_brand_id
         ,i_manufact_id
         ,i_manufact
 ) LIMIT 100;

SELECT * FROM (SELECT  s_store_name, s_store_id,
        SUM(CASE WHEN (d_day_name='Sunday') THEN ss_sales_price ELSE null END)
sun_sales,
        SUM(CASE WHEN (d_day_name='Monday') THEN ss_sales_price ELSE null END)
mon_sales,
        SUM(CASE WHEN (d_day_name='Tuesday') THEN ss_sales_price ELSE  null END)
tue_sales,
        SUM(CASE WHEN (d_day_name='Wednesday') THEN ss_sales_price ELSE null END)
wed_sales,
        SUM(CASE WHEN (d_day_name='Thursday') THEN ss_sales_price ELSE null END)
thu_sales,
        SUM(CASE WHEN (d_day_name='Friday') THEN ss_sales_price ELSE null END)
fri_sales,
        SUM(CASE WHEN (d_day_name='Saturday') THEN ss_sales_price ELSE null END)
sat_sales
 FROM date_dim, store_sales, store
 WHERE d_date_sk = ss_sold_date_sk AND
       s_store_sk = ss_store_sk AND
       s_gmt_offset = -5 AND
       d_year = 2000
 GROUP BY s_store_name, s_store_id
 ORDER BY s_store_name,
s_store_id,sun_sales,mon_sales,tue_sales,wed_sales,thu_sales,fri_sales,sat_sales
  ) LIMIT 100;
```

----Fim

Após a recolha das estatísticas anteriores, a tabela de referência é a seguinte:

Referência	Antes de	Depois
Tempo de carregamento (11 tabelas)	341584 ms	-
Espaço de armazenamento ocupado		
Store_Sales	42 GB	-
Date_Dim	11 MB	-
Store	232 KB	-
Item	110 MB	-
Time_Dim	11 MB	-
Promotion	256 KB	-
Customer_Demographics	171 MB	-
Customer_Address	170 MB	-

Referência	Antes de	Depois
Household_Demographics	504 KB	-
Customer	441 MB	-
Income_Band	88 KB	-
Espaço total de armazenamento	42 GB	-
Tempo de execução da consulta		
Consulta 1	14552,05 ms	-
Consulta 2	27952,36 ms	-
Consulta 3	17721,15 ms	-
Tempo total de execução	60225,56 ms	-

3.6 Etapa 3: otimizar uma tabela

Selecionar um tipo de armazenamento

As tabelas de exemplo usadas nessa prática são típicas tabelas TPC-DS de várias colunas, nas quais muitas consultas de análise estatística são realizadas. Portanto, o modo de armazenamento de coluna é recomendado.

```
WITH (ORIENTATION = column)
```

Selecionar um nível de compressão

Nenhuma taxa de compressão é especificada em **Passo 1: criar uma tabela inicial e carregar dados de amostra**, e a baixa taxa de compressão é selecionada por GaussDB(DWS) por padrão. Especifique **COMPRESSION** para **MIDDLE** e compare o resultado com aquele quando **COMPRESSION** é definido como **LOW**.

Segue-se um exemplo de selecção de um modo de armazenamento e a taxa de compressão **MIDDLE** para uma tabela.

```
CREATE TABLE store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer                not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          bigint                 not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
```



```
    ss_ext_discount_amt       decimal(7,2)                ,
    ss_ext_sales_price        decimal(7,2)                ,
    ss_ext_wholesale_cost     decimal(7,2)                ,
    ss_ext_list_price         decimal(7,2)                ,
    ss_ext_tax                decimal(7,2)                ,
    ss_coupon_amt             decimal(7,2)                ,
    ss_net_paid               decimal(7,2)                ,
    ss_net_paid_inc_tax       decimal(7,2)                ,
    ss_net_profit             decimal(7,2)
)
WITH (ORIENTATION = column,COMPRESSION=middle);
```

Selecionar um modo de distribuição

Com base nos tamanhos de tabela fornecidos em **Passo 2: testar o desempenho do sistema da tabela inicial e estabelecer uma linha de base**, defina o modo de distribuição da seguinte forma.

Nome da tabela	Número de linhas	Modo de distribuição
Store_Sales	287997024	Hash
Date_Dim	73049	Replicação
Store	402	Replicação
item	204000	Replicação
Time_Dim	86400	Replicação
Promotion	1000	Replicação
Customer_Demographics	1920800	Hash
Customer_Address	1000000	Hash
Household_Demographics	7200	Replicação
Customer	1981703	Hash
Income_Band	20	Replicação

Selecionar uma chave de distribuição

Se sua tabela for distribuída usando hash, escolha uma chave de distribuição adequada. É aconselhável selecionar uma chave de distribuição de acordo com **Selecionar uma chave de distribuição**.

Selecione a chave primária de cada tabela como a chave de distribuição da tabela de hash.

Nome da tabela	Número de registros	Modo de distribuição	Chave de distribuição
Store_Sales	287997024	Hash	ss_item_sk

Nome da tabela	Número de registros	Modo de distribuição	Chave de distribuição
Date_Dim	73049	Replicação	-
Store	402	Replicação	-
Item	204000	Replicação	-
Time_Dim	86400	Replicação	-
Promotion	1000	Replicação	-
Customer_Demographics	1920800	Hash	cd_demo_sk
Customer_Address	1000000	Hash	ca_address_sk
Household_Demographics	7200	Replicação	-
Customer	1981703	Hash	c_customer_sk
Income_Band	20	Replicação	-

3.7 Etapa 4: criar outra tabela e carregar dados

Depois de selecionar um modo de armazenamento, nível de compactação, modo de distribuição e chave de distribuição para cada tabela, use esses atributos para criar tabelas e recarregar dados. Compare o desempenho do sistema antes e depois da recriação da mesa.

Passo 1 Exclua as tabelas criadas anteriormente.

```
DROP TABLE store_sales;
DROP TABLE date_dim;
DROP TABLE store;
DROP TABLE item;
DROP TABLE time_dim;
DROP TABLE promotion;
DROP TABLE customer_demographics;
DROP TABLE customer_address;
DROP TABLE household_demographics;
DROP TABLE customer;
DROP TABLE income_band;

DROP FOREIGN TABLE obs_from_store_sales_001;
DROP FOREIGN TABLE obs_from_date_dim_001;
DROP FOREIGN TABLE obs_from_store_001;
DROP FOREIGN TABLE obs_from_item_001;
DROP FOREIGN TABLE obs_from_time_dim_001;
DROP FOREIGN TABLE obs_from_promotion_001;
DROP FOREIGN TABLE obs_from_customer_demographics_001;
DROP FOREIGN TABLE obs_from_customer_address_001;
DROP FOREIGN TABLE obs_from_household_demographics_001;
DROP FOREIGN TABLE obs_from_customer_001;
DROP FOREIGN TABLE obs_from_income_band_001;
```

Passo 2 Crie tabelas e especifique modos de armazenamento e distribuição para elas.

Somente a sintaxe para recriar a tabela **store_sales** é fornecida para simplificar. Para recriar todas as outras tabelas, copie a sintaxe em **Criação de uma outra tabela após a otimização do design**.

```
CREATE TABLE store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer               not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          bigint                not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
    ss_ext_discount_amt       decimal(7,2)                  ,
    ss_ext_sales_price        decimal(7,2)                  ,
    ss_ext_wholesale_cost     decimal(7,2)                  ,
    ss_ext_list_price         decimal(7,2)                  ,
    ss_ext_tax                decimal(7,2)                  ,
    ss_coupon_amt             decimal(7,2)                  ,
    ss_net_paid               decimal(7,2)                  ,
    ss_net_paid_inc_tax       decimal(7,2)                  ,
    ss_net_profit             decimal(7,2)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY hash (ss_item_sk);
```

Passo 3 **Carregue dados de exemplo nessas tabelas.**

Passo 4 Registre o tempo de carregamento nas tabelas de referência.

Referência	Antes de	Depois
Tempo de carregamento (11 tabelas)	341584 ms	257241 ms
Espaço de armazenamento ocupado		
Store_Sales	42 GB	-
Date_Dim	11 MB	-
Store	232 KB	-
Item	110 MB	-
Time_Dim	11 MB	-
Promotion	256 KB	-
Customer_Demographics	171 MB	-
Customer_Address	170 MB	-
Household_Demographics	504 KB	-
Customer	441 MB	-
Income_Band	88 KB	-

Referência	Antes de	Depois
Espaço total de armazenamento	42 GB	-
Tempo de execução da consulta		
Consulta 1	14552,05 ms	-
Consulta 2	27952,36 ms	-
Consulta 3	17721,15 ms	-
Tempo total de execução	60225,56 ms	-

Passo 5 Execute o comando **ANALYZE** para atualizar as estatísticas.

```
ANALYZE;
```

Se **ANALYZE** for retornado, a execução é bem-sucedida.

```
ANALYZE
```

Passo 6 Verifique se há distorção de dados.

Para uma tabela hash, uma chave de distribuição imprópria pode causar distorção de dados ou desempenho ruim de I/O em determinados DNs. Portanto, você precisa verificar a tabela para garantir que os dados sejam distribuídos uniformemente em cada DN. Você pode executar as seguintes instruções SQL para verificar a distorção de dados:

```
SELECT a.count,b.node_name FROM (SELECT count(*) AS count,xc_node_id FROM
table_name GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER
BY a.count desc;
```

xc_node_id corresponde a um DN. Geralmente, **mais de 5% de diferença entre a quantidade de dados em diferentes DNs é considerada como distorção de dados. Se a diferença for superior a 10%, escolha outra chave de distribuição.** No GaussDB(DWS), você pode selecionar várias chaves de distribuição para distribuir os dados uniformemente.

----Fim

3.8 Passo 5: testar o desempenho do sistema na nova tabela

Depois de recriar o conjunto de dados de teste com os modos de armazenamento selecionados, níveis de compactação, modos de distribuição e chaves de distribuição, você testará novamente o desempenho do sistema.

Passo 1 Registre o uso do espaço de armazenamento de cada tabela.

Determine quanto espaço em disco é usado para cada tabela usando a função **pg_size_pretty** e registre os resultados em tabelas básicas.

```
SELECT T_NAME, PG_SIZE_PRETTY(PG_RELATION_SIZE(t_name)) FROM
(VALUES('store_sales'),('date_dim'),('store'),('item'),('time_dim'),('promotion'),
('customer_demographics'),('customer_address'),('household_demographics'),
('customer'),('income_band')) AS names1(t_name);
         t_name         | pg_size_pretty
------------------------+----------------
 store_sales            | 14 GB
 date_dim               | 27 MB
```



```
 store                  | 4352 kB
 item                   | 259 MB
 time_dim               | 14 MB
 promotion              | 3200 kB
 customer_demographics  | 11 MB
 customer_address       | 27 MB
 household_demographics | 1280 kB
 customer               | 111 MB
 income_band            | 896 kB
(11 rows)
```

Passo 2 Teste o desempenho da consulta e registre os dados de desempenho na tabela de referência.

Execute as consultas a seguir novamente e registre o tempo gasto em cada consulta.

```
\timing on
SELECT * FROM (SELECT  COUNT(*)
FROM store_sales
    ,household_demographics
    ,time_dim, store
WHERE ss_sold_time_sk = time_dim.t_time_sk
    AND ss_hdemo_sk = household_demographics.hd_demo_sk
    AND ss_store_sk = s_store_sk
    AND time_dim.t_hour = 8
    AND time_dim.t_minute >= 30
    AND household_demographics.hd_dep_count = 5
    AND store.s_store_name = 'ese'
ORDER BY COUNT(*)
 ) LIMIT 100;

SELECT * FROM (SELECT  i_brand_id brand_id, i_brand brand, i_manufact_id,
i_manufact,
SUM(ss_ext_sales_price) ext_price
 FROM date_dim, store_sales, item,customer,customer_address,store
 WHERE d_date_sk = ss_sold_date_sk
   AND ss_item_sk = i_item_sk
   AND i_manager_id=8
   AND d_moy=11
   AND d_year=1999
   AND ss_customer_sk = c_customer_sk
   AND c_current_addr_sk = ca_address_sk
   AND substr(ca_zip,1,5) <> substr(s_zip,1,5)
   AND ss_store_sk = s_store_sk
 GROUP BY i_brand
      ,i_brand_id
      ,i_manufact_id
      ,i_manufact
 ORDER BY ext_price desc
         ,i_brand
         ,i_brand_id
         ,i_manufact_id
         ,i_manufact
 ) LIMIT 100;

SELECT * FROM (SELECT  s_store_name, s_store_id,
        SUM(CASE WHEN (d_day_name='Sunday') THEN ss_sales_price ELSE null END)
sun_sales,
        SUM(CASE WHEN (d_day_name='Monday') THEN ss_sales_price ELSE null END)
mon_sales,
        SUM(CASE WHEN (d_day_name='Tuesday') THEN ss_sales_price ELSE  null END)
tue_sales,
        SUM(CASE WHEN (d_day_name='Wednesday') THEN ss_sales_price ELSE null END)
wed_sales,
        SUM(CASE WHEN (d_day_name='Thursday') THEN ss_sales_price ELSE null END)
thu_sales,
        SUM(CASE WHEN (d_day_name='Friday') THEN ss_sales_price ELSE null END)
fri_sales,
        SUM(CASE WHEN (d_day_name='Saturday') THEN ss_sales_price ELSE null END)
sat_sales
 FROM date_dim, store_sales, store
```



```
 WHERE d_date_sk = ss_sold_date_sk AND
       s_store_sk = ss_store_sk AND
       s_gmt_offset = -5 AND
       d_year = 2000
 GROUP BY s_store_name, s_store_id
 ORDER BY s_store_name,
s_store_id,sun_sales,mon_sales,tue_sales,wed_sales,thu_sales,fri_sales,sat_sales
  ) LIMIT 100;
```

A tabela de referência a seguir mostra os resultados de validação do cluster usado neste tutorial. Seus resultados podem variar com base em vários fatores, mas os resultados relativos devem ser semelhantes. As durações de execução de consultas com a mesma estrutura de tabela podem ser diferentes, dependendo do cache do sistema operacional durante a execução. É aconselhável realizar várias rodadas de testes e selecionar um grupo com valores médios.

Referência	Antes de	Depois
Tempo de carregamento (11 tabelas)	341584 ms	257241 ms
Espaço de armazenamento ocupado		
Store_Sales	42 GB	14 GB
Date_Dim	11 MB	27 MB
Armazenamento	232 KB	4352 KB
Item	110 MB	259 MB
Time_Dim	11 MB	14 MB
Promotion	256 KB	3200 KB
Customer_Demographics	171 MB	11 MB
Customer_Address	170 MB	27 MB
Household_Demographics	504 KB	1280 KB
Cliente	441 MB	111 MB
Income_Band	88 KB	896 KB
Espaço total de armazenamento	42 GB	15 GB
Tempo de execução da consulta		
Consulta 1	14552,05 ms	1783,353 ms
Consulta 2	27952,36 ms	14247,803 ms
Consulta 3	17721,15 ms	11441,659 ms
Tempo total de execução	60225,56 ms	27472,815 ms

Passo 3 Se você tiver expectativas mais altas para o desempenho após o design da tabela, poderá executar o comando **EXPLAIN PERFORMANCE** para exibir o plano de execução para ajuste.

Para obter mais detalhes sobre planos de execução e ajuste de consulta, consulte **Plano de execução SQL** e **Visão geral do ajuste de desempenho de consultas**.

----Fim

3.9 Passo 6: avaliar o desempenho da tabela otimizada

Compare o tempo de carregamento, o uso do espaço de armazenamento e o tempo de execução da consulta antes e depois do ajuste da tabela.

A tabela a seguir mostra os resultados de exemplo do cluster usado neste tutorial. Seus resultados serão diferentes, mas devem mostrar melhorias semelhantes.

Referência	Antes de	Depois	Alteração	Porcentagem (%)
Tempo de carregamento (11 mesas)	341584 ms	257241 ms	-84343 ms	-24,7%
Espaço de armazenamento ocupado			-	-
Store_Sales	42 GB	14 GB	-28 GB	-66,7%
Date_Dim	11 MB	27 MB	16 MB	145,5%
Store	232 KB	4352 KB	4120 KB	1775,9%
Item	110 MB	259 MB	149 MB	1354,5%
Time_Dim	11 MB	14 MB	13 MB	118,2%
Promotion	256 KB	3200 KB	2944 KB	1150%
Customer_Demographics	171 MB	11 MB	-160 MB	-93,6
Customer_Address	170 MB	27 MB	-143 MB	-84,1%
Household_Demographics	504 KB	1280 KB	704 KB	139,7%
Customer	441 MB	111 MB	-330 MB	-74,8%
Income_Band	88 KB	896 KB	808 KB	918,2%
Espaço total de armazenamento	42 GB	15 GB	-27 GB	-64,3%
Tempo de execução da consulta			-	-
Consulta 1	14552,05 ms	1783,353 ms	-12768,697 ms	-87,7%
Consulta 2	27952,36 ms	14247,803 ms	-13704,557 ms	-49,0%

Referência	Antes de	Depois	Alteração	Porcentagem (%)
Consulta 3	17721,15 ms	11441,659 ms	-6279,491 ms	-35,4%
Tempo total de execução	60225,56 ms	27472,815 ms	-32752,745 ms	-54,4%

Avaliar a tabela após a otimização

- O tempo de carregamento foi reduzido em 24,7%.

  O modo de distribuição tem um impacto óbvio no carregamento de dados. O modo de distribuição de hash melhora a eficiência de carregamento. O modo de distribuição de replicação reduz a eficiência de carregamento. Quando a CPU e a I/O são suficientes, o nível de compactação tem pouco impacto na eficiência de carregamento. Normalmente, a eficiência de carregar uma tabela de armazenamento de colunas é maior do que a de uma tabela de armazenamento de linhas.

- O espaço de uso de armazenamento foi reduzido em 64,3%.

  O nível de compressão, o armazenamento da coluna e a distribuição de hash podem economizar o espaço de armazenamento. Uma tabela de replicação aumenta o uso do armazenamento, mas reduz a sobrecarga da rede. Usar o modo de replicação para pequenas tabelas é uma maneira positiva de usar pequeno espaço para desempenho.

- O desempenho da consulta (velocidade) aumentou 54,4%, indicando que o tempo de consulta diminuiu 54,4%.

  O desempenho da consulta é melhorado pela otimização dos modos de armazenamento, modos de distribuição e chaves de distribuição. Em uma consulta de análise estatística em tabelas de várias colunas, o armazenamento de colunas pode melhorar o desempenho da consulta. Em uma tabela de hash, os recursos de I/O em cada nó podem ser usados durante a leitura/gravação de I/O, o que melhora a velocidade de leitura/gravação de uma tabela.

  Muitas vezes, o desempenho da consulta pode ser melhorado ainda mais reescrevendo consultas e configurando o gerenciamento de carga de trabalho (WLM). Para obter mais informações, consulte **Visão geral da otimização de desempenho de consulta**.

Você pode adaptar as operações em **Práticas da otimização de tabela** para melhorar ainda mais a distribuição de tabelas e o desempenho do carregamento, armazenamento e consulta de dados.

Excluir recursos

Após concluir o exercício, exclua o cluster consultando **Exclusão de um cluster**.

Se você quiser manter o cluster, mas excluir o espaço de armazenamento usado pelas tabelas SS, execute os seguintes comandos:

```
DROP TABLE store_sales;
DROP TABLE date_dim;
DROP TABLE store;
DROP TABLE item;
DROP TABLE time_dim;
DROP TABLE promotion;
DROP TABLE customer_demographics;
DROP TABLE customer_address;
```



```
DROP TABLE household_demographics;
DROP TABLE customer;
DROP TABLE income_band;
```

3.10 Apêndice: sintaxe de criação de tabela

3.10.1 Uso

Esta seção fornece instruções de teste de SQL usadas neste tutorial. É aconselhável copiar as instruções SQL em cada seção e salvá-las como um arquivo .sql. Por exemplo, crie um arquivo chamado **create_table_fir.sql** e cole as instruções SQL na seção **Criação de uma tabela inicial** no arquivo. A execução do arquivo em uma ferramenta cliente de SQL é eficiente e o tempo total decorrido dos casos de teste é fácil de calcular. Execute o arquivo **.sql** usando **gsql** da seguinte maneira:

```
gsql -d database_name -h dws_ip -U username -p port_number -W password -f XXX.sql
```

Substitua as partes em itálico no exemplo por valores reais em GaussDB(DWS). Por exemplo:

```
gsql -d postgres -h 10.10.0.1 -U dbadmin -p 8000 -W password -f
create_table_fir.sql
```

Substitua as seguintes informações no exemplo com base nos requisitos do site:

- **postgres**: indica o nome do banco de dados a ser conectado.
- **10.10.0.1**: endereço de conexão do cluster.
- **dbadmin**: nome do usuário do banco de dados do cluster. O administrador padrão é **dbadmin**.
- **8000**: porta do banco de dados definida durante a criação do cluster.
- **password**: senha definida durante a criação do cluster.

3.10.2 Criação de uma tabela inicial

Esta seção contém a sintaxe de criação de tabela usada quando você cria uma tabela pela primeira vez neste tutorial.

As tabelas são criadas sem especificar seus modos de armazenamento, chaves de distribuição, modos de distribuição ou modos de compactação.

```
CREATE TABLE store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer                 not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          bigint                  not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
    ss_ext_discount_amt       decimal(7,2)                  ,
    ss_ext_sales_price        decimal(7,2)                  ,
    ss_ext_wholesale_cost     decimal(7,2)                  ,
    ss_ext_list_price         decimal(7,2)                  ,
```



```
    ss_ext_tax                decimal(7,2)                  ,
    ss_coupon_amt             decimal(7,2)                  ,
    ss_net_paid               decimal(7,2)                  ,
    ss_net_paid_inc_tax       decimal(7,2)                  ,
    ss_net_profit             decimal(7,2)
) ;

CREATE TABLE date_dim
(
    d_date_sk                 integer               not null,
    d_date_id                 char(16)              not null,
    d_date                    date                          ,
    d_month_seq               integer                       ,
    d_week_seq                integer                       ,
    d_quarter_seq             integer                       ,
    d_year                    integer                       ,
    d_dow                     integer                       ,
    d_moy                     integer                       ,
    d_dom                     integer                       ,
    d_qoy                     integer                       ,
    d_fy_year                 integer                       ,
    d_fy_quarter_seq          integer                       ,
    d_fy_week_seq             integer                       ,
    d_day_name                char(9)                       ,
    d_quarter_name            char(6)                       ,
    d_holiday                 char(1)                       ,
    d_weekend                 char(1)                       ,
    d_following_holiday       char(1)                       ,
    d_first_dom               integer                       ,
    d_last_dom                integer                       ,
    d_same_day_ly             integer                       ,
    d_same_day_lq             integer                       ,
    d_current_day             char(1)                       ,
    d_current_week            char(1)                       ,
    d_current_month           char(1)                       ,
    d_current_quarter         char(1)                       ,
    d_current_year            char(1)
) ;

CREATE TABLE store
(
    s_store_sk                integer               not null,
    s_store_id                char(16)              not null,
    s_rec_start_date          date                          ,
    s_rec_end_date            date                          ,
    s_closed_date_sk          integer                       ,
    s_store_name              varchar(50)                   ,
    s_number_employees        integer                       ,
    s_floor_space             integer                       ,
    s_hours                   char(20)                      ,
    s_manager                 varchar(40)                   ,
    s_market_id               integer                       ,
    s_geography_class         varchar(100)                  ,
    s_market_desc             varchar(100)                  ,
    s_market_manager          varchar(40)                   ,
    s_division_id             integer                       ,
    s_division_name           varchar(50)                   ,
    s_company_id              integer                       ,
    s_company_name            varchar(50)                   ,
    s_street_number           varchar(10)                   ,
    s_street_name             varchar(60)                   ,
    s_street_type             char(15)                      ,
    s_suite_number            char(10)                      ,
    s_city                    varchar(60)                   ,
    s_county                  varchar(30)                   ,
    s_state                   char(2)                       ,
    s_zip                     char(10)                      ,
    s_country                 varchar(20)                   ,
    s_gmt_offset              decimal(5,2)                  ,
```



```
    s_tax_precentage          decimal(5,2)
) ;

CREATE TABLE item
(
    i_item_sk                 integer               not null,
    i_item_id                 char(16)              not null,
    i_rec_start_date          date                        ,
    i_rec_end_date            date                        ,
    i_item_desc               varchar(200)                ,
    i_current_price           decimal(7,2)                ,
    i_wholesale_cost          decimal(7,2)                ,
    i_brand_id                integer                     ,
    i_brand                   char(50)                    ,
    i_class_id                integer                     ,
    i_class                   char(50)                    ,
    i_category_id             integer                     ,
    i_category                char(50)                    ,
    i_manufact_id             integer                     ,
    i_manufact                char(50)                    ,
    i_size                    char(20)                    ,
    i_formulation             char(20)                    ,
    i_color                   char(20)                    ,
    i_units                   char(10)                    ,
    i_container               char(10)                    ,
    i_manager_id              integer                     ,
    i_product_name            char(50)
) ;

CREATE TABLE time_dim
(
    t_time_sk                 integer               not null,
    t_time_id                 char(16)              not null,
    t_time                    integer                     ,
    t_hour                    integer                     ,
    t_minute                  integer                     ,
    t_second                  integer                     ,
    t_am_pm                   char(2)                     ,
    t_shift                   char(20)                    ,
    t_sub_shift               char(20)                    ,
    t_meal_time               char(20)
) ;

CREATE TABLE promotion
(
    p_promo_sk                integer               not null,
    p_promo_id                char(16)              not null,
    p_start_date_sk           integer                     ,
    p_end_date_sk             integer                     ,
    p_item_sk                 integer                     ,
    p_cost                    decimal(15,2)               ,
    p_response_target         integer                     ,
    p_promo_name              char(50)                    ,
    p_channel_dmail           char(1)                     ,
    p_channel_email           char(1)                     ,
    p_channel_catalog         char(1)                     ,
    p_channel_tv              char(1)                     ,
    p_channel_radio           char(1)                     ,
    p_channel_press           char(1)                     ,
    p_channel_event           char(1)                     ,
    p_channel_demo            char(1)                     ,
    p_channel_details         varchar(100)                ,
    p_purpose                 char(15)                    ,
    p_discount_active         char(1)
) ;

CREATE TABLE customer_demographics
(
    cd_demo_sk                integer               not null,
```



```
    cd_gender                 char(1)                       ,
    cd_marital_status         char(1)                       ,
    cd_education_status       char(20)                      ,
    cd_purchase_estimate      integer                       ,
    cd_credit_rating          char(10)                      ,
    cd_dep_count              integer                       ,
    cd_dep_employed_count     integer                       ,
    cd_dep_college_count      integer
) ;

CREATE TABLE customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(5,2)                  ,
    ca_location_type          char(20)
) ;

CREATE TABLE household_demographics
(
    hd_demo_sk                integer               not null,
    hd_income_band_sk         integer                       ,
    hd_buy_potential          char(15)                      ,
    hd_dep_count              integer                       ,
    hd_vehicle_count          integer
) ;

CREATE TABLE customer
(
    c_customer_sk             integer               not null,
    c_customer_id             char(16)              not null,
    c_current_cdemo_sk        integer                       ,
    c_current_hdemo_sk        integer                       ,
    c_current_addr_sk         integer                       ,
    c_first_shipto_date_sk    integer                       ,
    c_first_sales_date_sk     integer                       ,
    c_salutation              char(10)                      ,
    c_first_name              char(20)                      ,
    c_last_name               char(30)                      ,
    c_preferred_cust_flag     char(1)                       ,
    c_birth_day               integer                       ,
    c_birth_month             integer                       ,
    c_birth_year              integer                       ,
    c_birth_country           varchar(20)                   ,
    c_login                   char(13)                      ,
    c_email_address           char(50)                      ,
    c_last_review_date        char(10)
) ;

CREATE TABLE income_band
(
    ib_income_band_sk         integer               not null,
    ib_lower_bound            integer                       ,
    ib_upper_bound            integer
) ;
```

3.10.3 Criação de uma outra tabela após a otimização do design

Esta seção contém a sintaxe de criação de outra tabela após os modos de armazenamento,

níveis de compactação, modos de distribuição e chaves de distribuição serem selecionados nesta prática.

```
CREATE TABLE store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer               not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          bigint               not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
    ss_ext_discount_amt       decimal(7,2)                  ,
    ss_ext_sales_price        decimal(7,2)                  ,
    ss_ext_wholesale_cost     decimal(7,2)                  ,
    ss_ext_list_price         decimal(7,2)                  ,
    ss_ext_tax                decimal(7,2)                  ,
    ss_coupon_amt             decimal(7,2)                  ,
    ss_net_paid               decimal(7,2)                  ,
    ss_net_paid_inc_tax       decimal(7,2)                  ,
    ss_net_profit             decimal(7,2)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY hash (ss_item_sk);

CREATE TABLE date_dim
(
    d_date_sk                 integer               not null,
    d_date_id                 char(16)              not null,
    d_date                    date                          ,
    d_month_seq               integer                       ,
    d_week_seq                integer                       ,
    d_quarter_seq             integer                       ,
    d_year                    integer                       ,
    d_dow                     integer                       ,
    d_moy                     integer                       ,
    d_dom                     integer                       ,
    d_qoy                     integer                       ,
    d_fy_year                 integer                       ,
    d_fy_quarter_seq          integer                       ,
    d_fy_week_seq             integer                       ,
    d_day_name                char(9)                       ,
    d_quarter_name            char(6)                       ,
    d_holiday                 char(1)                       ,
    d_weekend                 char(1)                       ,
    d_following_holiday       char(1)                       ,
    d_first_dom               integer                       ,
    d_last_dom                integer                       ,
    d_same_day_ly             integer                       ,
    d_same_day_lq             integer                       ,
    d_current_day             char(1)                       ,
    d_current_week            char(1)                       ,
    d_current_month           char(1)                       ,
    d_current_quarter         char(1)                       ,
    d_current_year            char(1)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY replication;

CREATE TABLE store
(
    s_store_sk                integer               not null,
```



```
    s_store_id                char(16)              not null,
    s_rec_start_date          date                          ,
    s_rec_end_date            date                          ,
    s_closed_date_sk          integer                       ,
    s_store_name              varchar(50)                   ,
    s_number_employees        integer                       ,
    s_floor_space             integer                       ,
    s_hours                   char(20)                      ,
    s_manager                 varchar(40)                   ,
    s_market_id               integer                       ,
    s_geography_class         varchar(100)                  ,
    s_market_desc             varchar(100)                  ,
    s_market_manager          varchar(40)                   ,
    s_division_id             integer                       ,
    s_division_name           varchar(50)                   ,
    s_company_id              integer                       ,
    s_company_name            varchar(50)                   ,
    s_street_number           varchar(10)                   ,
    s_street_name             varchar(60)                   ,
    s_street_type             char(15)                      ,
    s_suite_number            char(10)                      ,
    s_city                    varchar(60)                   ,
    s_county                  varchar(30)                   ,
    s_state                   char(2)                       ,
    s_zip                     char(10)                      ,
    s_country                 varchar(20)                   ,
    s_gmt_offset              decimal(5,2)                  ,
    s_tax_precentage          decimal(5,2)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY replication;

CREATE TABLE item
(
    i_item_sk                 integer               not null,
    i_item_id                 char(16)              not null,
    i_rec_start_date          date                          ,
    i_rec_end_date            date                          ,
    i_item_desc               varchar(200)                  ,
    i_current_price           decimal(7,2)                  ,
    i_wholesale_cost          decimal(7,2)                  ,
    i_brand_id                integer                       ,
    i_brand                   char(50)                      ,
    i_class_id                integer                       ,
    i_class                   char(50)                      ,
    i_category_id             integer                       ,
    i_category                char(50)                      ,
    i_manufact_id             integer                       ,
    i_manufact                char(50)                      ,
    i_size                    char(20)                      ,
    i_formulation             char(20)                      ,
    i_color                   char(20)                      ,
    i_units                   char(10)                      ,
    i_container               char(10)                      ,
    i_manager_id              integer                       ,
    i_product_name            char(50)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY replication;

CREATE TABLE time_dim
(
    t_time_sk                 integer               not null,
    t_time_id                 char(16)              not null,
    t_time                    integer                       ,
    t_hour                    integer                       ,
    t_minute                  integer                       ,
    t_second                  integer                       ,
    t_am_pm                   char(2)                       ,
```



```
    t_shift                   char(20)                      ,
    t_sub_shift               char(20)                      ,
    t_meal_time               char(20)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY replication;

CREATE TABLE promotion
(
    p_promo_sk                integer               not null,
    p_promo_id                char(16)              not null,
    p_start_date_sk           integer                       ,
    p_end_date_sk             integer                       ,
    p_item_sk                 integer                       ,
    p_cost                    decimal(15,2)                 ,
    p_response_target         integer                       ,
    p_promo_name              char(50)                      ,
    p_channel_dmail           char(1)                       ,
    p_channel_email           char(1)                       ,
    p_channel_catalog         char(1)                       ,
    p_channel_tv              char(1)                       ,
    p_channel_radio           char(1)                       ,
    p_channel_press           char(1)                       ,
    p_channel_event           char(1)                       ,
    p_channel_demo            char(1)                       ,
    p_channel_details         varchar(100)                  ,
    p_purpose                 char(15)                      ,
    p_discount_active         char(1)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY replication;

CREATE TABLE customer_demographics
(
    cd_demo_sk                integer               not null,
    cd_gender                 char(1)                       ,
    cd_marital_status         char(1)                       ,
    cd_education_status       char(20)                      ,
    cd_purchase_estimate      integer                       ,
    cd_credit_rating          char(10)                      ,
    cd_dep_count              integer                       ,
    cd_dep_employed_count     integer                       ,
    cd_dep_college_count      integer
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY hash (cd_demo_sk);

CREATE TABLE customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(5,2)                  ,
    ca_location_type          char(20)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY hash (ca_address_sk);

CREATE TABLE household_demographics
(
    hd_demo_sk                integer               not null,
```



```
    hd_income_band_sk         integer                       ,
    hd_buy_potential          char(15)                      ,
    hd_dep_count              integer                       ,
    hd_vehicle_count          integer
)
 WITH (ORIENTATION = column,COMPRESSION=middle)
 DISTRIBUTE BY replication;

 CREATE TABLE customer
(
    c_customer_sk             integer               not null,
    c_customer_id             char(16)              not null,
    c_current_cdemo_sk        integer                       ,
    c_current_hdemo_sk        integer                       ,
    c_current_addr_sk         integer                       ,
    c_first_shipto_date_sk    integer                       ,
    c_first_sales_date_sk     integer                       ,
    c_salutation              char(10)                      ,
    c_first_name              char(20)                      ,
    c_last_name               char(30)                      ,
    c_preferred_cust_flag     char(1)                       ,
    c_birth_day               integer                       ,
    c_birth_month             integer                       ,
    c_birth_year              integer                       ,
    c_birth_country           varchar(20)                   ,
    c_login                   char(13)                      ,
    c_email_address           char(50)                      ,
    c_last_review_date        char(10)
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY hash (c_customer_sk);

CREATE TABLE income_band
(
    ib_income_band_sk         integer               not null,
    ib_lower_bound            integer                       ,
    ib_upper_bound            integer
)
WITH (ORIENTATION = column,COMPRESSION=middle)
DISTRIBUTE BY replication;
```

3.10.4 Criação de uma tabela estrangeira

Esta seção contém a sintaxe de tabelas estrangeiras para obter dados de exemplo usados neste tutorial. Os dados de amostra são armazenados em um bucket do OBS acessível a todos os usuários da nuvem autenticados.

NOTA

- Observe que <*obs_bucket_name*> na instrução a seguir indica o nome do bucket do OBS. Apenas algumas regiões são suportadas. Para obter detalhes sobre as regiões suportadas e os nomes dos bucket do OBS, consulte **Regiões suportadas**. Os clusters do GaussDB(DWS) não oferecem suporte ao acesso entre regiões aos dados do bucket do OBS.
- Você pode substituir **ACCESS_KEY** e **SECRET_ACCESS_KEY** por suas próprias credenciais neste exemplo.
- Quando uma tabela estrangeira do OBS é criada, somente a relação de mapeamento é criada e os dados não são extraídos para o disco do GaussDB (DWS).

```
CREATE FOREIGN TABLE obs_from_store_sales_001
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer               not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
```



```
    ss_addr_sk               integer                       ,
    ss_store_sk              integer                       ,
    ss_promo_sk              integer                       ,
    ss_ticket_number         bigint                not null,
    ss_quantity              integer                       ,
    ss_wholesale_cost        decimal(7,2)                  ,
    ss_list_price            decimal(7,2)                  ,
    ss_sales_price           decimal(7,2)                  ,
    ss_ext_discount_amt      decimal(7,2)                  ,
    ss_ext_sales_price       decimal(7,2)                  ,
    ss_ext_wholesale_cost    decimal(7,2)                  ,
    ss_ext_list_price        decimal(7,2)                  ,
    ss_ext_tax               decimal(7,2)                  ,
    ss_coupon_amt            decimal(7,2)                  ,
    ss_net_paid              decimal(7,2)                  ,
    ss_net_paid_inc_tax      decimal(7,2)                  ,
    ss_net_profit            decimal(7,2)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/store_sales',
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_store_sales_001;

CREATE FOREIGN TABLE obs_from_date_dim_001
(
    d_date_sk                integer               not null,
    d_date_id                char(16)              not null,
    d_date                   date                          ,
    d_month_seq              integer                       ,
    d_week_seq               integer                       ,
    d_quarter_seq            integer                       ,
    d_year                   integer                       ,
    d_dow                    integer                       ,
    d_moy                    integer                       ,
    d_dom                    integer                       ,
    d_qoy                    integer                       ,
    d_fy_year                integer                       ,
    d_fy_quarter_seq         integer                       ,
    d_fy_week_seq            integer                       ,
    d_day_name               char(9)                       ,
    d_quarter_name           char(6)                       ,
    d_holiday                char(1)                       ,
    d_weekend                char(1)                       ,
    d_following_holiday      char(1)                       ,
    d_first_dom              integer                       ,
    d_last_dom               integer                       ,
    d_same_day_ly            integer                       ,
    d_same_day_lq            integer                       ,
    d_current_day            char(1)                       ,
    d_current_week           char(1)                       ,
    d_current_month          char(1)                       ,
    d_current_quarter        char(1)                       ,
    d_current_year           char(1)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/date_dim' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
```



```
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_date_dim_001;

CREATE FOREIGN TABLE obs_from_store_001
(
    s_store_sk               integer               not null,
    s_store_id               char(16)              not null,
    s_rec_start_date         date                        ,
    s_rec_end_date           date                        ,
    s_closed_date_sk         integer                     ,
    s_store_name             varchar(50)                 ,
    s_number_employees       integer                     ,
    s_floor_space            integer                     ,
    s_hours                  char(20)                    ,
    s_manager                varchar(40)                 ,
    s_market_id              integer                     ,
    s_geography_class        varchar(100)                ,
    s_market_desc            varchar(100)                ,
    s_market_manager         varchar(40)                 ,
    s_division_id            integer                     ,
    s_division_name          varchar(50)                 ,
    s_company_id             integer                     ,
    s_company_name           varchar(50)                 ,
    s_street_number          varchar(10)                 ,
    s_street_name            varchar(60)                 ,
    s_street_type            char(15)                    ,
    s_suite_number           char(10)                    ,
    s_city                   varchar(60)                 ,
    s_county                 varchar(30)                 ,
    s_state                  char(2)                     ,
    s_zip                    char(10)                    ,
    s_country                varchar(20)                 ,
    s_gmt_offset             decimal(5,2)                ,
    s_tax_precentage         decimal(5,2)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/store' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_store_001;

CREATE FOREIGN TABLE obs_from_item_001
(
    i_item_sk                integer               not null,
    i_item_id                char(16)              not null,
    i_rec_start_date         date                        ,
    i_rec_end_date           date                        ,
    i_item_desc              varchar(200)                ,
    i_current_price          decimal(7,2)                ,
    i_wholesale_cost         decimal(7,2)                ,
    i_brand_id               integer                     ,
    i_brand                  char(50)                    ,
    i_class_id               integer                     ,
    i_class                  char(50)                    ,
    i_category_id            integer                     ,
    i_category               char(50)                    ,
```



```
    i_manufact_id             integer                       ,
    i_manufact                char(50)                      ,
    i_size                    char(20)                      ,
    i_formulation             char(20)                      ,
    i_color                   char(20)                      ,
    i_units                   char(10)                      ,
    i_container               char(10)                      ,
    i_manager_id              integer                       ,
    i_product_name            char(50)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/item' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_item_001;

CREATE FOREIGN TABLE obs_from_time_dim_001
(
    t_time_sk                 integer               not null,
    t_time_id                 char(16)              not null,
    t_time                    integer                       ,
    t_hour                    integer                       ,
    t_minute                  integer                       ,
    t_second                  integer                       ,
    t_am_pm                   char(2)                       ,
    t_shift                   char(20)                      ,
    t_sub_shift               char(20)                      ,
    t_meal_time               char(20)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/time_dim' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_time_dim_001;

CREATE FOREIGN TABLE obs_from_promotion_001
(
    p_promo_sk                integer               not null,
    p_promo_id                char(16)              not null,
    p_start_date_sk           integer                       ,
    p_end_date_sk             integer                       ,
    p_item_sk                 integer                       ,
    p_cost                    decimal(15,2)                 ,
    p_response_target         integer                       ,
    p_promo_name              char(50)                      ,
    p_channel_dmail           char(1)                       ,
    p_channel_email           char(1)                       ,
    p_channel_catalog         char(1)                       ,
    p_channel_tv              char(1)                       ,
    p_channel_radio           char(1)                       ,
    p_channel_press           char(1)                       ,
    p_channel_event           char(1)                       ,
    p_channel_demo            char(1)                       ,
```



```
    p_channel_details         varchar(100)                ,
    p_purpose                 char(15)                    ,
    p_discount_active         char(1)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/promotion' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_promotion_001;

CREATE FOREIGN TABLE obs_from_customer_demographics_001
(
    cd_demo_sk                integer               not null,
    cd_gender                 char(1)                       ,
    cd_marital_status         char(1)                       ,
    cd_education_status       char(20)                      ,
    cd_purchase_estimate      integer                       ,
    cd_credit_rating          char(10)                      ,
    cd_dep_count              integer                       ,
    cd_dep_employed_count     integer                       ,
    cd_dep_college_count      integer
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/customer_demographics' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_customer_demographics_001;

CREATE FOREIGN TABLE obs_from_customer_address_001
(
ca_address_sk integer not null,
ca_address_id char(16) not null,
ca_street_number char(10) ,
ca_street_name varchar(60) ,
ca_street_type char(15) ,
ca_suite_number char(10) ,
ca_city varchar(60) ,
ca_county varchar(30) ,
ca_state char(2) ,
ca_zip char(10) ,
ca_country varchar(20) ,
ca_gmt_offset float4 ,
ca_location_type char(20)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/customer_address' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
```



```
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_customer_address_001;

CREATE FOREIGN TABLE obs_from_household_demographics_001
(
    hd_demo_sk               integer               not null,
    hd_income_band_sk        integer                       ,
    hd_buy_potential         char(15)                      ,
    hd_dep_count             integer                       ,
    hd_vehicle_count         integer
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/household_demographics' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_household_demographics_001;

CREATE FOREIGN TABLE obs_from_customer_001
(
    c_customer_sk            integer               not null,
    c_customer_id            char(16)              not null,
    c_current_cdemo_sk       integer                       ,
    c_current_hdemo_sk       integer                       ,
    c_current_addr_sk        integer                       ,
    c_first_shipto_date_sk   integer                       ,
    c_first_sales_date_sk    integer                       ,
    c_salutation             char(10)                      ,
    c_first_name             char(20)                      ,
    c_last_name              char(30)                      ,
    c_preferred_cust_flag    char(1)                       ,
    c_birth_day              integer                       ,
    c_birth_month            integer                       ,
    c_birth_year             integer                       ,
    c_birth_country          varchar(20)                   ,
    c_login                  char(13)                      ,
    c_email_address          char(50)                      ,
    c_last_review_date       char(10)
)
SERVER gsmpp_server
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/customer' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_customer_001;

CREATE FOREIGN TABLE obs_from_income_band_001
(
    ib_income_band_sk        integer               not null,
    ib_lower_bound           integer                       ,
    ib_upper_bound           integer
)
SERVER gsmpp_server
```



```
OPTIONS (
LOCATION 'obs://<obs_bucket_name>/tpcds/income_band' ,
FORMAT 'text',
DELIMITER '|',
ENCODING 'utf8',
NOESCAPING 'true',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
REJECT_LIMIT 'unlimited',
CHUNKSIZE '64'
)
WITH err_obs_from_income_band_001;
```


4 Recursos avançados

4.1 Criação de uma tabela de séries temporais

Cenários

Tabelas de séries temporais herdam a sintaxe de tabelas comuns de coluna-armazenamento e linha-armazenamento, facilitando a compreensão e uso.

As tabelas de séries temporais podem ser gerenciadas por meio do ciclo de vida dos dados. Os dados aumentam explosivamente todos os dias com muitas dimensões. Novas partições precisam ser adicionadas à tabela periodicamente para armazenar novos dados. Os dados gerados há muito tempo geralmente são de baixo valor e não são acessados com frequência. Portanto, pode ser excluído periodicamente. Portanto, as tabelas de séries temporais devem ter a capacidade de adicionar e excluir periodicamente partições.

Esta prática demonstra como criar rapidamente suas tabelas de séries temporais e gerenciá-las por partições. Especificar um tipo adequado para uma coluna ajuda a melhorar o desempenho de operações como importação e consulta, tornando seu serviço mais eficiente. A figura a seguir usa a amostragem de dados do conjunto de gerador como exemplo.

Figura 4-1 Amostra de dados do conjunto de gerador

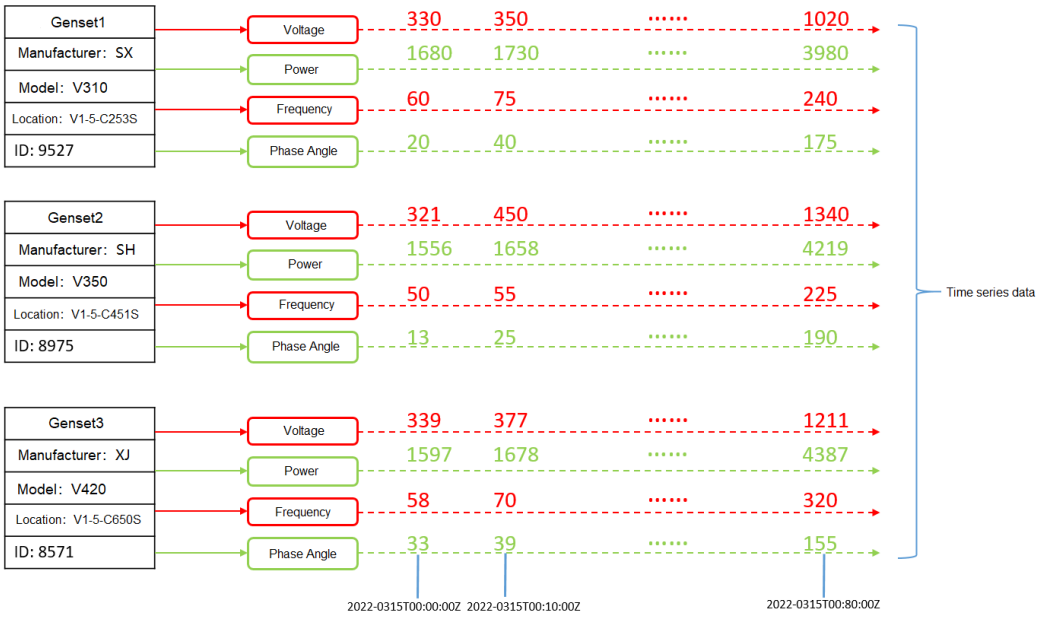
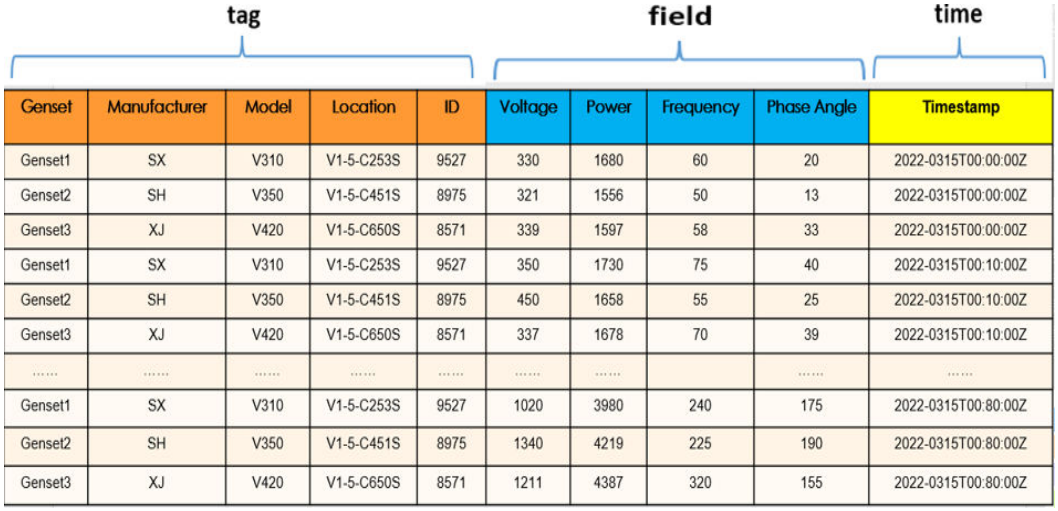


Figura 4-2 Tabela de dados do conjunto de gerador



Genset	Manufacturer	Model	Location	ID	Voltage	Power	Frequency	Phase Angle	Timestamp
Genset1	SX	V310	V1-5-C253S	9527	330	1680	60	20	2022-0315T00:00:00Z
Genset2	SH	V350	V1-5-C451S	8975	321	1556	50	13	2022-0315T00:00:00Z
Genset3	XJ	V420	V1-5-C650S	8571	339	1597	58	33	2022-0315T00:00:00Z
Genset1	SX	V310	V1-5-C253S	9527	350	1730	75	40	2022-0315T00:10:00Z
Genset2	SH	V350	V1-5-C451S	8975	450	1658	55	25	2022-0315T00:10:00Z
Genset3	XJ	V420	V1-5-C650S	8571	337	1678	70	39	2022-0315T00:10:00Z
·····	·····	·····	·····	·····	·····	·····		·····	·····
Genset1	SX	V310	V1-5-C253S	9527	1020	3980	240	175	2022-0315T00:80:00Z
Genset2	SH	V350	V1-5-C451S	8975	1340	4219	225	190	2022-0315T00:80:00Z
Genset3	XJ	V420	V1-5-C650S	8571	1211	4387	320	155	2022-0315T00:80:00Z

- As colunas que descrevem atributos do gerador (informações do gerador, fabricante, modelo, localização e ID) são definidas como colunas de tag. Durante a criação da tabela, elas são especificadas como **TSTag**
- Os valores das métricas de dados de amostragem (tensão, potência, frequência e ângulo de fase atual) variam com o tempo. Durante a criação da tabela, eles são especificados como **TSField**.
- A última coluna é especificada como a coluna de tempo, que armazena as informações de tempo correspondentes aos dados nas colunas de campo. Durante a criação da tabela, ela é especificada como **TSTime**.

Procedimento

Essa prática leva cerca de 30 minutos. O processo básico é o seguinte:

1. **Criar um ECS**.
2. **Criar um armazém de dados de fluxo**.
3. **Usar o cliente de CLI gsql para conectar-se a um cluster**.
4. **Criação de uma tabela de séries temporais**.

Criar um ECS

Para obter detalhes, consulte **Compra de um ECS**. Após a compra de um ECS, faça logon no ECS consultando **Efetuar logon em um ECS de Linux**.

AVISO

Ao criar um ECS, certifique-se de que o ECS esteja na mesma região, AZ e sub-rede da VPC que o armazém de dados de fluxo. Selecione o SO usado pelo cliente gsql (o CentOS 7.6 é usado como um exemplo) como o SO do ECS e selecione usar senhas para fazer logon.

Criar um armazém de dados de fluxo

Passo 1 Efetue logon no console de gerenciamento da Huawei Cloud.

Passo 2 Escolha **Service List** > **Analytics** > **Data Warehouse Service**. Na página exibida, clique em **Create Cluster** no canto superior direito.

Passo 3 Configure parâmetros de acordo com **Tabela 4-1**.

Tabela 4-1 Configuração de software

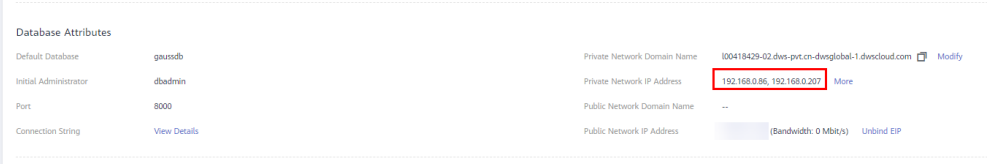
Parâmetro	Configuração
Region	Selecione **CN-Hong Kong**. **NOTA** ● **CN-Hong Kong** é usada como exemplo. Você pode selecionar outras regiões, conforme necessário. Certifique-se de que todas as operações sejam realizadas na mesma região. ● Verifique se o GaussDB(DWS) e o ECS estão na mesma região, AZ e sub-rede da VPC.
AZ	AZ2
Product	Stream data warehouse
Compute Resource	ECS
Storage Type	Cloud SSD
CPU Architecture	x86

Parâmetro	Configuração
Node Flavor	dwsx2.rt.2xlarge.m6 (8 vCPU | 64GB | 100-4,000 GB SSD) **NOTA** Se esse flavor estiver esgotado, selecione outras AZs ou flavors.
Hot Storage	200 GB/node
Nodes	3
Cluster Name	dws-demo01
Administrator Account	dbadmin
Administrator Password	*User-defined*
Confirm Password	Digite a senha de administrador definida pelo usuário novamente.
Database Port	8000
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24) **AVISO** Verifique se o cluster e o ECS estão na mesma sub-rede da VPC.
Security Group	Automatic creation
EIP	Buy now
Enterprise Project	default
Advanced settings	Default

Passo 4 Confirme as informações, clique em **Next** e, em seguida, clique em **Submit**.

Passo 5 Aguarde cerca de 10 minutos. Depois que o cluster for criado, clique no nome do cluster para ir para a página **Basic Information**. Escolha **Network**, clique em um nome de grupo de segurança e verifique se uma regra de grupo de segurança foi adicionada. Neste exemplo, o endereço IP do cliente é 192.168.0.*x* (o endereço IP da rede privada do ECS onde o gsql está localizado é 192.168.0.90). Portanto, você precisa adicionar uma regra de grupo de segurança na qual o endereço IP é 192.168.0.0/24 e o número da porta é 8000.

Passo 6 Retorne à guia **Basic Information** do cluster e registre o valor de **Private Network IP Address**.



----Fim

Usar o cliente de CLI gsql para conectar-se a um cluster

Passo 1 Faça logon remotamente no servidor Linux onde o gsql deve ser instalado como usuário **root** e execute o seguinte comando na janela de comando do Linux para fazer o download do cliente gsql:

```
wget https://obs.ap-southeast-1.myhuaweicloud.com/dws/download/
dws_client_8.1.x_redhat_x64.zip --no-check-certificate
```

Passo 2 Descompacte o cliente.

```
cd <Path_for_storing_the_client> unzip dws_client_8.1.x_redhat_x64.zip
```

Onde,

- *<Path_for_storing_the_client>*: Substitua-o pelo caminho real.
- *dws_client_8.1.x_redhat_x64.zip*: Este é o nome do pacote de ferramentas cliente do **RedHat x64**. Substitua-o pelo nome real.

Passo 3 Configure o cliente de GaussDB(DWS).

```
source gsql_env.sh
```

Se as seguintes informações forem exibidas, o cliente gsql será configurado com êxito:

```
All things done.
```

Passo 4 Use o cliente gsql para conectar-se a um banco de dados do GaussDB(DWS) (usando a senha você definiu ao criar o cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

Se as informações a seguir forem exibidas, a conexão foi bem-sucedida.

```
gaussdb=>
```

----Fim

Criação de uma tabela de séries temporais

1. A seguir, descreve-se como criar uma tabela de séries temporais **GENERATOR** para armazenar os dados de amostra de conjunto de gerador.

```
CREATE TABLE IF NOT EXISTS GENERATOR(
genset text TSTag,
manufacturer text TSTag,
model text TSTag,
location text TSTag,
ID bigint TSTag,
voltage numeric TSField,
power bigint TSField,
frequency numeric TSField,
angle numeric TSField,
time timestamptz TSTime) with (orientation=TIMESERIES, period='1 hour',
ttl='1 month') distribute by hash(model);
```

2. Consultr a hora atual.

```
select now();
              now
```



```
------------------------------
2022-05-25 15:28:38.520757+08
(1 row)
```

3. Consulte a partição padrão e o limite de partição.

```
SELECT relname, boundaries FROM pg_partition where parentid=(SELECT oid FROM
pg_class where relname='generator') order by boundaries ;
    relname     |         boundaries
 ---------------+----------------------------
default_part_1 | {"2022-05-25 16:00:00+08"}
default_part_2 | {"2022-05-25 17:00:00+08"}
p1653505200    | {"2022-05-25 18:00:00+08"}
p1653541200    | {"2022-05-25 19:00:00+08"}
p1653577200    | {"2022-05-25 20:00:00+08"}
......
```

As colunas **TSTAG** suportam os tipos text, char, bool, int e big int.

A coluna **TSTime** suporta o carimbo de data/hora com fuso horário e carimbo de data/hora sem tipos de fuso horário. Também suporta o tipo de data em bancos de dados compatíveis com a sintaxe Oracle. Se operações relacionadas ao fuso horário estiverem envolvidas, selecione um tipo de horário com fuso horário.

Os tipos de dados suportados pelas colunas de **TSField** são os mesmos suportados pelas tabelas de armazenamento de colunas.

NOTA

- Ao escrever instruções de criação de tabela, você pode otimizar a sequência de colunas de tags. Colunas mais exclusivas (valores mais distintos) são escritas na frente para melhorar o desempenho em cenários de sequência de tempo.
- Ao criar uma tabela de séries temporais, defina o parâmetro em nível de tabela **orientation** para **timeseries**.
- Não é necessário especificar manualmente **DISTRIBUTE BY** e **PARTITION BY** para uma tabela de séries temporais. Por padrão, os dados são distribuídos com base em todas as colunas de tags e a chave de partição é a coluna TStime.
- Na sintaxe **create table like**, os nomes das colunas e os tipos **kv_type** são herdados automaticamente da tabela de origem. Se a tabela de origem for uma tabela de série não temporal e a nova tabela for uma tabela de série temporal, o tipo **kv_type** da coluna correspondente não poderá ser determinado. Como resultado, a criação falha.
- Um e somente um atributo **TSTIME** deve ser especificado. Colunas do tipo TSTIME não podem ser excluídas. Deve haver pelo menos uma coluna **TSTag** e **TSField**. Caso contrário, um erro será reportado durante a criação da tabela.

As tabelas de séries temporais usam a coluna TSTIME como chave de partição e têm a função de gerenciamento automático de partição. Tabelas de partição com a função de gerenciamento automático de partição ajudam os usuários a reduzir significativamente o tempo de O&M. Na instrução de criação de tabela anterior, você pode ver nos parâmetros de nível de tabela que dois parâmetros **period** e **ttl** são especificados para a tabela de séries temporais.

– **period**: intervalo para criar partições automaticamente. O valor padrão é 1 dia. A faixa de valor é de 1 hora a 100 anos. Por padrão, uma tarefa de partição de incremento automático é criada para a tabela de séries temporais. A tarefa de partição de incremento automático cria partições dinamicamente para garantir que partições suficientes estejam disponíveis para importar dados.
– **ttl**: tempo para eliminar automaticamente as partições. A faixa de valor é de 1 hora a 100 anos. Por padrão, nenhuma tarefa de eliminação de partição é criada. Você precisa especificar manualmente a tarefa de eliminação de partição ao criar uma tabela ou usar a sintaxe ALTER TABLE para definir a tarefa de eliminação de partição após criar uma tabela. A política de eliminação de partição baseia-se na condição de nowtime - partition boundary > ttl. As partições que atendem a essa

condição serão eliminadas. Esse recurso ajuda os usuários a excluir dados obsoletos periodicamente.

NOTA

Para limites de partição

- Se a unidade **period** for hora, o valor do limite inicial será a hora seguinte e o intervalo de partição será o valor do **period**.
- Se a unidade **period** for dia, o valor do limite inicial será 00:00 do próximo dia e o intervalo de partição será o valor do **period**.
- Se a unidade **period** for mês, o valor do limite inicial será 00:00 do próximo mês e o intervalo de partição será o valor do **period**.
- Se a unidade **period** for ano, o valor do limite inicial será 00:00 do ano seguinte e o intervalo de partição será o valor do **period**.

Criar uma tabela de séries temporais (definindo limites de partição manualmente)

1. Especifique manualmente o valor do limite inicial. Por exemplo, crie a tabela de séries temporais **GENERATOR1** com o limite inicial padrão da partição **P1** como **2022-05-30 16:32:45** e a partição **P2** como **2022-05-31 16:56:12**.

```
CREATE TABLE IF NOT EXISTS GENERATOR1(
genset text TSTag,
manufacturer text TSTag,
model text TSTag,
location text TSTag,
ID bigint TSTag,
voltage numeric TSField,
power bigint TSField,
frequency numeric TSField,
angle numeric TSField,
time timestamptz TSTime) with (orientation=TIMESERIES, period='1 day')
distribute by hash(model)
partition by range(time)
(
PARTITION P1 VALUES LESS THAN('2022-05-30 16:32:45'),
PARTITION P2 VALUES LESS THAN('2022-05-31 16:56:12')
);
```

2. Consulte a hora atual:

```
select now();
              now
-------------------------------
2022-05-31 20:36:09.700096+08(1 row)
```

3. Execute o seguinte comando para consultar partições e limites de partição:

```
SELECT relname, boundaries FROM pg_partition where parentid=(SELECT oid FROM
pg_class where relname='generator1') order by boundaries ;
   relname   |         boundaries
-------------+----------------------------
p1           | {"2022-05-30 16:32:45+08"}
p2           | {"2022-05-31 16:56:12+08"}
p1654073772 | {"2022-06-01 16:56:12+08"}
p1654160172 | {"2022-06-02 16:56:12+08"}
......
```


4.2 Melhores práticas de gerenciamento de dados quentes e frios

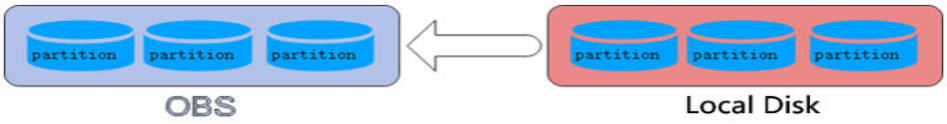
Cenários

Em cenários massivos de Big Data, com o crescimento dos dados, o armazenamento e o consumo de dados aumentam rapidamente. A necessidade de dados pode variar em diferentes períodos de tempo, portanto, os dados são gerenciados de maneira hierárquica, melhorando o desempenho da análise de dados e reduzindo os custos do serviço. Em alguns cenários de uso de dados, os dados podem ser classificados em dados quentes e dados frios acessando a frequência.

Os dados quentes e frios são classificados com base na frequência de acesso aos dados e na frequência de atualização.

- Dados quentes: dados que são acessados e atualizados com frequência e exigem resposta rápida.
- Dados frios: dados que não podem ser atualizados ou raramente são acessados e não exigem resposta rápida

Você pode definir tabelas de gerenciamento frias e quentes para alternar dados frios que atendam às regras especificadas para o OBS para armazenamento. Dados frios e quentes podem ser automaticamente determinados e migrados por partição.



As partições quentes e frias podem ser comutadas com base nas políticas LMT (Tempo de última modificação) e HPN (Número de partição quente). LMT indica que a alternância está executada baseado no tempo da última atualização da divisória, e HPN indica que a alternância está executada baseado no número de partições quentes reservadas.

- **LMT**: alterne os dados da partição quente que não foram atualizados nos últimos *[day]* dias para o espaço de tabela do OBS como dados da partição fria. *[day]* é um número inteiro que varia de 0 a 36500, em dias.
- **HPN**: indica o número de partições quentes a serem reservadas. Durante a alternância de frio e quente, os dados precisam ser migrados para o OBS. HPN é um número inteiro que varia de 0 a 1600.

Restrições

- Se uma tabela tiver partições frias e quentes, a consulta torna-se lenta porque os dados frios são armazenados no OBS e a velocidade de leitura/gravação é menor do que a das consultas locais.
- Atualmente, as tabelas frias e quentes suportam apenas tabelas particionadas de armazenamento de colunas da versão 2.0. Tabelas estrangeiras não suportam partições frias e quentes.
- Somente dados quentes podem ser trocados por dados frios. Dados frios não podem ser alternados para dados quentes.

Procedimento

Essa prática leva cerca de 30 minutos. O processo básico é o seguinte:

1. **Criação de um cluster**.
2. **Usar o cliente de CLI gsql para conectar-se a um cluster**.
3. **Criar tabelas quentes e frias**.
4. **Alternância de dados quentes e frios**.
5. **Visualizar a distribuição de dados em tabelas quentes e frias**.

Criação de um cluster

Passo 1 Faça logon no console de gerenciamento da Huawei Cloud.

Passo 2 Escolha **Service List** > **Analytics** > **Data Warehouse Service**. Na página exibida, clique em **Create Cluster** no canto superior direito.

Passo 3 Configure parâmetros de acordo com **Tabela 4-2**.

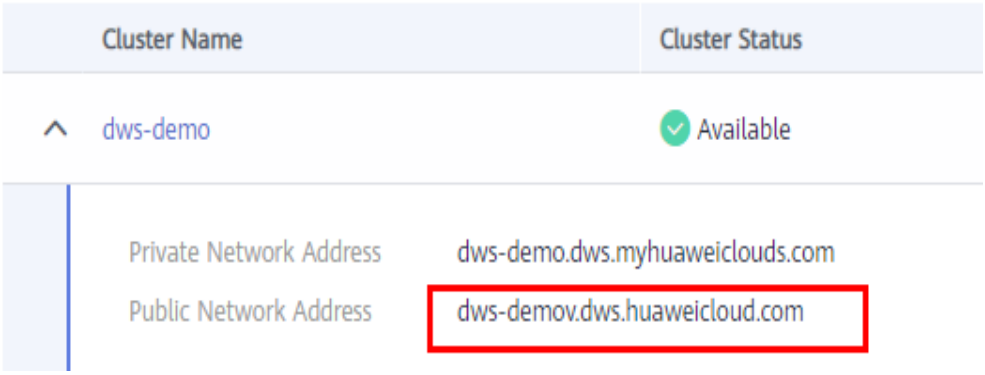
Tabela 4-2 Configuração de software

Parâmetro	Configuração
Region	Selecione CN-Hong Kong. **NOTA** **CN-Hong Kong** é usada como exemplo. Você pode selecionar outras regiões, conforme necessário. Certifique-se de que todas as operações sejam realizadas na mesma região.
AZ	AZ2
Product	Standard data warehouse
CPU Architecture	X86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD) **NOTA** Se esse flavor estiver esgotado, selecione outras AZs ou flavors.
Nodes	3
Cluster Name	dws-demo
Administrator Account	dbadmin
Administrator Password	-
Confirm Password	-
Database Port	8000

Parâmetro	Configuração
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24)
Security Group	Automatic creation
EIP	Buy now
Bandwidth	1Mbit/s
Advanced Settings	Default

Passo 4 Confirme as informações, clique em **Next** e, em seguida, clique em **Submit**.

Passo 5 Espere cerca de 6 minutos. Depois que o cluster for criado, clique em ao lado do nome do cluster. Na página de informações do cluster exibida, registre o valor de **Public Network Address**, por exemplo, **dws-demov.dws.huaweicloud.com**.



----Fim

Usar o cliente de CLI gsql para conectar-se a um cluster

Passo 1 Faça logon remotamente no servidor Linux onde o gsql deve ser instalado como usuário **root** e execute o seguinte comando na janela de comando do Linux para fazer o download do cliente gsql:

```
wget https://obs.ap-southeast-1.myhuaweicloud.com/dws/download/
dws_client_8.1.x_redhat_x64.zip --no-check-certificate
```

Passo 2 Descompacte o cliente.

```
cd <Path_for_storing_the_client> unzip dws_client_8.1.x_redhat_x64.zip
```

Onde,

- *<Path_for_storing_the_client>*: Substitua-o pelo caminho real.
- *dws_client_8.1.x_redhat_x64.zip*: Este é o nome do pacote de ferramentas cliente do **RedHat x64**. Substitua-o pelo nome real.

Passo 3 Configure o cliente de GaussDB(DWS).

```
source gsql_env.sh
```

Se as seguintes informações forem exibidas, o cliente gsql será configurado com êxito:

```
All things done.
```

Passo 4 Use o cliente gsql para conectar-se a um banco de dados do GaussDB(DWS) (usando a senha você definiu ao criar o cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

Se as informações a seguir forem exibidas, a conexão foi bem-sucedida.

```
gaussdb=>
```

----Fim

Criar tabelas quentes e frias

Crie uma tabela de gerenciamento de dados frios e quentes **lifecycle_table** e defina o período de validade de dados quentes de LMT como 100 dias.

```
CREATE TABLE lifecycle_table(i int, val text) WITH (ORIENTATION = COLUMN,
storage_policy = 'LMT:100')
PARTITION BY RANGE (i)
(
PARTITION P1 VALUES LESS THAN(5),
PARTITION P2 VALUES LESS THAN(10),
PARTITION P3 VALUES LESS THAN(15),
PARTITION P8 VALUES LESS THAN(MAXVALUE)
)
ENABLE ROW MOVEMENT;
```

Alternância de dados quentes e frios

Alterne dados frios para o espaço de tabela do OBS.

- Alternância automática: o agendador aciona automaticamente a alternância às 00:00 todos os dias.

  Você pode usar a função pg_obs_cold_refresh_time(table_name, time) para personalizar o tempo de alternância automática. Por exemplo, defina o horário de disparo automático para 06:30 todas as manhãs com base nos requisitos de serviço.

  ```
  SELECT * FROM pg_obs_cold_refresh_time('lifecycle_table', '06:30:00');
  pg_obs_cold_refresh_time
  --------------------------
   SUCCESS
  (1 row)
  ```

- Manual

  Execute a instrução ALTER TABLE para alternar manualmente uma única tabela.

  ```
  ALTER TABLE lifecycle_table refresh storage;
  ALTER TABLE
  ```

  Use a função pg_refresh_storage() para alternar todas as tabelas quentes e frias em lotes.

  ```
  SELECT pg_catalog.pg_refresh_storage();
   pg_refresh_storage
  --------------------
   (1,0)
  (1 row)
  ```


Visualizar a distribuição de dados em tabelas quentes e frias

- Veja a distribuição de dados em uma única tabela:

```
SELECT * FROM
pg_catalog.pg_lifecycle_table_data_distribute('lifecycle_table');
schemaname |    tablename    |   nodename   | hotpartition | coldpartition |
switchablepartition | hotdatasize | colddatasize | switchabledatasize
------------+-----------------+--------------+--------------+---------------
+---------------------+-------------+--------------+--------------------
 public     | lifecycle_table | dn_6001_6002 | p1,p2,p3,p8  |
|                     | 96 KB       | 0 bytes      | 0 bytes
 public     | lifecycle_table | dn_6003_6004 | p1,p2,p3,p8  |
|                     | 96 KB       | 0 bytes      | 0 bytes
 public     | lifecycle_table | dn_6005_6006 | p1,p2,p3,p8  |
|                     | 96 KB       | 0 bytes      | 0 bytes
(3 rows)
```

- Veja a distribuição de dados em todas as tabelas quentes e frias:

```
SELECT * FROM pg_catalog.pg_lifecycle_node_data_distribute();
schemaname |    tablename    |   nodename   | hotpartition | coldpartition |
switchablepartition | hotdatasize | colddatasize | switchabledatasize
------------+-----------------+--------------+--------------+---------------
+---------------------+-------------+--------------+--------------------
 public     | lifecycle_table | dn_6001_6002 | p1,p2,p3,p8  |
|                     |       98304 |            0 |                    0
 public     | lifecycle_table | dn_6003_6004 | p1,p2,p3,p8  |
|                     |       98304 |            0 |                    0
 public     | lifecycle_table | dn_6005_6006 | p1,p2,p3,p8  |
|                     |       98304 |            0 |                    0
(3 rows)
```

4.3 Melhores práticas para gerenciamento automático de partições

Cenários

Para tabelas de partição cujas colunas de partição são tempo, a função de gerenciamento automático de partição pode ser adicionada para criar automaticamente partições e excluir partições expiradas, reduzindo os custos de manutenção da tabela de partição e melhorando o desempenho da consulta. Para facilitar a consulta e a manutenção de dados, a coluna de tempo é frequentemente usada como a coluna de partição de uma tabela particionada que armazena dados relacionados ao tempo, como informações de pedidos de comércio eletrônico e dados de IoT em tempo real. Quando os dados relacionados ao tempo são importados para uma tabela particionada, a tabela deve ter partições dos intervalos de tempo correspondentes. Tabelas de partições comuns não criam automaticamente novas partições nem excluem partições expiradas. Portanto, o pessoal de manutenção precisa criar periodicamente novas partições e excluir partições expiradas, levando ao aumento dos custos de O&M.

Para resolver isso, o GaussDB(DWS) introduz o recurso de gerenciamento automático de partições. Você pode definir os parâmetros de nível de tabela **period** e **ttl** para habilitar a função de gerenciamento automático de partição, que cria automaticamente partições e exclui partições expiradas, reduzindo os custos de manutenção da tabela de partição e melhorando o desempenho da consulta.

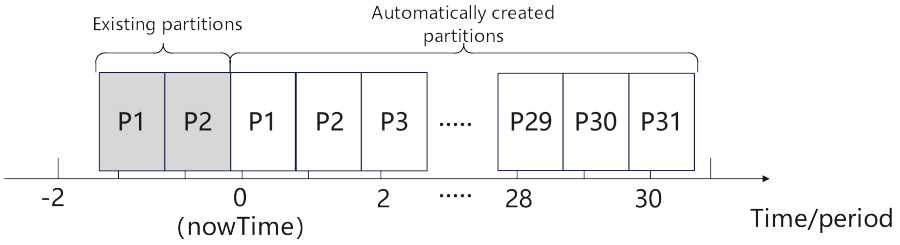
period: intervalo para criar partições automaticamente. O valor padrão é 1 dia. A faixa de valor é de 1 hora a 100 anos.

ttl: tempo para eliminar automaticamente as partições. A faixa de valor é de 1 hora a 100 anos. A política de eliminação de partição baseia-se na condição de nowtime - partition boundary > ttl. As partições que atendem a essa condição serão eliminadas.

- Criação automática de partições

  Uma ou mais partições são criadas automaticamente no intervalo especificado por **period** para tornar o tempo máximo de limite de partição maior que nowTime + 30 x período. Enquanto houver uma partição criada automaticamente, os dados em tempo real não deixarão de ser importados nos próximos 30 períodos.

  Figura 4-3 Criação automática de partições

- Exclusão automática de partições expiradas

  As partições cujo tempo limite é anterior a **nowTime-ttl** são consideradas partições expiradas. A função automática da gestão da divisória atravessa todas as divisórias e suprime divisórias expiradas após cada **period**. Se todas as partições forem partições expiradas, o sistema retém uma partição e trunca a tabela.

Restrições

Ao usar a função de gerenciamento de partição, certifique-se de que os seguintes requisitos sejam atendidos:

- Ela não pode ser usada em servidores de médio porte, clusters de aceleração ou clusters autônomos.
- Ela pode ser usada em clusters da versão 8.1.3 ou posterior.
- Ela só pode ser usada para tabelas particionadas de intervalo de armazenamento de linha, tabelas particionadas de intervalo de armazenamento de coluna, tabelas de séries temporais e tabelas frias e quentes.
- A chave de partição deve ser exclusiva e seu tipo deve ser timestamp, timestamptz ou date.
- A partição maxvalue não é suportada.
- O valor de (nowTime - boundaryTime)/período deve ser menor que o número máximo de partições. **nowTime** indica o horário atual e **boundaryTime** indica o horário de limite de partição mais anterior.
- Os valores de **period** e **ttl** variam de 1 hora a 100 anos. Além disso, em um banco de dados compatível com Teradata ou MySQL, se o tipo de chave de partição é data, o valor do período não pode ser inferior a 1 dia.
- O parâmetro de nível de tabela **ttl** não pode existir de forma independente. Você deve definir **period** com antecedência ou ao mesmo tempo, e o valor de **ttl** deve ser maior ou igual ao de **period**.
- Durante a expansão de cluster online, as partições não podem ser adicionadas automaticamente. As partições reservadas cada vez que as partições são adicionadas garantirão que os serviços não sejam afetados.

Criar um ECS

Para obter detalhes, consulte **Compra de um ECS**. Após a compra de um ECS, faça logon no ECS consultando **Efetuar logon em um ECS de Linux**.

AVISO

Ao criar um ECS, certifique-se de que o ECS esteja na mesma região, AZ e sub-rede da VPC que o armazém de dados de fluxo. Selecione o SO usado pelo cliente gsql (o CentOS 7.6 é usado como um exemplo) como o SO do ECS e selecione usar senhas para fazer logon.

Criação de um cluster

Passo 1 Faça logon no console de gerenciamento da Huawei Cloud.

Passo 2 Escolha **Service List** > **Analytics** > **Data Warehouse Service**. Na página exibida, clique em **Create Cluster** no canto superior direito.

Passo 3 Configure parâmetros de acordo com **Tabela 4-3**.

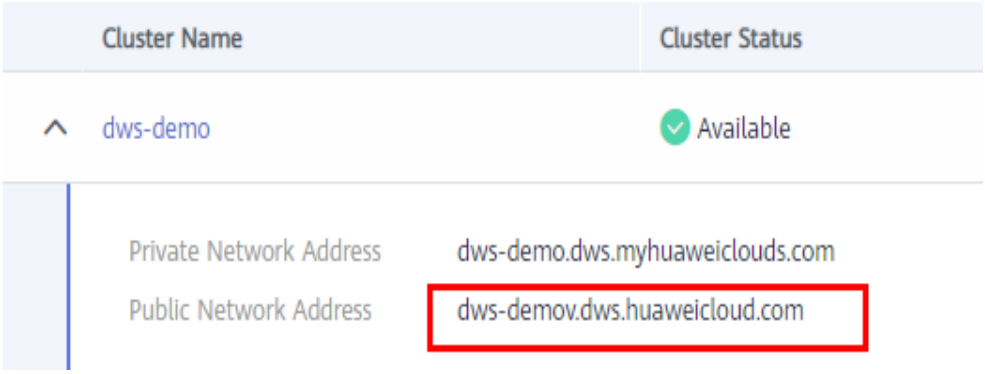
Tabela 4-3 Configuração de software

Parâmetro	Configuração
Region	Selecione CN-Hong Kong. **NOTA** **CN-Hong Kong** é usada como exemplo. Você pode selecionar outras regiões, conforme necessário. Certifique-se de que todas as operações sejam realizadas na mesma região.
AZ	AZ2
Product	Standard data warehouse
CPU Architecture	X86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD) **NOTA** Se esse flavor estiver esgotado, selecione outras AZs ou flavors.
Nodes	3
Cluster Name	dws-demo
Administrator Account	dbadmin
Administrator Password	-
Confirm Password	-

Parâmetro	Configuração
Database Port	8000
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24)
Security Group	Automatic creation
EIP	Buy now
Bandwidth	1Mbit/s
Advanced Settings	Default

Passo 4 Confirme as informações, clique em **Next** e, em seguida, clique em **Submit**.

Passo 5 Espere cerca de 6 minutos. Depois que o cluster for criado, clique em ao lado do nome do cluster. Na página de informações do cluster exibida, registre o valor de **Public Network Address**, por exemplo, **dws-demov.dws.huaweicloud.com**.



----Fim

Usar o cliente de CLI gsql para conectar-se a um cluster

Passo 1 Faça logon remotamente no servidor Linux onde o gsql deve ser instalado como usuário **root** e execute o seguinte comando na janela de comando do Linux para fazer o download do cliente gsql:

```
wget https://obs.ap-southeast-1.myhuaweicloud.com/dws/download/
dws_client_8.1.x_redhat_x64.zip --no-check-certificate
```

Passo 2 Descompacte o cliente.

```
cd <Path_for_storing_the_client> unzip dws_client_8.1.x_redhat_x64.zip
```

Onde,

- *<Path_for_storing_the_client>*: Substitua-o pelo caminho real.
- *dws_client_8.1.x_redhat_x64.zip*: Este é o nome do pacote de ferramentas cliente do **RedHat x64**. Substitua-o pelo nome real.

Passo 3 Configure o cliente de GaussDB(DWS).

```
source gsql_env.sh
```

Se as seguintes informações forem exibidas, o cliente gsql será configurado com êxito:

```
All things done.
```

Passo 4 Use o cliente gsql para conectar-se a um banco de dados do GaussDB(DWS) (usando a senha você definiu ao criar o cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

Se as informações a seguir forem exibidas, a conexão foi bem-sucedida.

```
gaussdb=>
```

----Fim

Gerenciamento de partições automático

A função de gerenciamento de partição está vinculada aos parâmetros de nível de tabela **period** e **ttl**. A criação automática de partições é permitida com a habilitação de **period**, e a exclusão automática de partições é permitida com a habilitação de **ttl**. 30 segundos após **period** ou **ttl** é ajustado, a criação ou a exclusão automática de partições funciona pela primeira vez.

Você pode habilitar a função de gerenciamento de partições de uma das seguintes maneiras:

- Especifique **period** e **ttl** ao criar uma tabela.

  Essa maneira é aplicável quando você cria uma tabela de gerenciamento de partição. Há duas sintaxes para criar uma tabela de gerenciamento de partição. Uma especifica partições e a outra não.

  Se as partições forem especificadas quando uma tabela de gerenciamento de partição for criada, as regras de sintaxe serão as mesmas para a criação de uma tabela de partição comum. A única diferença é que a sintaxe especifica os parâmetros de nível de tabela **period** e **ttl**.

  O exemplo a seguir mostra como criar uma tabela de gerenciamento de partições **CPU1** e especificar partições.

  ```
  CREATE TABLE CPU1(
      id integer,
      IP text,
      time timestamp
  ) with (TTL='7 days',PERIOD='1 day')
  partition by range(time)
  (
      PARTITION P1 VALUES LESS THAN('2023-02-13 16:32:45'),
      PARTITION P2 VALUES LESS THAN('2023-02-15 16:48:12')
  );
  ```

  Ao criar uma tabela de gerenciamento de partição, você pode especificar apenas a chave de partição, mas não partições. Nesse caso, duas partições padrão serão criadas com **period** como o intervalo de tempo da partição. A hora limite da primeira partição padrão é a primeira hora, dia, semana, mês ou ano após a hora atual. A unidade de tempo é selecionada com base na unidade máxima de PERIOD. O tempo limite da segunda partição padrão é o tempo limite da primeira partição mais o PERIOD. Suponha que a

hora atual é 2023-02-17 16:32:45 e o limite da primeira partição padrão é descrito na tabela a seguir.

Tabela 4-4 Descrição do parâmetro período

Período	Unidade do PERIOD máximo	Limite da primeira partição padrão
1 hora	hora	2023-02-17 17:00:00
1 dia	Dia	18/02/2023 00:00:00
1 mês	Mês	01/03/2023 00:00:00
13 meses	Ano	01/01/2024 00:00:00

Execute o seguinte comando para criar a tabela de gerenciamento de partições **CPU2** sem partições especificadas:

```
CREATE TABLE CPU2(
    id integer,
    IP text,
    time timestamp
) with (TTL='7 days',PERIOD='1 day')
partition by range(time);
```

- Execute o comando **ALTER TABLE RESET** para definir **period** e **ttl**.

  Esse método é usado para adicionar a função de gerenciamento de partição a uma tabela particionada comum que atenda às restrições de gerenciamento de partição.

  – Execute o seguinte comando para criar uma tabela de partição comum **CPU3**:

  ```
  CREATE TABLE CPU3(
      id integer,
      IP text,
      time timestamp
  )
  partition by range(time)
  (
      PARTITION P1 VALUES LESS THAN('2023-02-14 16:32:45'),
      PARTITION P2 VALUES LESS THAN('2023-02-15 16:56:12')
  );
  ```

  – Para ativar as funções automáticas de criação e exclusão de partições, execute o seguinte comando:

  ```
  ALTER TABLE CPU3 SET (PERIOD='1 day',TTL='7 days');
  ```

  – Para ativar apenas a função de criação automática de partição, execute o seguinte comando:

  ```
  ALTER TABLE CPU3 SET (PERIOD='1 day');
  ```

  – Para ativar apenas a função de exclusão automática de partição, execute o seguinte comando (Se a criação automática de partição não estiver ativada antecipadamente, a operação falhará):

  ```
  ALTER TABLE CPU3 SET (TTL='7 days');
  ```

  – Modifique os parâmetros **period** e **ttl** para alterar a função de gerenciamento de partição.

  ```
  ALTER TABLE CPU3 SET (TTL='10 days',PERIOD='2 days');
  ```

- Desativar a função de gerenciamento de partição

  Você pode executar o comando **ALTER TABLE RESET** para excluir os parâmetros de nível de tabela **period** e **ttl** para desativar a função de gerenciamento de partição.

NOTA

- O **period** não pode ser excluído separadamente com **TTL**.
- A tabela de séries temporais não suporta **ALTER TABLE RESET**.

– Execute o seguinte comando para desabilitar as funções automáticas de criação e exclusão de partições:

```
ALTER TABLE CPU1 RESET (PERIOD,TTL);
```

– Para desabilitar apenas a exclusão automática de partição, execute o seguinte comando:

```
ALTER TABLE CPU3 RESET (TTL);
```

– Para desabilitar apenas a função de criação automática de partições, execute o seguinte comando (Se a tabela contiver o parâmetro **ttl**, a operação falhará):

```
ALTER TABLE CPU3 RESET (PERIOD);
```

4.4 Desacoplamento e reconstrução automática de exibição do GaussDB(DWS)

Para resolver o problema de que os objetos da tabela base não podem ser modificados independentemente devido à dependência de exibição e tabela, o GaussDB(DWS) implementa desacoplamento e reconstrução de exibição. Este documento descreve os cenários de aplicação e os métodos de utilização da função de reconstrução automática da exibição.

Cenário

GaussDB(DWS) usa identificadores de objeto (OIDs) para armazenar relações de referência entre objetos. Quando uma exibição é definida, o OID do objeto de banco de dados do qual a exibição depende é vinculado a ela. Não importa como o nome da exibição muda, a dependência não muda. Se você modificar algumas colunas na tabela base, um erro será relatado porque as colunas estão fortemente vinculadas a alguns objetos. Se você quiser excluir uma coluna de tabela ou toda a tabela, precisará usar a palavra-chave **cascade** para excluir as exibições associadas. Depois que a coluna da tabela for excluída ou a tabela for recriada, você precisará recriar as exibições de diferentes níveis, uma por uma. Isso aumenta a carga de trabalho e deteriora a usabilidade.

Para resolver esse problema, o GaussDB(DWS) 8.1.0 desacopla as exibições de suas tabelas base dependentes ou outros objetos de banco de dados exibições, sinônimos, funções e colunas de tabela), para que esses objetos possam ser excluídos independentemente. Depois que a tabela base for reconstruída, você poderá executar o comando **ALTER VIEW view_name REBUILD** para reconstruir a dependência. Em 8.1.1, a reconstrução automática é implementada. As relações de dependência podem ser reconstruídas automaticamente sem serem percebidas. Depois que a reconstrução automática for ativada, podem ocorrer conflitos de bloqueio. Portanto, não é aconselhável ativar a reconstrução automática.

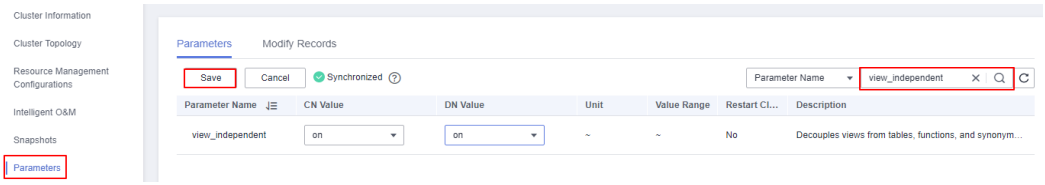
Uso

Passo 1 Crie um cluster no console de gerenciamento. Para obter detalhes, consulte a seção **Criação de um cluster**.

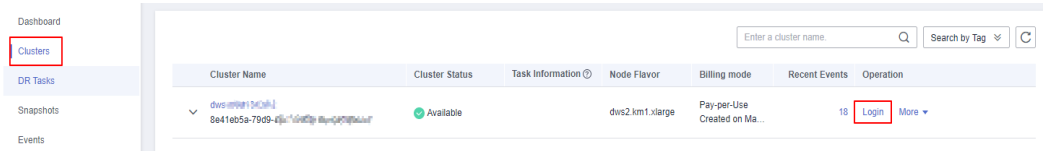
Passo 2 Ative o parâmetro de GUC **view_independent**.

O parâmetro de GUC **view_independent** controla se uma exibição deve ser desacoplada de seus objetos. Este parâmetro está desativado por padrão. Você precisa ativar manualmente o

parâmetro. Para ativar o parâmetro **view_independent**, efetue logon no console de gerenciamento e clique no nome do cluster. Na página **Cluster Details** exibida, clique na guia **Parameters**, procure por **view_independent**, modifique o parâmetro e salve a modificação.



Passo 3 Use o DAS para se conectar a um cluster. Localize o cluster necessário na lista de clusters e clique em **Log In** na coluna **Operation**. Na página do DAS exibida, digite o nome do usuário, o nome do banco de dados e a senha e teste a conexão. Se a conexão for bem-sucedida, faça logon no cluster. Para obter detalhes, consulte **Uso do DAS para conectar-se a um cluster**.



Passo 4 Crie uma tabela de exemplo **t1** e insira dados na tabela.

```
SET current_schema='public';
CREATE TABLE t1 (a int, b int, c char(10)) DISTRIBUTE BY HASH (a);
INSERT INTO t1 VALUES(1,1,'a'),(2,2,'b');
```

Passo 5 Crie a exibição **v1** que depende da tabela **t1** e crie a exibição **v11** que depende da exibição **v1**. Consulte exibição **v11**.

```
CREATE VIEW v1 AS SELECT a, b FROM t1;
CREATE VIEW v11 AS SELECT a FROM v1;

SELECT * FROM v11;
 a
---
 1
 2
(2 rows)
```

Passo 6 Depois que a tabela **t1** é excluída, um erro é relatado quando você consulta a exibição **v11**. No entanto, as exibições ainda existem.

GaussDB(DWS) fornece a exibição **GS_VIEW_INVALID** para consultar todas as exibições inválidas visíveis para o usuário. Se a tabela base, função ou sinônimo de que a exibição depende for anormal, a coluna **validtype** da exibição será exibida como "invalid".

```
DROP TABLE t1;

SELECT * FROM v11;
ERROR:  relation "public.t1" does not exist

SELECT * FROM gs_view_invalid;
  oid   | schemaname | viewname | viewowner |         definition          |
validtype
--------+------------+----------+-----------+-----------------------------
+-----------
 213563 | public     | v1       | dbadmin   | SELECT a, b FROM public.t1; |
invalid
 213567 | public     | v11      | dbadmin   | SELECT a FROM public.v1;    |
invalid
(2 rows)
```


Passo 7 Em um cluster de uma versão anterior após a recriação da tabela t1, a exibição é automaticamente recriada. As exibições são automaticamente atualizadas somente quando são usadas.

```
CREATE TABLE t1 (a int, b int, c char(10)) DISTRIBUTE BY HASH (a);
INSERT INTO t1 VALUES(1,1,'a'),(2,2,'b');

SELECT * from v1;
 a | b
---+---
 1 | 1
 2 | 2
(2 rows)

SELECT * FROM gs_view_invalid;
  oid   | schemaname | viewname | viewowner |        definition        | validtype
--------+------------+----------+-----------+--------------------------
+-----------
 213567 | public     | v11      | dbadmin   | SELECT a FROM public.v1; | invalid
(1 row)

SELECT * from v11;
 a
---
 1
 2
(2 rows)

SELECT * FROM gs_view_invalid;
 oid | schemaname | viewname | viewowner | definition | validtype
-----+------------+----------+-----------+------------+-----------
(0 rows)
```

----Fim

4.5 Melhores práticas de tabelas delta de armazenamento de coluna

Princípios de funcionamento

No GaussDB(DWS), os dados em uma tabela de armazenamento de coluna são armazenados por coluna. Por padrão, as 60.000 linhas em cada coluna são armazenadas em uma CU. Uma CU é a unidade mínima para armazenar dados em uma tabela de armazenamento de colunas. Depois que uma CU é gerada, os dados nela são fixos e não podem ser modificados. Não importa se um ou 60.000 registros de dados são inseridos em uma tabela de coluna de armazenamento, apenas uma CU é gerada. Quando uma pequena quantidade de dados é inserida em uma tabela de armazenamento de colunas várias vezes, ela não pode ser bem pressionada. Como resultado, ocorre o inchaço dos dados, o que afeta o desempenho da consulta e o uso do disco.

Os dados em um arquivo de CU não podem ser modificados e só podem ser anexados. Excluir os dados do arquivo de CU é marcar os dados anteriores como inválidos no dicionário. A atualização dos dados do arquivo de CU é marcar os dados antigos como inválidos e gravar um novo registro na nova CU. Se uma tabela de armazenamento de colunas for atualizada ou excluída várias vezes ou se apenas uma pequena quantidade de dados for inserida a cada vez, o espaço de tabela de armazenamento de colunas inchará e uma grande quantidade de espaço não poderá ser usada efetivamente.

As tabelas de armazenamento de colunas são projetadas para importar uma grande quantidade de dados e armazená-los por coluna para consulta. Para resolver os problemas anteriores, a

tabela delta é introduzida, que é uma tabela de armazenamento de linha anexada a uma tabela de armazenamento de coluna. Depois que a tabela delta é habilitada, quando um único dado ou um pequeno lote de dados é importado, os dados são armazenados na tabela delta para evitar pequenas CUs. A adição, exclusão, modificação e consulta da tabela delta são as mesmas das tabelas de armazenamento de linha. Depois que a tabela delta é ativada, o desempenho da importação de tabelas de armazenamento de colunas é bastante aprimorado.

Casos de uso

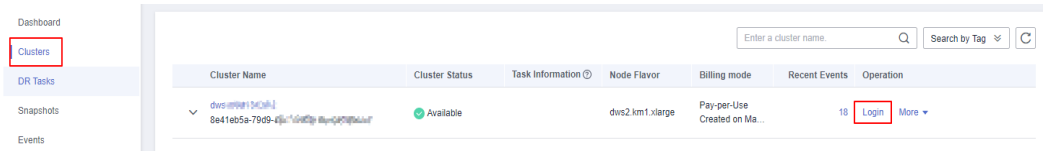
A tabela delta de armazenamento de colunas é usada para armazenamento híbrido de linhas e colunas e é adequada para análise e estatísticas em tempo real. Ela resolve o problema de desempenho causado pela importação de pequenos lotes de dados e mescla periodicamente os dados à tabela primária para garantir o desempenho da análise e da consulta. Você precisa determinar se deve habilitar tabelas delta com base na situação real. Caso contrário, as vantagens das tabelas de armazenamento de colunas de GaussDB(DWS) não podem ser totalmente utilizadas, desperdiçando espaço e tempo extra.

Preparativos

- Você registrou uma conta do GaussDB(DWS) e verificou o status da conta antes de usar GaussDB(DWS). A conta não pode estar em atraso ou congelada.
- Você obteve o AK e SK da conta.
- Os dados de amostra foram carregados na pasta **traffic-data** em um bucket do OBS, e todas as contas da Huawei Cloud receberam a permissão somente leitura para acessar o bucket do OBS. Para mais detalhes, consulte **Análise de veículos no ponto de verificação**.

Procedimento

Passo 1 Use o DAS para se conectar a um cluster. Localize o cluster necessário na lista de clusters e clique em **Log In** na coluna **Operation**. Na página de DAS exibida, digite o nome do usuário, o nome do banco de dados e a senha e teste a conexão. Se a conexão for bem-sucedida, faça logon no cluster. Para obter detalhes, consulte **Uso do DAS para se conectar a um cluster**.



Passo 2 Execute a instrução a seguir para criar o banco de dados **traffic**:

```
CREATE DATABASE traffic encoding 'utf8' template template0;
```

Passo 3 Execute as seguintes instruções para criar as tabelas de banco de dados **GCJL** e **GCJL2** para armazenar informações do veículo do ponto de verificação: Por padrão, a tabela delta não está ativada para **GCJL**, mas para **GCJL2**.

```
CREATE SCHEMA traffic_data;
SET current_schema= traffic_data;
DROP TABLE if exists GCJL;
CREATE TABLE GCJL
(
        kkbh    VARCHAR(20),
        hphm    VARCHAR(20),
        gcsj    DATE ,
        cplx    VARCHAR(8),
```



```
        cllx   VARCHAR(8),
        csys   VARCHAR(8)
)
with (orientation = column, COMPRESSION=MIDDLE)
distribute by hash(hphm);

DROP TABLE if exists GCJL2;
CREATE TABLE GCJL2
(
        kkbh   VARCHAR(20),
        hphm   VARCHAR(20),
        gcsj   DATE ,
        cplx   VARCHAR(8),
        cllx   VARCHAR(8),
        csys   VARCHAR(8)
)
with (orientation = column, COMPRESSION=MIDDLE, ENABLE_DELTA = TRUE)
distribute by hash(hphm);
```

NOTA

- As tabelas delta são desativadas por padrão. Para ativar tabelas delta, defina **enable_delta** como **true** ao criar tabelas de armazenamento de colunas.
- Você também pode executar o seguinte comando para ativar tabelas delta:
  ```
  ALTER TABLE table_name SET (enable_delta=TRUE);
  ```
- Se a tabela delta tiver sido ativada, você poderá executar o seguinte comando para desativá-la quando necessário:
  ```
  ALTER TABLE table_name SET (enable_delta=FALSE);
  ```

Passo 4 Crie uma tabela estrangeira, que é usada para identificar e associar os dados de origem no OBS.

AVISO

- *<obs_bucket_name>* indica o nome do bucket do OBS. Apenas algumas regiões são suportadas. Para obter detalhes sobre as regiões suportadas e os nomes dos bucket do OBS, consulte **Regiões suportadas**. Os clusters do GaussDB(DWS) não oferecem suporte ao acesso entre regiões aos dados do bucket do OBS.
- Nesta prática, a região **CN-Hong Kong** é usada como exemplo. Digite **dws-demo-ap-southeast-1** e substitua *<Access_Key_Id>* e *<Secret_Access_Key>* pelo valor atual.
- Se a mensagem"ERROR: schema "*xxx*" does not exist Position" for exibida quando você criar uma tabela estrangeira, o esquema não existe. Execute a etapa anterior para criar um esquema.

```
DROP FOREIGN table if exists GCJL_OBS;
CREATE FOREIGN TABLE GCJL_OBS
(
        like traffic_data.GCJL
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/traffic-data/gcxx',
        format 'text',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
```


Passo 5 Execute a instrução a seguir para importar dados da tabela estrangeira para a tabela do banco de dados:

```
INSERT INTO traffic_data.GCJL select * from GCJL_OBS;
INSERT INTO traffic_data.GCJL2 select * from GCJL_OBS;
```

Leva algum tempo para importar dados.

Passo 6 Execute a instrução a seguir para verificar o tamanho do espaço de armazenamento depois que a tabela do banco de dados é importada:

```
SELECT pg_size_pretty(pg_total_relation_size('traffic_data.GCJL'));
SELECT pg_size_pretty(pg_total_relation_size('traffic_data.GCJL2'));
```

Depois que a tabela delta é habilitada, o uso do espaço de armazenamento é reduzido de 8953 MB para 6053 MB, melhorando significativamente o desempenho da importação.

Passo 7 Execute a instrução a seguir para consultar dados na tabela. A velocidade da consulta é melhorada depois que a tabela delta é habilitada.

```
SELECT * FROM traffic_data.GCJL where hphm =  'YD38641';
SELECT * FROM traffic_data.GCJL2 where hphm =  'YD38641';
```

----Fim

Impacto da ativação da tabela delta

- A ativação da função de tabela delta de uma tabela de armazenamento de colunas pode impedir que pequenas CUs sejam geradas quando um único dado ou uma pequena quantidade de dados é importada para a tabela, melhorando assim o desempenho. Por exemplo, se 100 pedaços de dados são importados cada vez em um cluster com 3 CNs e 6 DNs, o tempo de importação pode ser reduzido em 25%, o uso do espaço de armazenamento pode ser reduzido em 97%. Portanto, você precisa habilitar a tabela delta antes de inserir um pequeno lote de dados por várias vezes e desabilitar a tabela delta depois de confirmar que nenhum pequeno lote de dados precisa ser importado.
- Uma tabela delta é uma tabela de armazenamento de linha anexada a uma tabela de armazenamento de coluna. Depois que os dados são inseridos em uma tabela delta, a alta taxa de compactação da tabela de armazenamento de colunas é perdida. Em casos normais, tabelas de armazenamento de colunas são usadas para importar uma grande quantidade de dados. Portanto, a tabela delta é desativada por padrão, se a tabela delta é ativada quando uma grande quantidade de dados é importada, mais tempo e espaço são consumidos. Se a tabela delta for ativada quando os 10.000 registros de dados forem importados em um cluster com 3 DNs e 6 DNs, a velocidade de importação será quatro vezes mais lenta e mais de 10 vezes o espaço será consumido do que quando a tabela delta estiver desabilitada. Portanto, tenha cuidado ao ativar a tabela delta.

5 Gerenciamento de banco de dados

5.1 Melhores práticas de gerenciamento de recursos

Essa prática demonstra como usar o GaussDB(DWS) para gerenciamento de recursos, ajudando as empresas a eliminar gargalos no desempenho de consultas simultâneas. Os trabalhos de SQL podem ser executados sem problemas sem afetar uns aos outros e consumir menos recursos do que antes.

Antes da preparação do experimento, se você não tiver conhecimento sobre gerenciamento de recursos, é aconselhável ler **Visão geral da página de gerenciamento de recursos**.

Essa prática leva cerca de 60 minutos. O procedimento é os seguintes:

1. **Passo 1: criar um cluster**
2. **Passo 2: conectar-se a um cluster e importar dados**
3. **Passo 3: criar um pool de recursos**
4. **Passo 4: verificar regras de exceção**

Cenários

Quando vários usuários de banco de dados executam trabalhos SQL no GaussDB(DWS) ao mesmo tempo, as seguintes situações podem ocorrer:

1. Algumas instruções SQL complexas ocupam recursos de cluster por um longo tempo, afetando o desempenho de outras consultas. Por exemplo, um grupo de usuários do banco de dados envia continuamente consultas complexas e demoradas, e outro grupo de usuários frequentemente envia consultas curtas. Nesse caso, as consultas curtas podem ter que esperar no pool de recursos para que as consultas demoradas sejam concluídas.
2. Algumas instruções SQL ocupam muita memória ou espaço em disco devido a distorção de dados ou planos de execução não otimizados. Como resultado, as instruções que não se aplicam a erros de relatório de memória ou o cluster muda para o modo somente leitura.

Para aumentar a taxa de transferência do sistema e melhorar o desempenho de SQL, você pode usar o gerenciamento de carga de trabalho do GaussDB(DWS). Por exemplo, crie um pool de recursos para usuários que enviam tarefas de consulta complexas com frequência e aloque mais recursos a esse pool de recursos. Os trabalhos complexos enviados por esses

usuários podem usar somente os recursos desse pool de recursos. Crie outro pool de recursos que ocupe menos recursos e adicione usuários que enviam consultas curtas a esse pool de recursos. Desta forma, os dois tipos de trabalhos podem ser executados sem problemas ao mesmo tempo.

Por exemplo, um banco processa serviços de processamento de transações on-line (OLTP) e processamento analítico on-line (OLAP). A prioridade do serviço de OLAP é menor que a do serviço de OLTP. Um grande número de consultas SQL complexas simultâneas pode causar contenção de recursos do servidor, enquanto um grande número de consultas SQL simples simultâneas podem ser rapidamente processadas sem serem colocadas em fila. Os recursos devem ser adequadamente alocados e gerenciados para garantir que os serviços de OLAP e OLTP possam funcionar sem problemas.

Os serviços OLAP são frequentemente complexos e não exigem alta prioridade ou resposta em tempo real. Os serviços de OLAP e OLTP são operados por usuários diferentes. Por exemplo, o usuário do banco de dados **budget_config_user** é usado para serviços de transação principal e o usuário **report_user** do banco de dados é usado para serviços de relatório. Os usuários estão sob gerenciamento independente de CPU e simultaneidade para melhorar a estabilidade do banco de dados.

Com base na pesquisa de carga de trabalho, monitoramento de rotina e teste e verificação de serviços de OLAP, descobriu-se que menos de 50 consultas SQL simultâneas não causam contenção de recursos do servidor ou resposta lenta do sistema de serviço. Os usuários de OLAP podem usar 20% dos recursos da CPU.

Com base na pesquisa de carga de trabalho, monitoramento de rotina e teste e verificação de serviços de OLTP, verifica-se que menos de 100 consultas SQL simultâneas não exercem pressão contínua sobre o sistema. Os usuários de OLTP podem usar 60% dos recursos da CPU.

- Configuração de recursos para usuários de OLAP (correspondente a **pool_1**): CPU = 20%, memória = 20%, armazenamento = 1.024.000 MB, simultaneidade = 20.
- Configuração de recursos para usuários de OLTP (correspondente a **pool_2**): CPU = 60%, memória = 60%, armazenamento = 1.024.000 MB, simultaneidade = 200.

Defina a memória máxima que pode ser usada por uma única instrução. Um erro será relatado se o uso de memória exceder o valor.

Em **Exception Rule**, defina **Blocking Time** como 1200s e **Execution Time** como 1800s. Um trabalho de consulta será encerrado após ser executado por mais de 1800 segundos.

Passo 1: criar um cluster

Crie um cluster referindo-se a **Criação de um cluster**.

Passo 2: conectar-se a um cluster e importar dados

Passo 1 Para obter detalhes, consulte **Usar o cliente de CLI gsql para conectar-se a um cluster**.

Passo 2 Importar dados de amostra. Para obter detalhes , consulte **Importação de dados do TPC-H**.

Passo 3 Execute as instruções a seguir para criar o usuário de OLTP **budget_config_user** e o usuário de OLAP **report_user**.

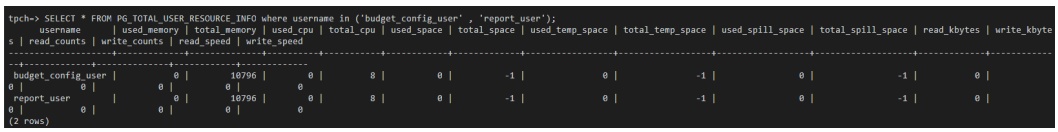
```
CREATE USER budget_config_user PASSWORD 'password';
CREATE USER report_user PASSWORD 'password';
```


Passo 4 Para fins de teste, conceda todas as permissões em todas as tabelas no esquema **tpch** para ambos os usuários.

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA tpch to
budget_config_user,report_user;
```

Passo 5 Verifique a alocação de recursos dos dois usuários.

```
SELECT * FROM PG_TOTAL_USER_RESOURCE_INFO where username in
('budget_config_user' , 'report_user');
```

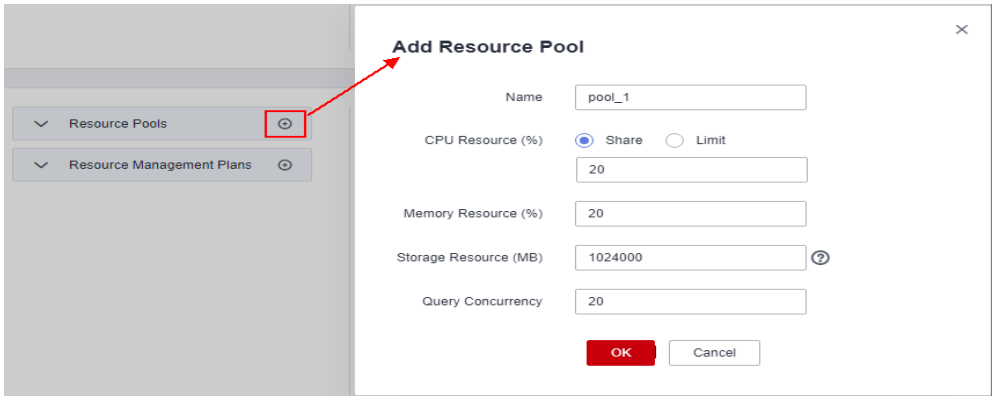


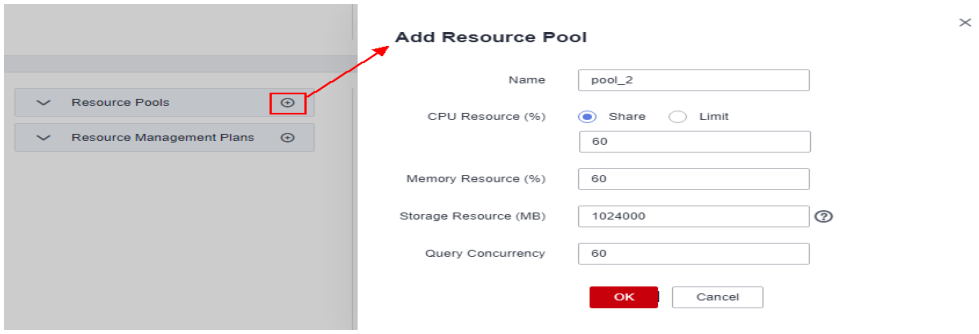
----Fim

Passo 3: criar um pool de recursos

Passo 1 Faça logon no console de gerenciamento GaussDB(DWS), clique em um nome de cluster na lista de clusters. A página **Resource Management Configurations** é exibida.

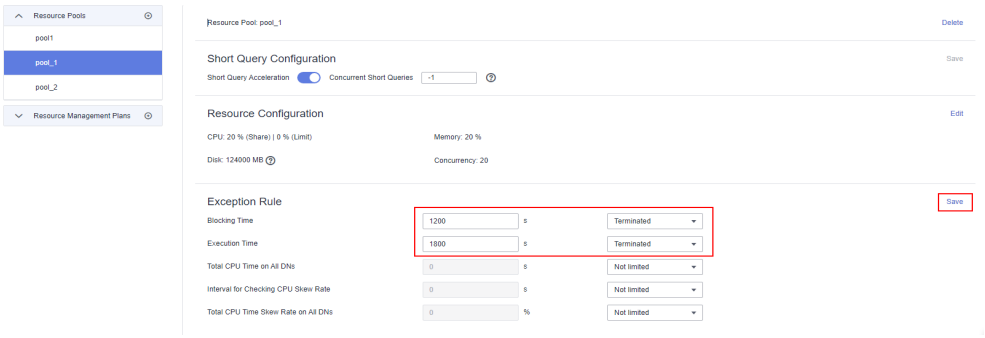
Passo 2 Clique em **Add Resource Pool** para criar um pool de recursos. Crie o pool de recursos de relatório **pool_1** e o pool de recursos de transação **pool_2** referindo-se a **Cenários**.





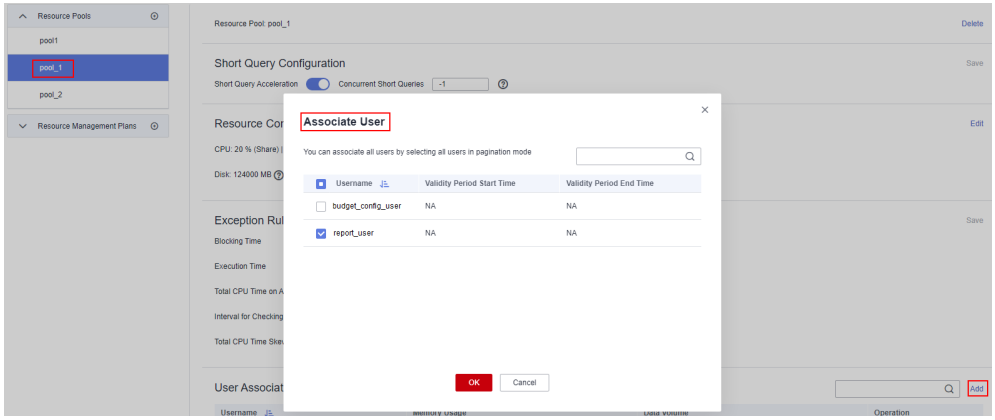
Passo 3 Modifique as regras de exceção.

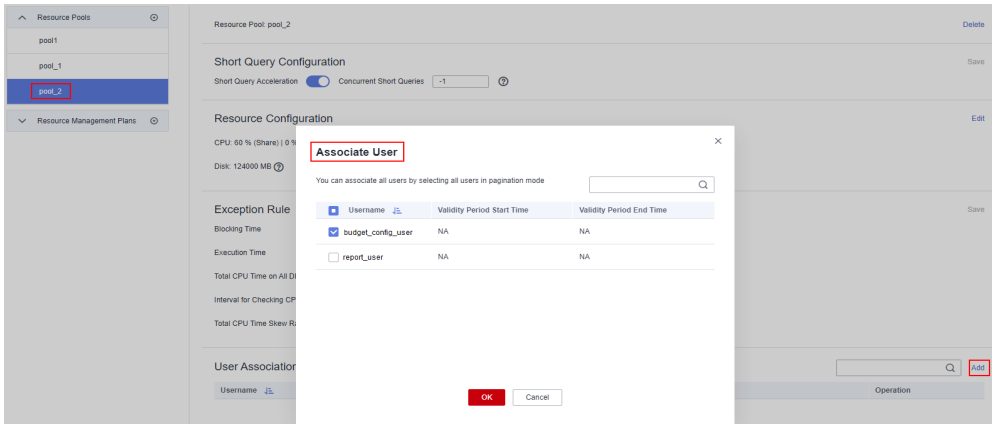
1. Clique em **pool_1** criado.
2. Na área**Exception Rule**, defina **Blocking Time** como 1200s e **Execution Time** como 1800s.
3. Clique em **Save**.
4. Repita as etapas anteriores para configurar **pool_2**.



Passo 4 Vincule usuários.

1. Clique em **pool_1** à esquerda.
2. Clique em **Add** à direita de **User Association**.
3. Selecione **report_user** e clique em **OK**.
4. Repita as etapas anteriores para adicionar **budget_config_user** a **pool_2**.





----Fim

Passo 4: verificar regras de exceção

Passo 1 Efetue logon no banco de dados como usuário **report_user**.

Passo 2 Execute o seguinte comando para verificar o pool de recursos ao qual pertence o usuário **report_user**:

```
SELECT usename,respool FROM pg_user WHERE usename = 'report_user';
```



```
gaussdb=> select usename,respool from pg_user where usename = 'report_user';
   usename   | respool
-------------+---------
 report_user | pool_1
(1 row)
```

O resultado da consulta mostra que o pool de recursos ao qual o usuário **report_user** pertence é **pool_1**.

Passo 3 Verifique a regra de exceção vinculada ao pool de recursos **pool_1**.

```
SELECT respool_name,mem_percent,active_statements,except_rule FROM
pg_resource_pool WHERE respool_name='pool_1';
```

```
gaussdb=> select respool_name,mem_percent,active_statements,except_rule from pg_resource_pool where respool_name='pool_1';
 respool_name | mem_percent | active_statements | except_rule
--------------+-------------+-------------------+-------------
 pool_1       |          20 |                20 | rule_1
(1 row)
```

Confirma-se que a regra de exceção **rule_1** está vinculada a **pool_1**.

Passo 4 Exiba o tipo de regra e o limite da regra de exceção para o usuário atual.

```
SELECT * FROM pg_except_rule WHERE name = 'rule_1';
```

```
gaussdb=> select * from pg_except_rule where name = 'rule_1';
  name  |    rule     | value
--------+-------------+-------
 rule_1 | action      | abort
 rule_1 | blocktime   | 1200
 rule_1 | elapsedtime | 1800
(3 rows)
```

O retorno mostra que a regra_1 tem 1200 segundos de tempo de bloco e 1800 segundos de duração de execução.

AVISO

- **PG_EXCEPT_RULE** registra informações sobre regras de exceção e é suportado apenas no cluster 8.2.0 ou posterior.
- A relação entre parâmetros na mesma regra de exceção é AND.

Passo 5 Quando o tempo de bloqueio de uma tarefa excede 1200s e a duração da execução excede 1800s, uma mensagem de erro é exibida, indicando que a regra de exceção é acionada e a tarefa é cancelada.

```
gaussdb=> insert into mytable select * from table1;
ERROR: canceling statement due to workload manager exception.
DETAIL: except rule [rule_1] is meet condition: rule [elapsedtime] is over limit. current value is: 1800. rule [blocktime] is over limit. current value is: 1200.
```

Se as informações de erro semelhantes a "ERROR: canceling statement due to workload manager exception." forem exibidas durante a execução do trabalho, o trabalho será encerrado porque excede o limite da regra de exceção. Se as regras não precisarem ser modificadas, você precisa otimizar as declarações de serviço para reduzir o tempo de execução.

Para obter detalhes sobre regras de exceção, consulte a seção **Regras de exceção**.

----Fim

5.2 Excelentes práticas para consultas SQL

Com base em um grande número de mecanismos e práticas de execução SQL, podemos otimizar instruções SQL seguindo certas regras para executar instruções SQL mais rapidamente e obter resultados corretos.

Para obter detalhes sobre o ajuste de SQL, consulte **Métodos típicos de otimização de SQL**.

5.3 Análise de instruções SQL que estão sendo executadas

Durante o desenvolvimento, os desenvolvedores geralmente encontram problemas como conexões SQL excessivas, tempo de consulta SQL longo e bloqueio de consulta SQL. Você pode usar as exibições **PG_STAT_ACTIVITY** and **PGXC_THREAD_WAIT_STATUS** para analisar e localizar problemas de SQL. Esta seção descreve alguns métodos comuns de localização.

Tabela 5-1 Alguns campos de PG_STAT_ACTIVITY

Nome	Tipo	Descrição
usename	name	Nome do usuário que efetua logon no back-end
client_addr	inet	O endereço IP do cliente conectado ao back-end **null** indica que o cliente está conectado por meio de um soquete Unix na máquina do servidor ou que este é um processo interno, como autovacuum.
application_name	text	Nome da aplicação conectada ao back-end

Nome	Tipo	Descrição
state	text	Estado geral do back-end. Os valores são: ● **active**: o back-end está executando consultas. ● **idle**: o back-end está aguardando novos comandos do cliente. ● **idle in transaction**: o backend está em uma transação, mas não há nenhuma instrução sendo executada na transação. ● **idle in transaction (aborted)**: o back-end está em uma transação, mas há declarações falhadas na transação. ● **fastpath function call**: o back-end está executando uma função **fast-path**. ● **disabled**: esse estado é relatado se **track_activities** estiver desabilitado nesse back-end. **NOTA** Os usuários comuns podem visualizar apenas o status da sessão de suas próprias contas. Ou seja, as informações de estado de outras contas estão vazias.
waiting	boolean	Se o back-end estiver atualmente esperando por um bloqueio, o valor será **t**. Caso contrário, o valor é **f**. ● **t** significa verdadeiro. ● **f** significa falso.

Nome	Tipo	Descrição
enqueue	text	Status de enfileiramento de uma instrução. Seu valor pode ser: ● **waiting in global queue**: a instrução está enfileirando na fila concorrente global. O número de instruções simultâneas excede o valor de **max_active_statements** configurado para um único CN. ● **waiting in respool queue**: a instrução está enfileirando no pool de recursos e a simultaneidade de trabalhos simples é limitada. A principal razão é que a simultaneidade de trabalhos simples excede o limite superior **max_dop** da via rápida. ● **waiting in ccn queue**: o trabalho está na fila CCN, que pode ser enfileiramento de memória global, enfileiramento de memória de faixa lenta ou enfileiramento simultâneo. Os cenários são: 1. A memória global disponível excede o limite superior, o trabalho está enfileirando na fila de memória global. 2. As solicitações simultâneas na pista lenta no pool de recursos excedem o limite superior, que é especificado por **active_statements**.

Nome	Tipo	Descrição
		3. A memória de faixa lenta do pool de recursos excede o limite superior, ou seja, a memória estimada de trabalhos simultâneos no pool de recursos excede o limite superior especificado por **mem_percent**. ● Vazio ou **no waiting queue**: A instrução está em execução.
pid	bigint	ID do thread de back-end.

Exibir informações de conexão

- Defina **track_activities** como **on**.
  ```
  SET track_activities = on;
  ```
  O banco de dados coleta as informações em execução sobre consultas ativas somente se esse parâmetro está definido como **on**.
- Você pode executar as seguintes instruções SQL para verificar o usuário de conexão atual, o endereço de conexão, a aplicação de conexão, o status, a espera de um bloqueio, o status do enfileiramento e o ID do thread.
  ```
  SELECT usename,client_addr,application_name,state,waiting,enqueue,pid FROM
  PG_STAT_ACTIVITY WHERE DATNAME='database name';
  ```
  A seguinte saída de comando é exibida:
  ```
   usename |  client_addr  | application_name | state  | waiting | enqueue
  |       pid
  ---------+---------------+------------------+--------+---------+---------
  +-----------------
   leo     | 192.168.0.133 | gsql             | idle   | f       |         |
  139666091022080
   dbadmin | 192.168.0.133 | gsql             | active | f       |         |
  139666212681472
   joe     | 192.168.0.133 |                  | idle   | f       |         |
  139665671489280
  (3 rows)
  ```
- Encerre uma sessão (somente o administrador do sistema tem a permissão).
  ```
  SELECT PG_TERMINATE_BACKEND(pid);
  ```

Exibir informações de execução de SQL

- Execute o seguinte comando para obter todas as informações SQL que o usuário atual tem permissão para exibir (se o usuário atual tiver permissão de administrador ou função predefinida, todas as informações de consulta do usuário poderão ser exibidas):
  ```
  SELECT usename,state,query FROM PG_STAT_ACTIVITY WHERE DATNAME='database
  name';
  ```
  Se o valor de state estiver active, a coluna de consulta indicará a instrução SQL que está sendo executada. Em outros casos, a coluna de consulta indica a instrução de consulta

anterior. Se o valor de state for idle, a conexão ficará ociosa e aguardará que o usuário insira um comando. A seguinte saída de comando é exibida:

```
 usename | state  |
query
---------+--------
+-------------------------------------------------------------------------------
 leo     | idle   | select * from joe.mytable;
 dbadmin | active | SELECT usename,state,query FROM PG_STAT_ACTIVITY WHERE
DATNAME='gaussdb';
 joe     | idle   | GRANT SELECT ON TABLE mytable to leo;
(3 rows)
```

- Execute o seguinte comando para exibir as informações sobre as instruções SQL que não estão no estado ocioso:

```
SELECT datname,usename,query FROM PG_STAT_ACTIVITY WHERE state != 'idle' ;
```

Exibir instruções que consomem muito tempo

- Verifique as instruções SQL que levam muito tempo para serem executadas.

```
SELECT current_timestamp - query_start as runtime, datname, usename, query
FROM PG_STAT_ACTIVITY WHERE state != 'idle' order by 1 desc;
```

As instruções de consulta são retornadas e classificadas por duração de tempo de execução em ordem decrescente. O primeiro registro é a instrução de consulta que leva mais tempo para ser executada.

```
     runtime     | datname  | usename
|
query
-----------------+----------+---------
+-------------------------------------------------------------------------------
-------------------------------------------------------------
 00:04:47.054958 | gaussdb  | leo     | insert into mytable1 select
generate_series(1, 10000000);
 00:00:01.72789  | gaussdb  | dbadmin | SELECT current_timestamp -
query_start as runtime, datname, usename, query FROM PG_STAT_ACTIVITY WHERE
state != 'idle' order by 1 desc;
(2 rows)
```

- Como alternativa, você pode definir **current_timestamp - query_start** para ser maior que um limite para identificar instruções de consulta que são executadas por um período maior que esse limite.

```
SELECT query from PG_STAT_ACTIVITY WHERE current_timestamp - query_start >
interval '2 days';
```

Consultar instruções bloqueadas

- Execute o seguinte comando para exibir instruções de consulta bloqueadas:

```
SELECT pid, datname, usename, state, query FROM PG_STAT_ACTIVITY WHERE state
<> 'idle' and waiting=true;
```

Execute a instrução a seguir para encerrar a sessão SQL bloqueada:

```
SELECT PG_TERMINATE_BACKEND(pid);
```

NOTA

- Na maioria dos casos, o bloqueio é causado por bloqueios internos e **waiting=true** é exibido. Você pode ver o bloqueio na exibição **pg_stat_activity**.
- As instruções bloqueadas sobre a gravação de arquivos e agendadores de eventos não podem ser visualizadas na exibição **pg_stat_activity**.

- Exiba informações sobre as instruções de consulta bloqueadas, tabelas e esquemas.

```
SELECT w.query as waiting_query,
w.pid as w_pid,
w.usename as w_user,
l.query as locking_query,
```



```
l.pid as l_pid,
l.usename as l_user,
t.schemaname || '.' || t.relname as tablename
from pg_stat_activity w join pg_locks l1 on w.pid = l1.pid
and not l1.granted join pg_locks l2 on l1.relation = l2.relation
and l2.granted join pg_stat_activity l on l2.pid = l.pid join
pg_stat_user_tables t on l1.relation = t.relid
where w.waiting;
```

A saída do comando inclui um ID de sessão, informações do usuário, status da consulta e tabela ou esquema que causou o bloqueio.

Depois de encontrar a tabela bloqueada ou informações de esquema, encerre a sessão defeituosa.

```
SELECT PG_TERMINATE_BACKEND(pid);
```

Se informações semelhantes às seguintes forem exibidas, a sessão será encerrada com êxito:

```
PG_TERMINATE_BACKEND
----------------------
t
(1 row)
```

Se informações semelhantes às seguintes forem exibidas, o usuário está tentando encerrar a sessão, mas a sessão será reconectada em vez de encerrada.

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

NOTA

Se a função **PG_TERMINATE_BACKEND** for usada pelo cliente de gsql para encerrar os threads em segundo plano da sessão, o cliente será reconectado automaticamente em vez de ser encerrado.

5.4 Excelentes práticas para consultas de distorção de dados

5.4.1 Detecção em tempo real de distorção de armazenamento durante a importação de dados

Durante a importação, o sistema coleta estatísticas sobre o número de linhas importadas em cada DN. Após a conclusão da importação, o sistema calcula a taxa de distorção. Se a taxa de distorção exceder o limite especificado, um alarme é gerado imediatamente. A taxa de distorção é calculada da seguinte forma: Taxa de distorção = (número máximo de linhas importadas em um DN – número mínimo de linhas importadas em um DN)/número de linhas importadas. Atualmente, os dados podem ser importados apenas executando **INSERT** ou **COPY**.

NOTA

enable_stream_operator deve ser definido como **on** para que os DNs possam retornar o número de linhas importadas no momento em que um plano for entregue a eles. Em seguida, a taxa de distorção é calculada no CN com base nos valores devolvidos.

Uso

1. Definir parâmetros **table_skewness_warning_threshold** (limite para acionar um alarme de distorção da tabela) e **table_skewness_warning_rows** (número mínimo de linhas para disparar um alarme de distorção de tabela).
   - O valor de **table_skewness_warning_threshold** varia de **0** a **1**. O valor padrão é **1**, indicando que o alarme está desativado. Outros valores indicam que o alarme está ativado.
   - O valor de **table_skewness_warning_rows** varia de **0** a **2147483647**. O valor padrão é **100.000**. O alarme é disparado somente quando a seguinte condição é atendida: número total de linhas importadas > valor de **table_skewness_warning_rows** x número de DNs envolvidos na importação.

   ```
   show table_skewness_warning_threshold;
   set table_skewness_warning_threshold = xxx;
   show table_skewness_warning_rows;
   set table_skewness_warning_rows = xxx;
   ```

2. Importe dados executando a instrução **INSERT** ou **COPY**.
3. Detecte e lide com alarmes. As informações de alarme incluem o nome da tabela, número mínimo de linhas, número máximo de linhas, número total de linhas, número médio de linhas, taxa de distorção e informações de prompt sobre distribuição de dados ou modificação de parâmetros.

   ```
   WARNING:  Skewness occurs, table name: xxx, min value: xxx, max value: xxx,
   sum value: xxx, avg value: xxx, skew ratio: xxx
   HINT:  Please check data distribution or modify warning threshold
   ```

5.4.2 Localização rápida das tabelas que causam distorção de dados

Atualmente, as seguintes APIs de consulta são fornecidas:**table_distribution(schemaname text, tablename text)**, **table_distribution()** e **PGXC_GET_TABLE_SKEWNESS**. Você pode selecionar uma com base nos requisitos do serviço.

Cenário 1: distorção de dados causado por um disco cheio

Primeiro, use a função **pg_stat_get_last_data_changed_time(oid)** para consultar as tabelas cujos dados foram alterados recentemente. O último tempo de mudança de uma tabela é registrado apenas no CN onde as operações **INSERT**, **UPDATE** e **DELETE** são executadas. Portanto, você precisa consultar tabelas que são alteradas no último dia (o período pode ser alterado na função).

```
CREATE OR REPLACE FUNCTION get_last_changed_table(OUT schemaname text, OUT
relname text)
RETURNS setof record
AS $$
DECLARE
row_data record;
row_name record;
query_str text;
query_str_nodes text;
BEGIN
query_str_nodes := 'SELECT node_name FROM pgxc_node where node_type = ''C''';
FOR row_name IN EXECUTE(query_str_nodes) LOOP
query_str := 'EXECUTE DIRECT ON (' || row_name.node_name || ') ''SELECT
b.nspname,a.relname FROM pg_class a INNER JOIN pg_namespace b on a.relnamespace =
b.oid where pg_stat_get_last_data_changed_time(a.oid) BETWEEN current_timestamp -
1 AND current_timestamp;''';
FOR row_data IN EXECUTE(query_str) LOOP
schemaname = row_data.nspname;
```



```
relname = row_data.relname;
return next;
END LOOP;
END LOOP;
return;
END; $$
LANGUAGE plpgsql;
```

Em seguida, execute a função **table_distribution(schemaname text, tablename text)** para consultar o espaço de armazenamento ocupado pelas tabelas em cada DN.

```
SELECT table_distribution(schemaname,relname) FROM get_last_changed_table();
```

Cenário 2: inspeção de distorção de dados de rotina

- Se o número de tabelas no banco de dados for menor que 10.000 use a exibição **PGXC_GET_TABLE_SKEWNESS** para consultar a distorção de dados de todas as tabelas no banco de dados.
  ```
  SELECT * FROM pgxc_get_table_skewness ORDER BY totalsize DESC;
  ```
- Se o número de tabelas no banco de dados não for menor que 10.000 é aconselhável usar a função **table_distribution()** em vez da exibição **PGXC_GET_TABLE_SKEWNESS** porque a exibição leva mais tempo (horas) devido à consulta de todo o banco de dados para colunas distorcidas. Quando você usa a função **table_distribution()**, você pode definir a saída baseada em **PGXC_GET_TABLE_SKEWNESS**, otimizando o cálculo e reduzindo as colunas de saída. Por exemplo:
  ```
  SELECT schemaname,tablename,max(dnsize) AS maxsize, min(dnsize) AS minsize
  FROM pg_catalog.pg_class c
  INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
  INNER JOIN pg_catalog.table_distribution() s ON s.schemaname = n.nspname AND
  s.tablename = c.relname
  INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype =
  'H'
  GROUP BY schemaname,tablename;
  ```

Cenário 3: consultar distorção de dados de uma tabela

Execute a seguinte instrução SQL para consultar a distorção de dados de uma tabela. Substitua **table_name** pelo nome real da tabela.

```
SELECT a.count,b.node_name FROM (SELECT count(*) AS count,xc_node_id FROM
table_name GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER
BY a.count desc;
```

Segue-se um exemplo das informações retornadas. Se o desvio de distribuição de dados em cada DN for inferior a 10%, os dados serão distribuídos uniformemente. Se for maior que 10%, ocorrerá uma distorção de dados.

```
gaussdb=>SELECT a.count,b.node_name FROM (select count(*) as count,xc_node_id
FROM staffs GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id
ORDER BY a.count desc;
count | node_name
------+-----------
11010 | datanode4
10000 | datanode3
12001 | datanode2
 8995 | datanode1
10000 | datanode5
 7999 | datanode6
 9995 | datanode7
10000 | datanode8
(8 rows)
```


5.5 Melhores práticas para gerenciamento de usuários

Um cluster do GaussDB(DWS) consiste principalmente de administradores de sistema e usuários comuns. Esta seção descreve as permissões de administradores de sistema e usuários comuns e descreve como criar usuários e consultar informações do usuário.

Administrador do sistema

O usuário **dbadmin** criado quando você inicia um cluster do GaussDB(DWS) é um administrador do sistema. Ele tem a mais alta permissão do sistema e pode executar todas as operações, incluindo operações em tablespaces, tabelas, índices, esquemas, funções e visualizações personalizadas, bem como consultar catálogos e visualizações do sistema.

Para criar um administrador de banco de dados, conecte-se ao banco de dados como administrador e execute a instrução **CREATE USER** ou**ALTER USER** com **SYSADMIN** especificado.

Exemplos:

Crie o usuário **Jim** como administrador do sistema.

```
CREATE USER Jim WITH SYSADMIN password '{Password}';
```

Altere o usuário **Tom** para um administrador do sistema. **ALTER USER** pode ser usado apenas para usuários existentes.

```
ALTER USER Tom SYSADMIN;
```

Usuário comum

Você pode executar a instrução SQL **CREATE USER** para criar um usuário comum. Um usuário comum não pode criar, modificar, deletar ou designar tablespaces e precisa receber a permissão para acessar tablespaces. Um usuário comum tem todas as permissões para suas próprias tabelas, esquemas, funções e exibições personalizadas, cria índices em suas próprias tabelas e consulta apenas alguns catálogos e exibições do sistema.

O cluster de banco de dados tem um ou mais bancos de dados nomeados. Os usuários são compartilhados dentro de todo o cluster, mas seus dados não são compartilhados.

As operações comuns do usuário são as seguintes. Substitua **password** pela senha real.

1. Criar um usuário
   ```
   CREATE USER Tom PASSWORD '{Password}';
   ```
2. Alterar a senha de um usuário

   Altere a senha de logon do usuário **Tom** de **password** para **newpassword**.
   ```
   ALTER USER Tom IDENTIFIED BY 'newpassword' REPLACE '{Password}';
   ```
3. Atribuir permissões a um usuário
   - Adicione **CREATEDB** quando você cria um usuário que tem a permissão para criar um banco de dados.
     ```
     CREATE USER Tom CREATEDB PASSWORD '{Password}';
     ```
   - Adicione a permissão **CREATEROLE** para um usuário.
     ```
     ALTER USER Tom CREATEROLE;
     ```
4. Revogar permissões de usuário
   ```
   REVOKE ALL PRIVILEGES FROM Tom;
   ```


5. Bloquear ou desbloquear um usuário
   – Bloqueie o usuário **Tom**.
   ```
   ALTER USER Tom ACCOUNT LOCK;
   ```
   – Desbloqueie o usuário **Tom**.
   ```
   ALTER USER Tom ACCOUNT UNLOCK;
   ```
6. Excluir um usuário
   ```
   DROP USER Tom CASCADE;
   ```

Consulta das informações de usuário

As exibições do sistema relacionadas a usuários, funções e permissões incluem **ALL_USERS**, **PG_USER** e **PG_ROLES**, e os catálogos do sistema incluem **PG_AUTHID** e **PG_AUTH_MEMBERS**.

- **ALL_USERS** exibe todos os usuários no banco de dados, mas não mostra os detalhes deles.
- **PG_USER** exibe informações do usuário, incluindo IDs de usuário, a permissão para criar bancos de dados e pools de recursos.
- **PG_ROLES** exibe informações sobre as atribuições do banco de dados.
- **PG_AUTHID** registra informações sobre identificadores de autenticação de banco de dados (funções), incluindo permissões de função para efetuar logon ou criar bancos de dados.
- **PG_AUTH_MEMBERS** armazena informações de funções contidas em um grupo de funções.

1. Você pode executar **PG_USER** para consultar todos os usuários no banco de dados. O ID de usuário (**USESYSID**) e as permissões também podem ser consultados.
   ```
   SELECT * FROM pg_user;
    usename | usesysid | usecreatedb | usesuper | usecatupd | userepl |  passwd
   | valbegin | valuntil |   respool    | parent | spacelimit | useconfig |
   nodegroup | tempspacelimit | spillspacelim
   it
   ---------+----------+-------------+----------+-----------+---------+----------
   +----------+----------+--------------+--------+------------+-----------
   +-----------+----------------+--------------
   ---
    Ruby     |       10 | t           | t        | t         | t       | ********
   |          |          | default_pool |      0 |            |
   |           |                |
    kim      |    21661 | f           | f        | f         | f       | ********
   |          |          | default_pool |      0 |            |
   |           |                |
    u3       |    22662 | f           | f        | f         | f       | ********
   |          |          | default_pool |      0 |            |
   |           |                |
    u1       |    22666 | f           | f        | f         | f       | ********
   |          |          | default_pool |      0 |            |
   |           |                |
    dbadmin  |    16396 | f           | f        | f         | f       | ********
   |          |          | default_pool |      0 |            |
   |           |                |
    u5       |    58421 | f           | f        | f         | f       | ********
   |          |          | default_pool |      0 |            |
   |           |                |
   (6 rows)
   ```
2. **ALL_USERS** exibe todos os usuários no banco de dados, mas não mostra os detalhes deles.
   ```
   SELECT * FROM all_users;
    username | user_id
   ```



```
----------+---------
 Ruby     |      10
 manager  |   21649
 kim      |   21661
 u3       |   22662
 u1       |   22666
 u2       |   22802
 dbadmin  |   16396
 u5       |   58421
(8 rows)
```

3. **PG_ROLES** armazena informações sobre funções que acessaram o banco de dados.

```
SELECT * FROM pg_roles;
 rolname | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcatupdate
| rolcanlogin | rolreplication | rolauditadmin | rolsystemadmin |
rolconnlimit | rolpassword | rolvalidbegin | rolv
aliduntil |  rolrespool  | rolparentid | roltabspace | rolconfig |  oid  |
roluseft | rolkind | nodegroup | roltempspace | rolspillspace
---------+----------+------------+---------------+-------------+--------------
+-------------+----------------+---------------+----------------
+--------------+-------------+---------------+-----
----------+--------------+-------------+-------------+-----------+-------
+----------+---------+-----------+--------------+---------------
 Ruby    | t        | t          | t             | t           | t
| t           | t             | t             | t              |
-1 | ********    |            |
          | default_pool |           0 |             |           |    10 |
t        | n       |           |               |
 manager | f        | t          | f             | f           | f
| f           | f             | f             | f              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 21649 |
f        | n       |           |               |
 kim     | f        | t          | f             | f           | f
| t           | f             | f             | f              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 21661 |
f        | n       |           |               |
 u3      | f        | t          | f             | f           | f
| t           | f             | f             | f              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 22662 |
f        | n       |           |               |
 u1      | f        | t          | f             | f           | f
| t           | f             | f             | f              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 22666 |
f        | n       |           |               |
 u2      | f        | t          | f             | f           | f
| f           | f             | f             | f              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 22802 |
f        | n       |           |               |
 dbadmin | f        | t          | f             | f           | f
| t           | f             | f             | t              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 16396 |
f        | n       |           |               |
 u5      | f        | t          | f             | f           | f
| t           | f             | f             | f              |
-1 | ********    |            |
          | default_pool |           0 |             |           | 58421 |
f        | n       |           |               |
(8 rows)
```

4. Para exibir as propriedades do usuário, consulte o catálogo do sistema **PG_AUTHID**, que armazena informações sobre identificadores de autorização do banco de dados (funções). Cada cluster, e não cada banco de dados, tem apenas um catálogo do sistema

PG_AUTHID. Somente usuários com permissões de administrador do sistema podem acessar o catálogo.

```
SELECT * FROM pg_authid;
rolname  | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcatupdate
| rolcanlogin | rolreplication | rolauditadmin | rolsystemadmin |
rolconnlimit
|


rolpassword

                                      | rolvalidbegin | rolvaliduntil |
rolrespool  | roluseft | rolparentid | roltabspace | rolkind | rolnodegroup |
roltempspace | rolspillspace | rolexcpdata | rolauthinfo
----------+----------+------------+---------------+-------------
+--------------+-------------+----------------+---------------
+----------------+--------------
+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
+---------------+---------------+--------------+----------+-------------
+--------------+---------+--------------+--------------+---------------
+--------------+-------------
 Ruby     | t        | t          | t             | t           |
t            | t           | t              | t             | t
|           -1 |
sha256366f1e665be208e6015bc3c5795d13e4dc297a148dca6c60346018c80e5c04c9ba170384
ce44609b31baa741f09a3ea5bedc7dadb906286ca994067c3fbf672dc08c981929e326ca08c005
d8df942994e146ed3302af47000b36e9852b50e39dmd585de11aafebd90ec620b201fc36f07a5e
cdficefade3a1456ec0aca9a0ee01e3bf2971d1dbafd604e596149e2e2928be4060dec2bd86887
76588b4cd8c64fd38f1b0beab1603129fa396556ba8aa4c7d6e137a04623 |
|               | default_pool | t        |           0 |             |
n       |           0 |              |               |             |
 sysadmin | f        | t          | f             | f           |
f            | t           | f              | f             | t
|           -1 |
sha256ecaa7f0ca4436143af43074f16cdd825783ad1a5d659fd94f5e2fa5124e7da44045ecf40
bda1a97975fcf5920dca0c8be375be5c71b51cb1eeeba0851fb3648cfa49f55989f83fd9baf1a9
d5853ce19125f4fc29a7c709c095ed02d00638410dmd556d6e2dcc41594dc7ad8ee909ef81637e
cdficefadefd7d9704ee06affef9581cd6a50a546607f88891198e96a5e84e7e83dccf56c5cd20
a500bbc5248e8ea51f0bca70c5a8dcf00953f8b62c7a181368153abce760 |
|               | default_pool | f        |           0 |             |
n       |             |              |               |             |
 Tom      | f        | t          | f             | t           |
f            | t           | f              | f             | f
|           -1 |
sha256f43c4f52ac51e297bc4dbdbc751fcf05319c15681dbf5a9c5777d2edce45cb592a948b25
457a728e99a3e0608592f33b0a4312eba6124936522304ba298caa2002a04578860fecb0286d7c
7baec09365eafd049b2b99f74f21a08864dd7d3f2amd515ee49f0b18ef8e7d0cd27d91ce2fa9de
cdficefade16bab5f05b6d7c86a19ae6406cc59c437506c3f6187bfdf3eefc7a7c7033afa07636
1b255cc8b6ccb6e19d4767effaec654b3308cc72cebb891d00a4a10362da |
|               | default_pool | f        |           0 |             |
n       |             |              |               |             |
(3 rows)
```

Consulta de recursos do usuário

1. Consultar a cota de recursos e o uso de todos os usuários

   ```
   SELECT * FROM PG_TOTAL_USER_RESOURCE_INFO;
   ```

   Exemplo do uso de recursos de todos os usuários:

   ```
    username | used_memory | total_memory | used_cpu | total_cpu | used_space |
   total_space | used_temp_space | total_temp_space | used_spill_space |
   total_spill_space | read_kbytes | write_kbytes | read_counts | write_counts |
   read_speed | write_speed
   ----------+-------------+--------------+----------+-----------+------------
   +-------------+-----------------+------------------+------------------
   ```



```
+------------------+------------+--------------+------------+-------------
+------------+-------------
perfadm   |           0 |        17250 |         0 |          0 |          0
|          -1 |               0 |                -1 |                0
|                -1 |            0 |             0 |            0 |            0
|          0 |           0
usern     |           0 |        17250 |         0 |         48 |          0
|          -1 |               0 |                -1 |                0
|                -1 |            0 |             0 |            0 |            0
|          0 |           0
userg     |          34 |        15525 |    23.53 |         48 |          0
|          -1 |               0 |                -1 |         814955731
|                -1 |     6111952 |      1145864 |      763994 |       143233
|      42678 |        8001
userg1    |          34 |        13972 |    23.53 |         48 |          0
|          -1 |               0 |                -1 |         814972419
|                -1 |     6111952 |      1145864 |      763994 |       143233
|      42710 |        8007
(4 rows)
```

2. Consultar a cota de recursos e o uso de um usuário especificado

   ```
   SELECT * FROM GS_WLM_USER_RESOURCE_INFO('username');
   ```

   Exemplo do uso de recursos do usuário **Tom**:

   ```
   SELECT * FROM GS_WLM_USER_RESOURCE_INFO('Tom');
   userid | used_memory | total_memory | used_cpu | total_cpu | used_space |
   total_space | used_temp_space | total_temp_space | used_spill_space |
   total_spill_space | read_kbytes | write_kbytes | read_counts | write_counts |
   read_speed | write_speed
   -------+-------------+--------------+----------+-----------+------------
   +-------------+-----------------+------------------+------------------
   +-------------------+-------------+--------------+-------------+--------------
   +------------+-------------
    16523 |          18 |         2831 |        0 |        19 |          0
   |          -1 |               0 |                -1 |                0
   |               -1 |            0 |             0 |            0 |            0
   |          0 |           0
   (1 row)
   ```

3. Consultar o uso de I/O de um usuário especificado

   ```
   SELECT * FROM pg_user_iostat('username');
   ```

   Exemplo do uso de I/O do usuário **Tom**:

   ```
   SELECT * FROM pg_user_iostat('Tom');
   userid | min_curr_iops | max_curr_iops | min_peak_iops | max_peak_iops |
   io_limits | io_priority
   -------+---------------+---------------+---------------+---------------
   +-----------+-------------
    16523 |             0 |             0 |             0 |             0
   |         0 | None
   (1 row)
   ```

5.6 Exibição de informações sobre tabela e banco de dados

Consultar informações da tabela

- Consultar informações sobre todas as tabelas em um banco de dados usando o catálogo do sistema **pg_tables**

  ```
  SELECT * FROM pg_tables;
  ```

- Consultar a estrutura da tabela usando o comando **\d+** da ferramenta **gsql**.

  Exemplo: crie uma tabela **customer_t1** e insira dados na tabela.

  ```
  CREATE TABLE customer_t1
  (
      c_customer_sk             integer,
      c_customer_id             char(5),
      c_first_name              char(6),
  ```



```
    c_last_name               char(8)
)
with (orientation = column,compression=middle)
distribute by hash (c_last_name);
INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES
    (6885, 'map', 'Peter'),
    (4321, 'river', 'Lily'),
    (9527, 'world', 'James');
```

Consulte a estrutura da tabela. Se nenhum esquema for especificado quando você criar uma tabela, o esquema da tabela assumirá como padrão **public**.

```
\d+ customer_t1;
                        Table "public.customer_t1"
    Column     |     Type     | Modifiers | Storage  | Stats target |
Description
---------------+--------------+-----------+----------+--------------
+-------------
 c_customer_sk | integer      |           | plain    |              |
 c_customer_id | character(5) |           | extended |              |
 c_first_name  | character(6) |           | extended |              |
 c_last_name   | character(8) |           | extended |              |
Has OIDs: no
Distribute By: HASH(c_last_name)
Location Nodes: ALL DATANODES
Options: orientation=column, compression=middle, colversion=2.0,
enable_delta=false
```

NOTA

As opções podem variar em versões diferentes, mas a diferença não afeta os serviços. As opções aqui são apenas para referência. As opções reais estão sujeitas à versão.

- Use **pg_get_tabledef** para consultar a definição da tabela.

```
SELECT * FROM PG_GET_TABLEDEF('customer_t1');

pg_get_tabledef
--------------------------------------------------------------------------------
-----
 SET search_path =
tpchobs;                                                        +
 CREATE  TABLE customer_t1
(                                                             +
         c_customer_sk
integer,                                                        +
         c_customer_id
character(5),                                                   +
         c_first_name
character(6),                                                    +
         c_last_name
character(8)                                                      +
 )
    +
 WITH (orientation=column, compression=middle, colversion=2.0,
enable_delta=false)+
 DISTRIBUTE BY
HASH(c_last_name)                                                       +
 TO GROUP group_version1;
(1 row)
```

- Consultar todos os dados em **customer_t1**

```
SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          6885 | map           | Peter        |
          4321 | river         | Lily         |
          9527 | world         | James        |
(3 rows)
```

- Consultar todos os dados de uma coluna em **customer_t1** usando **SELECT**


```
SELECT c_customer_sk FROM customer_t1;
 c_customer_sk
---------------
          6885
          4321
          9527
(3 rows)
```

- Verifique se uma tabela foi analisada. A hora em que a tabela foi analisada será devolvida. Se nada for retornado, isso indica que a tabela não foi analisada.

  ```
  SELECT pg_stat_get_last_analyze_time(oid),relname FROM pg_class where
  relkind='r';
  ```

  Consulte a hora em que a tabela **public** foi analisada.

  ```
  SELECT pg_stat_get_last_analyze_time(c.oid),c.relname FROM pg_class c LEFT
  JOIN pg_namespace n ON c.relnamespace = n.oid WHERE c.relkind='r' AND
  n.nspname='public';
   pg_stat_get_last_analyze_time |       relname
  -------------------------------+----------------------
   2022-05-17 07:48:26.923782+00 | warehouse_t19
   2022-05-17 07:48:26.964512+00 | emp
   2022-05-17 07:48:27.016709+00 | test_trigger_src_tbl
   2022-05-17 07:48:27.045385+00 | customer
   2022-05-17 07:48:27.062486+00 | warehouse_t1
   2022-05-17 07:48:27.114884+00 | customer_t1
   2022-05-17 07:48:27.172256+00 | product_info_input
   2022-05-17 07:48:27.197014+00 | tt1
   2022-05-17 07:48:27.212906+00 | timezone_test
  (9 rows)
  ```

- Consulte rapidamente as informações da coluna de uma tabela. Se uma visão em **information_schema** tiver um grande número de objetos no banco de dados, levará muito tempo para retornar o resultado. Você pode executar a seguinte instrução SQL para consultar rapidamente as informações da coluna de uma ou mais tabelas:

  ```
  SELECT /*+ nestloop(a c)*/ c.column_name, c.data_type, c.ordinal_position,
  pgd.description, pp.partkey, c.is_nullable, c.column_default,
  c.character_maximum_length, c.numeric_precision,  c.numeric_scale,
  c.datetime_precision, c.interval_type, c.udt_name from
  information_schema.columns as c left join pg_namespace sp on sp.nspname =
  c.table_schema left join pg_class cla on cla.relname = c.table_name and
  cla.relnamespace = sp.oid left join pg_catalog.pg_partition pp on
  (pp.parentid = cla.oid and pp.parttype = 'r') left join
  pg_catalog.pg_description pgd on (pgd.objoid=cla.oid and pgd.objsubid =
  c.ordinal_position)where c.table_name in ('tablename') and c.table_schema =
  'public';
  ```

  Por exemplo, para consultar rapidamente as informações da coluna da tabela **customer_t1**, execute o seguinte comando:

  ```
  SELECT /*+ nestloop(a c)*/ c.column_name, c.data_type, c.ordinal_position,
  pgd.description, pp.partkey, c.is_nullable, c.column_default,
  c.character_maximum_length, c.numeric_precision, c.numeric_scale,
  c.datetime_precision, c.interval_type, c.udt_name from
  information_schema.columns as c left join pg_namespace sp on sp.nspname =
  c.table_schema left join pg_class cla on cla.relname = c.table_name and
  cla.relnamespace = sp.oid left join pg_catalog.pg_partition pp on
  (pp.parentid = cla.oid and pp.parttype = 'r') left join
  pg_catalog.pg_description pgd on (pgd.objoid=cla.oid and pgd.objsubid =
  c.ordinal_position) where c.table_name in ('customer_t1') and c.table_schema
  = 'public';
    column_name  | data_type | ordinal_position | description | partkey |
  is_nullable | column_default | character_maximum_length | numeric_precision |
  numeric_scale | datetime_precision | interval_type | udt_name
  ---------------+-----------+------------------+-------------+---------
  +-------------+----------------+--------------------------+-------------------
  +-------------
  --+--------------------+---------------+----------
   c_last_name   | character |                4 |             |         |
  YES         |                |                        8 |                   |
    |                   |               | bpchar
  ```



```
 c_first_name  | character |                3 |             |         |
YES         |               |                        6 |                    |
  |                    |               | bpchar
 c_customer_id | character |                2 |             |         |
YES         |               |                        5 |                    |
  |                    |               | bpchar
 c_customer_sk | integer   |                1 |             |         |
YES         |               |                          |               32 |
0 |                    |               | int4
(4 rows)
```

- Obtenha a definição da tabela consultando logs de auditoria.

  Use a função **pgxc_query_audit** para consultar logs de auditoria de todos os CNs. A sintaxe é a seguinte:

  ```
  pgxc_query_audit(timestamptz startime,timestamptz endtime)
  ```

  Consulte os registros de auditoria de vários objetos.

  ```
  SET audit_object_name_format TO 'all';
  SELECT object_name,result,operation_type,command_text FROM
  pgxc_query_audit('2022-08-26 8:00:00','2022-08-26 22:55:00') where
  command_text like '%student%';
  ```

Consultar o tamanho da tabela

- Consultar o tamanho total de uma tabela (índices e dados incluídos)

  ```
  SELECT pg_size_pretty(pg_total_relation_size('<schemaname>.<tablename>'));
  ```

  Exemplo:

  Primeiro, crie um índice em **customer_t1**.

  ```
  CREATE INDEX index1 ON customer_t1 USING btree(c_customer_sk);
  ```

  Em seguida, consulte o tamanho da tabela **customer_t1** de **public**.

  ```
  SELECT pg_size_pretty(pg_total_relation_size('public.customer_t1'));
   pg_size_pretty
  ----------------
   264 kB
  (1 row)
  ```

- Consultar o tamanho de uma tabela (índices excluídos)

  ```
  SELECT pg_size_pretty(pg_relation_size('<schemaname>.<tablename>'));
  ```

  Exemplo: consulte o tamanho da tabela **customer_t1** de **public**.

  ```
  SELECT pg_size_pretty(pg_relation_size('public.customer_t1'));
   pg_size_pretty
  ----------------
   208 kB
  (1 row)
  ```

- Consulte todas as tabelas, classificadas por seu espaço ocupado.

  ```
  SELECT table_schema || '.' || table_name AS table_full_name,
  pg_size_pretty(pg_total_relation_size('"' || table_schema || '"."' ||
  table_name || '"')) AS size FROM information_schema.tables
  ORDER BY
  pg_total_relation_size('"' || table_schema || '"."' || table_name || '"')
  DESC limit xx;
  ```

  Exemplo 1: consulte as 15 tabelas que ocupam mais espaço.

  ```
  SELECT table_schema || '.' || table_name AS table_full_name,
  pg_size_pretty(pg_total_relation_size('"' || table_schema || '"."' ||
  table_name || '"')) AS size FROM information_schema.tables
  ORDER BY
  pg_total_relation_size('"' || table_schema || '"."' || table_name || '"')
  DESC limit 15;
        table_full_name      |  size
  ---------------------------+---------
   pg_catalog.pg_attribute   | 2048 KB
   pg_catalog.pg_rewrite     | 1888 KB
   pg_catalog.pg_depend      | 1464 KB
  ```



```
 pg_catalog.pg_proc         | 1464 KB
 pg_catalog.pg_class        | 512 KB
 pg_catalog.pg_description  | 504 KB
 pg_catalog.pg_collation    | 360 KB
 pg_catalog.pg_statistic    | 352 KB
 pg_catalog.pg_type         | 344 KB
 pg_catalog.pg_operator     | 224 KB
 pg_catalog.pg_amop         | 208 KB
 public.tt1                 | 160 KB
 pg_catalog.pg_amproc       | 120 KB
 pg_catalog.pg_index        | 120 KB
 pg_catalog.pg_constraint   | 112 KB
(15 rows)
```

Exemplo 2: consulte as 20 principais tabelas com o maior uso de espaço no esquema **public**.

```
SELECT table_schema || '.' || table_name AS table_full_name,
pg_size_pretty(pg_total_relation_size('"' || table_schema || '"."' ||
table_name || '"')) AS size FROM information_schema.tables where
table_schema='public'
ORDER BY
pg_total_relation_size('"' || table_schema || '"."' || table_name || '"')
DESC limit 20;
       table_full_name       |  size
-----------------------------+---------
 public.tt1                  | 160 KB
 public.product_info_input   | 112 KB
 public.customer_t1          | 96 KB
 public.warehouse_t19        | 48 KB
 public.emp                  | 32 KB
 public.customer             | 0 bytes
 public.test_trigger_src_tbl | 0 bytes
 public.warehouse_t1         | 0 bytes
(8 rows)
```

Consultar rapidamente o espaço ocupado por todas as tabelas no banco de dados

Em um cluster grande com uma grande quantidade de dados (mais de 1000 tabelas), é aconselhável usar o modo de exibição pgxc_wlm_table_distribution_skewness para consultar todas as tabelas no banco de dados. Essa exibição pode ser usada para consultar o uso do tablespace e a distribuição de desvio de dados no banco de dados. A unidade de total_size e avg_size é byte.

```
SELECT *, pg_size_pretty(total_size) as tableSize FROM
pgxc_wlm_table_distribution_skewness ORDER BY total_size desc;
    schema_name     |                    table_name                     |
total_size | avg_size  | max_percent | min_percent | skew_percent | tablesize
-------------------+---------------------------------------------------
+------------+-----------+-------------+-------------+--------------+-----------
 public             | history_tbs_test_row_1                            |
804347904 | 134057984 |       18.02 |       15.63 |         7.53 | 767 MB
 public             | history_tbs_test_row_3                            |
402096128 |  67016021 |       18.30 |       15.60 |         8.90 | 383 MB
 public             | history_tbs_test_row_2                            |
401743872 |  66957312 |       18.01 |       15.01 |         7.47 | 383 MB
 public             | i_history_tbs_test_1                              |
325263360 |  54210560 |       17.90 |       15.50 |         6.90 | 310 MB
```

O resultado da consulta mostra que a tabela history_tbs_test_row_1 ocupa o maior espaço e ocorre uma distorção de dados.

⚠ CUIDADO

1. A exibição pgxc_wlm_table_distribution_skewness só pode ser consultada quando os parâmetros de GUC use_workload_manager e enable_perm_space estão ativados. Em versões anteriores, você é aconselhado a usar a função table_distribution() para consultar o banco de dados inteiro. Se apenas o tamanho de uma tabela for consultado, a função table_distribution(schemaname text, tablename text) é recomendada.
2. Em versões de cluster 8.2.1 e posteriores, o GaussDB(DWS) suporta a visualização pgxc_wlm_table_distribution_skewness, que pode ser consultada diretamente.
3. Na versão de cluster 8.1.3, você pode usar a seguinte definição para criar um modo de exibição e, em seguida, consultar o modo de exibição:

```
CREATE OR REPLACE VIEW
pgxc_wlm_table_distribution_skewness AS
WITH skew AS
(
SELECT
schemaname,
tablename,
pg_catalog.sum(dnsize)
AS totalsize,
pg_catalog.avg(dnsize)
AS avgsize,
pg_catalog.max(dnsize)
AS maxsize,
pg_catalog.min(dnsize)
AS minsize,
(maxsize
- avgsize) * 100 AS skewsize
FROM
pg_catalog.gs_table_distribution()
GROUP
BY schemaname, tablename
)
SELECT
    schemaname AS schema_name,
    tablename AS table_name,
    totalsize AS total_size,
    avgsize::numeric(1000) AS avg_size,
    (
        CASE
            WHEN totalsize = 0 THEN 0.00
            ELSE (maxsize * 100 /
totalsize)::numeric(5, 2)
        END
    ) AS max_percent,
    (
        CASE
            WHEN totalsize = 0 THEN 0.00
            ELSE (minsize * 100 /
totalsize)::numeric(5, 2)
        END
    ) AS min_percent,
    (
        CASE
            WHEN totalsize = 0 THEN 0.00
            ELSE (skewsize /
maxsize)::numeric(5, 2)
        END
    ) AS skew_percent
FROM skew;
```


Consultar informações do banco de dados

- Consultar a lista de banco de dados usando o meta-comando **\l** da ferramenta **gsql**.

```
\l
                            List of databases
   Name    | Owner | Encoding  | Collate | Ctype | Access privileges
-----------+-------+-----------+---------+-------+-------------------
 gaussdb   | Ruby  | SQL_ASCII | C       | C     |
 template0 | Ruby  | SQL_ASCII | C       | C     | =c/Ruby           +
           |       |           |         |       | Ruby=CTc/Ruby
 template1 | Ruby  | SQL_ASCII | C       | C     | =c/Ruby           +
           |       |           |         |       | Ruby=CTc/Ruby
(3 rows)
```

  NOTA

  - Se os parâmetros **LC_COLLATE** e **LC_CTYPE** não forem especificados durante a instalação do banco de dados, os valores padrão deles serão **C**.
  - Se **LC_COLLATE** e **LC_CTYPE** não forem especificados durante a criação do banco de dados, a ordem de classificação e a classificação de caracteres do banco de dados de modelo serão usadas por padrão.

    Para obter detalhes, consulte **CREATE DATABASE**.

- Consultar a lista do banco de dados usando o catálogo do sistema **PG_DATABASE**

```
SELECT datname FROM pg_database;
  datname
-----------
 template1
 template0
 gaussdb
(3 rows)
```

Consultar o tamanho do banco de dados

Consultar o tamanho dos bancos de dados

```
select datname,pg_size_pretty(pg_database_size(datname)) from pg_database;
```

Exemplo:

```
select datname,pg_size_pretty(pg_database_size(datname)) from pg_database;
  datname  | pg_size_pretty
-----------+----------------
 template1 | 61 MB
 template0 | 61 MB
 postgres  | 320 MB
(3 rows)
```

Consultar o tamanho de uma tabela e o tamanho do índice correspondente em um esquema especificado

```
SELECT
    t.tablename,
    indexname,
    c.reltuples AS num_rows,
    pg_size_pretty(pg_relation_size(quote_ident(t.tablename)::text)) AS
table_size,
    pg_size_pretty(pg_relation_size(quote_ident(indexrelname)::text)) AS
index_size,
    CASE WHEN indisunique THEN 'Y'
       ELSE 'N'
    END AS UNIQUE,
    idx_scan AS number_of_scans,
    idx_tup_read AS tuples_read,
    idx_tup_fetch AS tuples_fetched
FROM pg_tables t
LEFT OUTER JOIN pg_class c ON t.tablename=c.relname
```



```
LEFT OUTER JOIN
    ( SELECT c.relname AS ctablename, ipg.relname AS indexname, x.indnatts AS
number_of_columns, idx_scan, idx_tup_read, idx_tup_fetch, indexrelname,
indisunique FROM pg_index x
           JOIN pg_class c ON c.oid = x.indrelid
           JOIN pg_class ipg ON ipg.oid = x.indexrelid
           JOIN pg_stat_all_indexes psai ON x.indexrelid = psai.indexrelid )
    AS foo
    ON t.tablename = foo.ctablename
WHERE t.schemaname='public'
ORDER BY 1,2;
```

5.7 Melhores práticas do banco de dados SEQUENCE

Uma sequência, também chamada de sequência, é um objeto de banco de dados usado para gerar um inteiro único. O valor de uma sequência aumenta ou diminui automaticamente com base em determinadas regras. Geralmente, uma sequência é usada como chave primária. No GaussDB(DWS), quando uma sequência é criada, uma tabela de metadados com o mesmo nome é criada para registrar informações de sequência. Por exemplo:

```
CREATE SEQUENCE seq_test;
CREATE SEQUENCE

SELECT * FROM seq_test;
 sequence_name | last_value | start_value | increment_by |      max_value      |
min_value | cache_value | log_cnt | is_cycled | is_called |  uuid
---------------+------------+-------------+--------------+---------------------
+-----------+-------------+---------+-----------+-----------+---------
 seq_test      |         -1 |           1 |            1 | 9223372036854775807
|         1 |           1 |       0 | f         | f         | 1400050
(1 row)
```

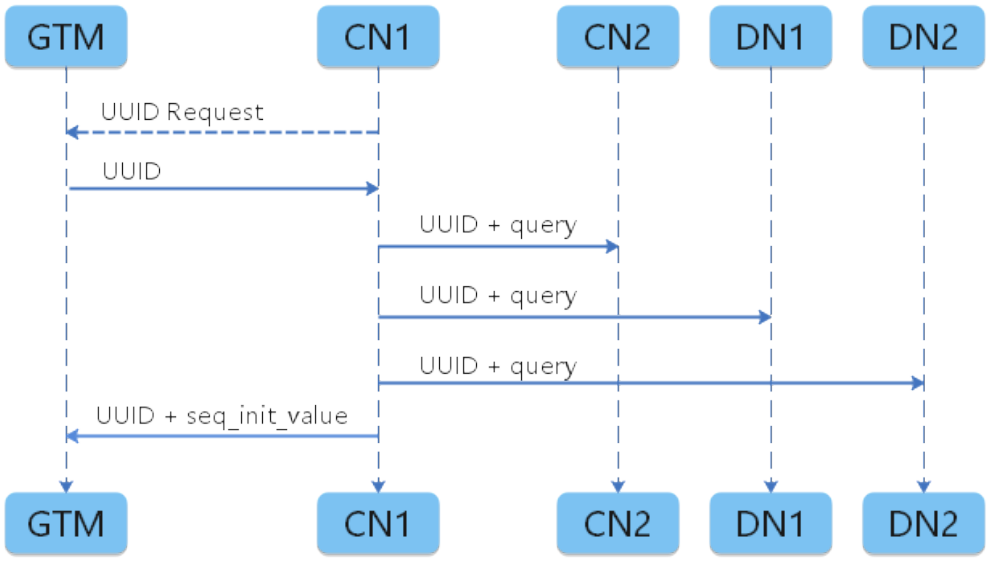
No comando anterior:

- sequence_name indica o nome de uma sequência.
- last_value não tem sentido.
- start_value indica o valor inicial da sequência.
- increment_by indica o passo da sequência.
- max_value indica o valor máximo de uma sequência.
- min_value indica o valor mínimo de sequência.
- cache_value indica o número de valores de sequência que são pré-armazenados para obter rapidamente o próximo valor de sequência. (Depois que o cache é definido, a continuidade dos valores de seqüência não pode ser assegurada, os furos são gerados e os segmentos de número de sequência são desperdiçados.)
- log_cnt indica o número de valores de sequência registrados nos logs WAL. No GaussDB(DWS), os valores de sequência são obtidos e gerenciados a partir do GTM. Portanto, log_cnt não tem sentido.
- is_cycled indica se deve continuar o loop após a sequência atingir o valor mínimo ou máximo.
- is_called indica se a sequência foi invocada. (Só indica se a sequência foi invocada na instância atual. Por exemplo, depois que a sequência é invocada em cn1, o valor da tabela de dados original em cn1 muda para t, e o valor do campo em cn2 ainda é f.)
- uuid indica o ID exclusivo da sequência.

Processo de criação de uma sequência

No GaussDB(DWS), o Global Transaction Manager (GTM) gera e mantém informações globalmente exclusivas, como IDs de transações globais, instantâneos de transações e sequências. A figura a seguir mostra o processo de criação de uma sequência no GaussDB(DWS).

Figura 5-1 Processo de criação de uma sequência



O processo específico é o seguinte:

1. O CN que aceita o comando SQL solicita um UUID do GTM.
2. O GTM retorna um UUID.
3. O CN vincula o UUID obtido ao sequenceName criado pelo usuário.
4. O CN fornece a relação de vinculação para outros nós, e outros nós criam a tabela de metadados de sequência de forma síncrona.
5. O CN envia o UUID e startID da sequência para o GTM para armazenamento permanente.

Portanto, a manutenção da sequência e a solicitação são realmente concluídas no GTM. Ao solicitar nextval, cada instância que invoca nextval solicita um valor de sequência do GTM com base no UUID da sequência. O intervalo de valores de sequência solicitado para cada vez está relacionado ao cache. A instância solicita um valor de sequência do GTM somente depois que o cache é usado. Portanto, aumentar o cache da sequência ajuda a reduzir o número de vezes que o CN/DN se comunica com o GTM.

Dois métodos de criar uma sequência

Método 1: execute a instrução CREATE SEQUENCE para criar uma sequência e use nextval para chamar a sequência na nova tabela.

```
CREATE SEQUENCE seq_test increment by 1 minvalue 1 no maxvalue start with 1;
CREATE SEQUENCE

CREATE TABLE table_1(id int not null default nextval('seq_test'), name text);
CREATE TABLE
```


Método 2: se o tipo serial for usado durante a criação da tabela, uma sequência será criada automaticamente e o valor padrão da coluna será definido como nextval.

```
CREATE TABLE mytable(a int, b serial) distribute by hash(a);
NOTICE:  CREATE TABLE will create implicit sequence "mytable_b_seq" for serial
column "mytable.b"
CREATE TABLE

 \d+ mytable
                                          Table "dbadmin.mytable"
 Column |  Type   |                      Modifiers                       | Storage
| Stats target | Description
--------+---------+------------------------------------------------------+---------
+--------------+-------------
 a      | integer |                                                      | plain
|              |
 b      | integer | not null default nextval('mytable_b_seq'::regclass) | plain
|              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no
```

Neste exemplo, uma sequência chamada mytable_b_seq é criada automaticamente. A rigor, o tipo serial não é um tipo real. É apenas um conceito para definir um identificador exclusivo em uma tabela. Quando um tipo serial é criado, uma sequência é criada e associada à coluna.

É equivalente à seguinte afirmação:

```
CREATE TABLE mytable01(a int, b int) distribute by hash(a);
CREATE TABLE

CREATE SEQUENCE mytable01_b_seq owned by mytable.b;
CREATE SEQUENCE

ALTER SEQUENCE mytable01_b_seq owner to u1; --u1 is the owner of the mytable01
table. If the current user is the owner, you do not need to run this statement.
ALTER SEQUENCE

ALTER TABLE mytable01 alter b set default nextval('mytable01_b_seq'), alter b set
not null;
ALTER TABLE

\d+ mytable01
                                          Table "dbadmin.mytable01"
 Column |  Type   |                       Modifiers                        |
Storage | Stats target | Description
--------+---------+--------------------------------------------------------
+---------+--------------+-------------
 a      | integer |                                                        |
plain   |              |
 b      | integer | not null default nextval('mytable01_b_seq'::regclass) |
plain   |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no
```

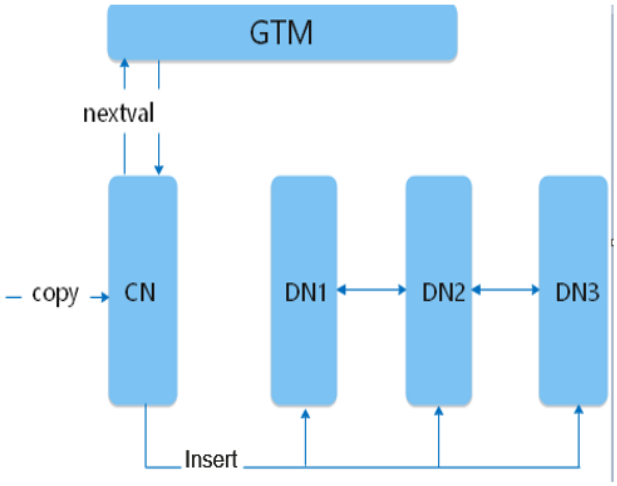
Uso comum de sequências em serviços

Sequências são frequentemente usadas para gerar chaves primárias ou colunas exclusivas durante a importação de dados em cenários de migração de dados. Diferentes ferramentas de migração ou cenários de importação de serviços usam diferentes métodos de importação. Os métodos comuns de importação são classificados em **copy** e **insert**. Para sequência, o processamento nos dois cenários é ligeiramente diferente.

- **Cenário 1:inserir pushdown**

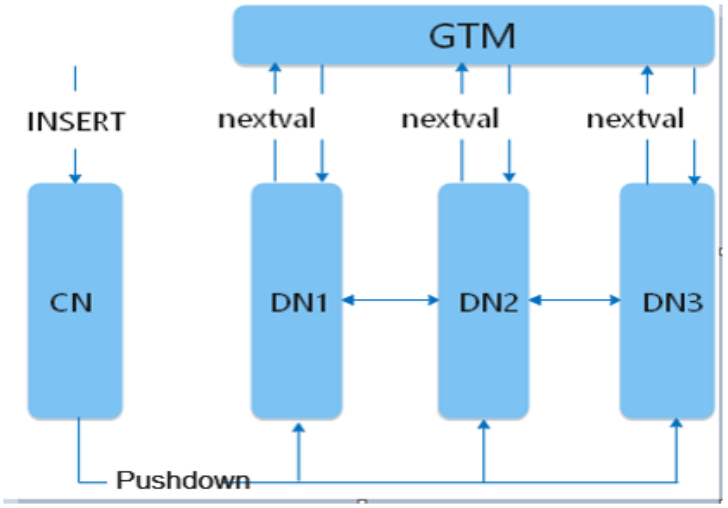
```
CREATE TABLE test1(a int, b serial) distribute by hash(a);
NOTICE:  CREATE TABLE will create implicit sequence "test1_b_seq" for serial
column "test1.b"
CREATE TABLE

CREATE TABLE test2(a int) distribute by hash(a);
CREATE TABLE

EXPLAIN VERBOSE INSERT INTO test1(a) SELECT a FROM test2;
                                          QUERY PLAN
------------------------------------------------------------------------------
------------------
  id |              operation              | E-rows | E-distinct | E-memory |
E-width | E-costs
 ----+-------------------------------------+--------+------------+----------
+---------+---------
   1 | ->  Streaming (type: GATHER)        |      1 |            |
|       4 | 16.34
   2 |    ->  Insert on dbadmin.test1      |     30 |            |
|       4 | 16.22
   3 |       ->  Seq Scan on dbadmin.test2 |     30 |            | 1MB
|       4 | 14.21

             RunTime Analyze Information
 ---------------------------------------------------
         "dbadmin.test2" runtime: 9.586ms, sync stats

      Targetlist Information (identified by plan id)
 ---------------------------------------------------------
   1 --Streaming (type: GATHER)
         Node/s: All datanodes
   3 --Seq Scan on dbadmin.test2
         Output: test2.a, nextval('test1_b_seq'::regclass)
         Distribute Key: test2.a

   ====== Query Summary =====
 -------------------------------
 System available mem: 1351680KB
 Query Max mem: 1351680KB
 Query estimated mem: 1024KB
 Parser runtime: 0.076 ms
 Planner runtime: 12.666 ms
 Unique SQL Id: 831364267
(26 rows)
```

No cenário INSERT, nextval pode ser empurrado para baixo para DNs para execução. Portanto, nextval é enviado para DNs para execução, independentemente de nextval com o valor padrão ser usado ou nextval ser chamado explicitamente. O plano de execução no exemplo anterior também mostra que nextval é empurrado para baixo para DNs para execução, a invocação de nextval está na camada de sequência, indicando que nextval é executado em DNs. Neste caso, os DNs solicitam diretamente valores de sequência do GTM, e os DNs executam a solicitação simultaneamente. Portanto, a eficiência é relativamente alta.



- **Cenário 2: copiar cenário**

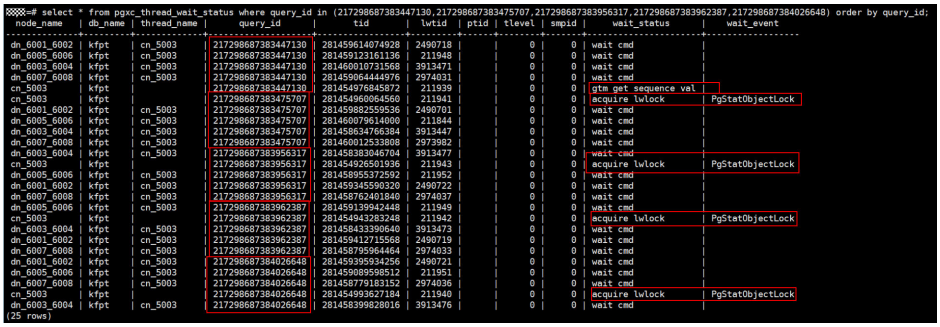
  Durante o desenvolvimento do serviço, além de INSERT, COPY pode ser usado para importar dados para o banco de dados. Esse método é usado para copiar o conteúdo do arquivo para o banco de dados ou usar a interface do CopyManager para importar o conteúdo do arquivo para o banco de dados. Além disso, a ferramenta de sincronização de dados do CDM importa dados para o banco de dados em lotes, copiando dados. Se a tabela de destino a ser copiada usar o valor padrão de nextval, o processo será o seguinte:



  No cenário de cópia, o CN solicita valores de sequência do GTM. Portanto, quando o valor de cache da sequência é pequeno, o CN frequentemente estabelece conexões com o GTM e solicita para nextval, causando um gargalo de desempenho. **Cenários típicos de otimização relacionados a sequências** descreve o desempenho do serviço nesse cenário e fornece métodos de otimização.

Cenários típicos de otimização relacionados a sequências

Cenário de serviço: em um cenário de serviço, a ferramenta de sincronização de dados do CDM é usada para migrar dados e importar dados da extremidade de origem para o GaussDB(DWS) de destino. A taxa de importação difere muito do valor empírico. Depois que a simultaneidade do CDM é alterada de 1 para 5, a taxa de sincronização ainda não pode ser melhorada. Verifique o status de execução da instrução. Exceto COPY, outros serviços são executados corretamente, sem gargalos de desempenho ou gargalos de recursos. Por isso, apura-se preliminarmente que o serviço tem um gargalo. Verifique a exibição de trabalho em espera relacionada a COPY.



```
=# select * from pgxc_thread_wait_status where query_id in (2172986873834471 30,2172986873834757 07,2172986873839563 17,2172986873839623 87,2172986873840266 48) order by query_id;
  node_name   | db_name | thread_name |      query_id       |       tid       |  lwtid  | ptid | tlevel | smpid |     wait_status      |     wait_event
--------------+---------+-------------+---------------------+-----------------+---------+------+--------+-------+----------------------+---------------------
 dn_6001_6002 | kfpt    | cn_5003     | 2172986873834471 30 | 281459614074928 | 2490718 |      |      0 |     0 | wait cmd             |
 dn_6005_6006 | kfpt    | cn_5003     | 2172986873834471 30 | 281459123161136 |  211948 |      |      0 |     0 | wait cmd             |
 dn_6003_6004 | kfpt    | cn_5003     | 2172986873834471 30 | 281460010731568 | 3913471 |      |      0 |     0 | wait cmd             |
 dn_6007_6008 | kfpt    | cn_5003     | 2172986873834471 30 | 281459064444976 | 2974031 |      |      0 |     0 | wait cmd             |
 cn_5003      | kfpt    |             | 2172986873834471 30 | 281454976845872 |  211939 |      |      0 |     0 | gtm get sequence val |
 cn_5003      | kfpt    |             | 2172986873834757 07 | 281454966064560 |  211941 |      |      0 |     0 | acquire lwlock       | PgStatObjectLock
 dn_6001_6002 | kfpt    | cn_5003     | 2172986873834757 07 | 281459882559536 | 2490701 |      |      0 |     0 | wait cmd             |
 dn_6005_6006 | kfpt    | cn_5003     | 2172986873834757 07 | 281460079614000 |  211844 |      |      0 |     0 | wait cmd             |
 dn_6003_6004 | kfpt    | cn_5003     | 2172986873834757 07 | 281458634766384 | 3913447 |      |      0 |     0 | wait cmd             |
 dn_6007_6008 | kfpt    | cn_5003     | 2172986873834757 07 | 281460012533808 | 2973982 |      |      0 |     0 | wait cmd             |
 dn_6003_6004 | kfpt    | cn_5003     | 2172986873839563 17 | 281458383046704 | 3913477 |      |      0 |     0 | wait cmd             |
 cn_5003      | kfpt    |             | 2172986873839563 17 | 281454926501936 |  211943 |      |      0 |     0 | acquire lwlock       | PgStatObjectLock
 dn_6005_6006 | kfpt    | cn_5003     | 2172986873839563 17 | 281458955372592 |  211952 |      |      0 |     0 | wait cmd             |
 dn_6001_6002 | kfpt    | cn_5003     | 2172986873839563 17 | 281459345590320 | 2490722 |      |      0 |     0 | wait cmd             |
 dn_6007_6008 | kfpt    | cn_5003     | 2172986873839563 17 | 281458762401840 | 2974037 |      |      0 |     0 | wait cmd             |
 dn_6005_6006 | kfpt    | cn_5003     | 2172986873839623 87 | 281459139942448 |  211949 |      |      0 |     0 | wait cmd             |
 cn_5003      | kfpt    |             | 2172986873839623 87 | 281454943283248 |  211942 |      |      0 |     0 | acquire lwlock       | PgStatObjectLock
 dn_6003_6004 | kfpt    | cn_5003     | 2172986873839623 87 | 281458433390640 | 3913473 |      |      0 |     0 | wait cmd             |
 dn_6001_6002 | kfpt    | cn_5003     | 2172986873839623 87 | 281459412715568 | 2490719 |      |      0 |     0 | wait cmd             |
 dn_6007_6008 | kfpt    | cn_5003     | 2172986873839623 87 | 281458795964464 | 2974033 |      |      0 |     0 | wait cmd             |
 dn_6001_6002 | kfpt    | cn_5003     | 2172986873840266 48 | 281459395934256 | 2490721 |      |      0 |     0 | wait cmd             |
 dn_6005_6006 | kfpt    | cn_5003     | 2172986873840266 48 | 281459089598512 |  211951 |      |      0 |     0 | wait cmd             |
 dn_6007_6008 | kfpt    | cn_5003     | 2172986873840266 48 | 281458779183152 | 2974036 |      |      0 |     0 | wait cmd             |
 cn_5003      | kfpt    |             | 2172986873840266 48 | 281454993627184 |  211940 |      |      0 |     0 | acquire lwlock       | PgStatObjectLock
 dn_6003_6004 | kfpt    | cn_5003     | 2172986873840266 48 | 281458399828016 | 3913476 |      |      0 |     0 | wait cmd             |
(25 rows)
```

Como mostrado na figura anterior, cinco trabalhos do CDM são executados simultaneamente. Portanto, você pode ver cinco instruções COPY na exibição ativa. Verifique a exibição em espera com base em query_id correspondente às cinco instruções COPY. Entre as cinco cópias, apenas uma cópia está solicitando um valor de sequência do GTM ao mesmo tempo, e outras cópias estão esperando por um bloqueio leve. Portanto, mesmo que cinco trabalhos simultâneos estejam ativados, o efeito real não é significativamente melhorado em comparação com o de um trabalho concorrente.

Motivo: o tipo serial é usado quando a tabela de destino é criada. Por padrão, o cache da sequência criada é 1. Como resultado, quando os dados são copiados simultaneamente para o banco de dados, o CN frequentemente estabelece conexões com o GTM, e a contenção de bloqueio leve existe entre várias tarefas simultâneas, resultando em baixa eficiência de sincronização de dados.

Solução: nesse cenário, aumente o valor de cache da sequência para evitar gargalos causados pelo estabelecimento de conexão do GTM frequente. Neste exemplo de cenário de serviço, cerca de 100.000 registros de dados são sincronizados cada vez. Com base na avaliação do serviço, altere o valor do cache para 10.000. (Na prática, defina um valor de cache adequado com base em serviços para garantir o acesso rápido e evitar o desperdício de números sequenciais.)

Nas versões de cluster 8.2.1.100 e posteriores, você pode usar ALTER SEQUENCE para alterar o valor do cache.

Em clusters de 8.2.1 e versões anteriores, o valor de cache de GaussDB(DWS) não pode ser alterado usando ALTER SEQUENCE. Você pode alterar o valor de cache de uma sequência existente da seguinte forma (a tabela mytable é usada como exemplo):

Passo 1 Remova a associação entre a sequência atual e a tabela de destino.

```
ALTER SEQUENCE mytable_b_seq owned by none;
ALTER TABLE mytable alter b drop default;
```

Passo 2 Registre o número de sequência atual como o valor inicial da nova sequência.

```
SELECT nextval('mytable_b_seq');
```


Elimine uma sequência.

```
DROP SEQUENCE mytable_b_seq;
```

Passo 3 Crie sequência e vincule-a à tabela de destino. Substitua xxx pelo valor de nextval obtido na etapa anterior.

```
CREATE SEQUENCE mytable_b_seq START with xxx cache 10000 owned by mytable.b;
ALTER SEQUENCE mytable_b_seq owner to u1;--u1 is the owner of the mytable table.
If the current user is the owner, you do not need to run this statement.
ALTER TABLE mytable alter b set default nextval('mytable_b_seq');
```

----Fim

6 Análise de dados de amostra

6.1 Análise de veículos no ponto de verificação

Esta prática mostra como analisar os veículos que passam nos postos de verificação. Nessa prática, 890 milhões de registros de dados de pontos de verificação são carregados em uma única tabela de banco de dados no GaussDB(DWS) para consultas precisas e difusas, demonstrando a capacidade do GaussDB(DWS) de executar consultas de alto desempenho para dados históricos.

NOTA

Os dados de amostra foram carregados na pasta **traffic-data** em um bucket do OBS, e todas as contas da Huawei Cloud receberam a permissão somente leitura para acessar o bucket do OBS.

Procedimento geral

Essa prática leva cerca de 40 minutos. O processo básico é o seguinte:

1. **Fazer preparações**
2. **Passo 1: criar um cluster**
3. **Passo 2: usar o Data Studio para conectar-se a um cluster**
4. **Passo 3: importar dados de amostra**
5. **Passo 4: realizar análise de veículos**

Regiões suportadas

Tabela 6-1 descreve as regiões onde os dados do OBS foram carregados.

Tabela 6-1 Regiões e nomes de bucket do OBS

Região	Bucket de OBS
CN North-Beijing1	dws-demo-cn-north-1
CN North-Beijing2	dws-demo-cn-north-2

Região	Bucket de OBS
CN North-Beijing4	dws-demo-cn-north-4
CN North-Ulanqab1	dws-demo-cn-north-9
CN East-Shanghai1	dws-demo-cn-east-3
CN East-Shanghai2	dws-demo-cn-east-2
CN South-Guangzhou	dws-demo-cn-south-1
CN South-Guangzhou-InvitationOnly	dws-demo-cn-south-4
CN-Hong Kong	dws-demo-ap-southeast-1
AP-Singapore	dws-demo-ap-southeast-3
AP-Bangkok	dws-demo-ap-southeast-2
LA-Santiago	dws-demo-la-south-2
AF-Johannesburg	dws-demo-af-south-1
LA-Mexico City1	dws-demo-na-mexico-1
LA-Mexico City2	dws-demo-la-north-2
RU-Moscow2	dws-demo-ru-northwest-2
LA-Sao Paulo1	dws-demo-sa-brazil-1

Fazer preparações

- Você registrou uma conta do GaussDB(DWS) e verificou o status da conta antes de usar GaussDB(DWS). A conta não pode estar em atraso ou congelada.
- Você obteve o AK e SK da conta.

Passo 1: criar um cluster

Passo 1 Faça logon no console de gerenciamento.

Passo 2 Clique em **Service List** e escolha **Analytics** > **GaussDB(DWS)**.

Passo 3 No painel de navegação à esquerda, escolha **Clusters**. Na página exibida, clique em **Create Cluster** no canto superior direito.

Passo 4 Configure parâmetros de acordo com **Tabela 6-2**.

Tabela 6-2 Configuração básica

Parâmetro	Configuração
Region	Selecione **CN North-Beijing4** or **CN-Hong KongEU-Dublin**. **NOTA** **CN-Hong Kong** é usado como exemplo. Você pode selecionar outras regiões, conforme necessário. Certifique-se de que todas as operações sejam realizadas na mesma região.
AZ	AZ2
Resource	Armazém padrão
Compute Resource	ECS
Storage type	Cloud SSD
CPU Architecture	X86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD) **NOTA** Se esse flavor estiver esgotado, selecione outras AZs ou flavors.
Hot Storage	100 GB/node
Nodes	3

Passo 5 Verifique se as informações estão corretas e clique em **Next: Configure Network**. Configure a rede fazendo referência a **Tabela 6-3**.

Tabela 6-3 Configuração da rede

Parâmetro	Configuração
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24)
Security Group	Automatic creation
EIP	Buy now
Bandwidth	1 Mbit/s
ELB	Não use

Passo 6 Verifique se as informações estão corretas e clique em **Next: Configure Advanced Settings**. Configure a rede fazendo referência a **Tabela 6-4**.

Tabela 6-4 Configuração de definições avançadas

Parâmetro	Configuração
Cluster Name	dws-demo
Cluster Version	Use a versão recomendada, por exemplo, 8.1.3.311.
Administrator Account	dbadmin
Administrator Password	-
Confirm Password	-
Database Port	8000
Enterprise Project	default
Advanced Settings	Default

Passo 7 Clique em **Next: Confirm**, confirme a configuração e clique em **Next**.

Passo 8 Espere cerca de 6 minutos. Depois que o cluster for criado, clique em ao lado do nome do cluster. Na página de informações do cluster exibida, registre o valor de **Public Network Address**, por exemplo, **dws-demov.dws.huaweicloud.com**.

----Fim

Passo 2: usar o Data Studio para conectar-se a um cluster

Passo 1 Certifique-se de que o JDK 1.8.0 ou posterior tenha sido instalado no host do cliente. Escolha **PC > Properties > Advanced System Settings > Environment Variables** e defina **JAVA_HOME** (por exemplo, **C:\Program Files\Java\jdk1.8.0_191**).
Adicione **;%JAVA_HOME%\bin** à variável **path**.

Passo 2 Na página **Connections** do console GaussDB(DWS), baixe o cliente de GUI do Data Studio.

Passo 3 Descompacte o pacote de software do Data Studio baixado, vá para o diretório descompactado e clique duas vezes em **Data Studio.exe** para iniciar o cliente.

Passo 4 No menu principal do Data Studio, escolha **File > New Connection**. Na caixa de diálogo exibida, configure a conexão com base em **Tabela 6-5**.

Tabela 6-5 Configuração do software Data Studio

Parâmetro	Configuração
Database Type	GaussDB(DWS)
Connection Name	dws-demo
Host	dws-demov.dws.huaweicloud.com O valor deste parâmetro deve ser o mesmo que o valor de **Public Network Address** consultado em **Passo 1: criar um cluster**.
Host Port	8000
Database Name	gaussdb
User Name	dbadmin
Password	-
Enable SSL	Disable

Passo 5 Clique em **OK**.

----Fim

Passo 3: importar dados de amostra

Depois de se conectar ao cluster usando a ferramenta de cliente SQL, execute as seguintes operações na ferramenta de cliente SQL para importar os dados de exemplo de pontos de verificação de tráfego e executar consultas de dados.

Passo 1 Execute a instrução a seguir para criar o banco de dados **traffic**:

```
CREATE DATABASE traffic encoding 'utf8' template template0;
```

Passo 2 Execute as seguintes etapas para alternar para o novo banco de dados:

1. Na janela **Object Browser** do cliente do Data Studio, clique com o botão direito do mouse na conexão de banco de dados e selecione **Refresh** no menu de atalho. Em seguida, o novo banco de dados é exibido.
2. Clique com o botão direito do mouse no nome do novo banco de dados **traffic** e escolha **Connect to DB** no menu de atalho.
3. Clique com o botão direito do mouse no nome do novo banco de dados **traffic** e escolha **Open Terminal** no menu de atalho. A janela de comando SQL para conexão com o banco de dados especificado é exibida. Execute os seguintes passos na janela.

Passo 3 Execute as seguintes instruções para criar uma tabela de banco de dados para armazenar informações de veículos de pontos de verificação de tráfego:


```
CREATE SCHEMA traffic_data;
SET current_schema= traffic_data;
DROP TABLE if exists GCJL;
CREATE TABLE GCJL
(
        kkbh   VARCHAR(20),
        hphm   VARCHAR(20),
        gcsj   DATE ,
        cplx   VARCHAR(8),
        cllx   VARCHAR(8),
        csys   VARCHAR(8)
)
with (orientation = column, COMPRESSION=MIDDLE)
distribute by hash(hphm);
```

Passo 4 Crie uma tabela estrangeira, que é usada para identificar e associar os dados de origem no OBS.

AVISO

- *<obs_bucket_name>* indica o nome do bucket do OBS. Apenas algumas regiões são suportadas. Para obter detalhes sobre as regiões suportadas e os nomes dos bucket do OBS, consulte **Regiões suportadas**. Os clusters do GaussDB(DWS) não oferecem suporte ao acesso entre regiões aos dados do bucket do OBS.
- Nesta prática, a região **CN-Hong Kong** é usada como exemplo. Digite **dws-demo-ap-southeast-1** e substitua *<Access_Key_Id>* e *<Secret_Access_Key>* pelo valor obtido em **Fazer preparações**.
- // AK e SK codificados rigidamente ou em texto não criptografado são arriscados. Para fins de segurança, criptografe seu AK e SK e armazene-os no arquivo de configuração ou nas variáveis de ambiente.
- Se a mensagem "ERROR: schema "*xxx*" does not exist Position" for exibida quando você criar uma tabela estrangeira, o esquema não existe. Execute a etapa anterior para criar um esquema.

```
CREATE SCHEMA tpchobs;
SET current_schema = 'tpchobs';
DROP FOREIGN table if exists GCJL_OBS;
CREATE FOREIGN TABLE GCJL_OBS
(
        like traffic_data.GCJL
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/traffic-data/gcxx',
        format 'text',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
```

Passo 5 Execute a instrução a seguir para importar dados da tabela estrangeira para a tabela do banco de dados:

```
INSERT INTO traffic_data.GCJL SELECT * FROM tpchobs.GCJL_OBS;
```

Leva algum tempo para importar dados.

----Fim

Passo 4: realizar análise de veículos

1. **Execução de ANALYZE**

   Esta instrução coleta estatísticas relacionadas a tabelas ordinárias em bancos de dados. As estatísticas são salvas no catálogo do sistema **PG_STATISTIC**. Quando você executa o planejador, as estatísticas o ajudam a desenvolver um plano de execução de consulta eficiente.

   Execute a instrução a seguir para gerar as estatísticas da tabela:

   ```
   ANALYZE;
   ```

2. **Consulta do volume de dados da tabela de dados**

   Execute a instrução a seguir para consultar o número de registros de dados carregados:

   ```
   SET current_schema= traffic_data;
   SELECT count(*) FROM traffic_data.gcjl;
   ```

3. **Consulta precisa do veículo**

   Execute as seguintes instruções para consultar a rota de condução de um veículo pelo número da placa de licença e segmento de tempo. GaussDB(DWS) responde à solicitação em segundos.

   ```
   SET current_schema= traffic_data;
   SELECT hphm, kkbh, gcsj
   FROM traffic_data.gcjl
   where hphm =  'YD38641'
   and gcsj between '2016-01-06' and '2016-01-07'
   order by gcsj desc;
   ```

4. **Consulta difusa do veículo**

   Execute as seguintes instruções para consultar a rota de condução de um veículo pelo número da placa de licença e segmento de tempo. GaussDB(DWS) responde à solicitação em segundos.

   ```
   SET current_schema= traffic_data;
   SELECT hphm, kkbh, gcsj
   FROM traffic_data.gcjl
   where hphm like  'YA23F%'
   and kkbh in('508', '1125', '2120')
   and gcsj between '2016-01-01' and '2016-01-07'
   order by hphm,gcsj desc;
   ```

6.2 Análise de requisitos da cadeia de suprimentos de uma empresa

Esta prática descreve como carregar o conjunto de dados de amostra do OBS para um cluster de armazém de dados e executar consultas de dados. Este exemplo compreende análise de várias tabelas e análise de temas no cenário de análise de dados.

> **NOTA**
>
> Neste exemplo, um conjunto de dados TPC-H-1x padrão de 1 GB de tamanho foi gerado no GaussDB(DWS) e foi carregado na pasta **tpch** de um bucket do OBS. Todas as contas da HUAWEI CLOUD receberam a permissão somente leitura para acessar o bucket do OBS. Os usuários podem importar facilmente o conjunto de dados usando suas contas.

Procedimento geral

Essa prática leva cerca de 60 minutos. O procedimento é os seguintes:

1. **Fazer preparações**
2. **Passo 1: importar dados de amostra**
3. **Passo 2: realizar análise de de várias tabelas e análise de temas**

Regiões suportadas

Tabela 6-6 descreve as regiões onde os dados do OBS foram carregados.

Tabela 6-6 Regiões e nomes de bucket do OBS

Região	Bucket de OBS
CN North-Beijing1	dws-demo-cn-north-1
CN North-Beijing2	dws-demo-cn-north-2
CN North-Beijing4	dws-demo-cn-north-4
CN North-Ulanqab1	dws-demo-cn-north-9
CN East-Shanghai1	dws-demo-cn-east-3
CN East-Shanghai2	dws-demo-cn-east-2
CN South-Guangzhou	dws-demo-cn-south-1
CN South-Guangzhou-InvitationOnly	dws-demo-cn-south-4
CN-Hong Kong	dws-demo-ap-southeast-1
AP-Singapore	dws-demo-ap-southeast-3
AP-Bangkok	dws-demo-ap-southeast-2
LA-Santiago	dws-demo-la-south-2
AF-Johannesburg	dws-demo-af-south-1
LA-Mexico City1	dws-demo-na-mexico-1
LA-Mexico City2	dws-demo-la-north-2
RU-Moscow2	dws-demo-ru-northwest-2
LA-Sao Paulo1	dws-demo-sa-brazil-1

Descrição do cenário

Compreenda as funções básicas do GaussDB(DWS) e como importar dados. Analise os dados de pedidos de uma empresa e seus fornecedores da seguinte forma:

1. Analise a receita trazida pelos fornecedores de uma região para a empresa. As estatísticas podem ser usadas para determinar se um centro de alocação local precisa ser estabelecido em uma determinada região.

2. Analise a relação entre peças e fornecedores para obter o número de fornecedores de peças com base nas condições de contribuição especificadas. As informações podem ser usadas para determinar se os fornecedores são suficientes para grandes quantidades de pedidos quando a tarefa é urgente.
3. Analise a perda de receita de pedidos pequenos. Você pode consultar a perda média de receita anual se não houver pedidos pequenos. Filtre pedidos pequenos inferiores a 20% do volume médio de fornecimento e calcule o valor total desses pedidos pequenos para descobrir a perda média de receita anual.

Fazer preparações

- Você registrou uma conta do GaussDB(DWS) e verificou o status da conta antes de usar GaussDB(DWS). A conta não pode estar em atraso ou congelada.
- Você obteve o AK e SK da conta.
- Um cluster foi criado e conectado usando o Data Studio. Para mais detalhes, consulte **Análise de veículos no ponto de verificação**.

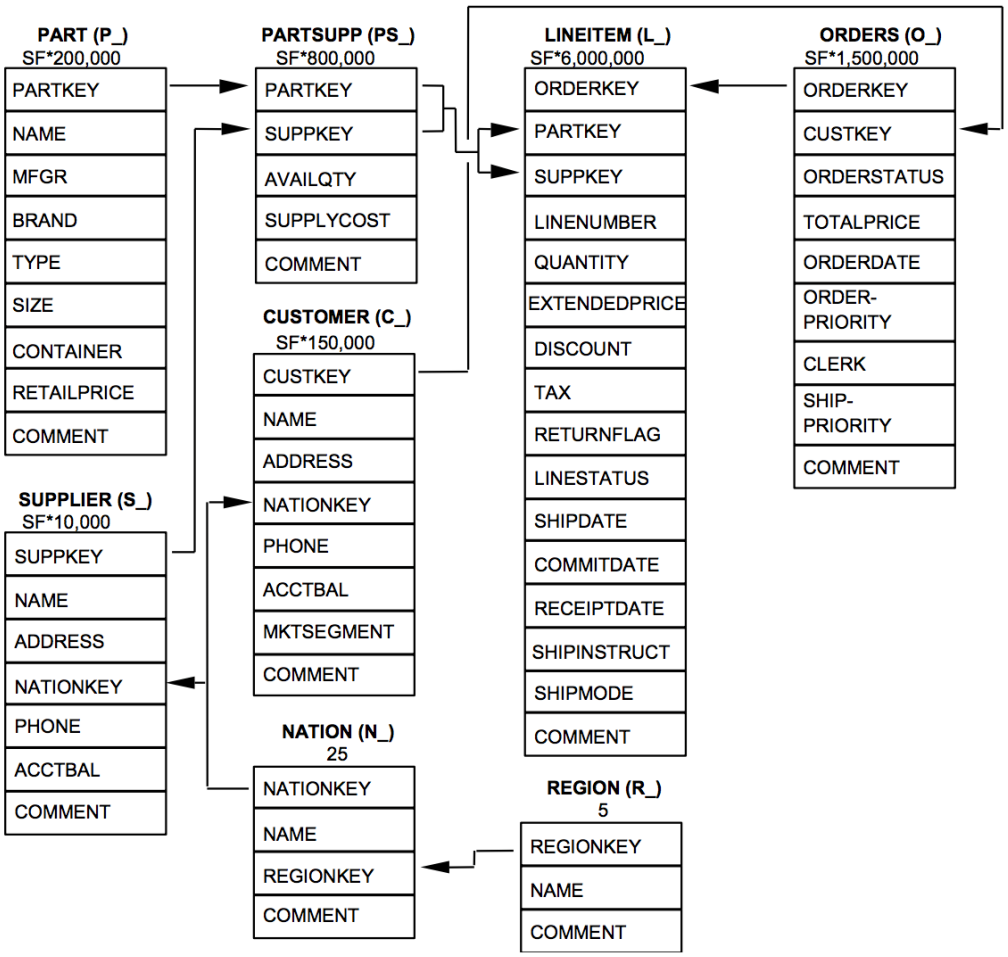
Passo 1: importar dados de amostra

Depois de se conectar ao cluster usando a ferramenta de cliente SQL, execute as seguintes operações na ferramenta de cliente SQL para importar os dados de amostra TPC-H e executar consultas de dados.

Passo 1 Crie uma tabela de banco de dados.

Os dados de amostra do TPC-H consistem em oito tabelas de banco de dados cujas associações são mostradas em **Figura 6-1**.

Figura 6-1 Tabelas de dados TPC-H



Execute as seguintes instruções para criar tabelas no banco de dados **gaussdb**.

```
CREATE SCHEMA tpch;
SET current_schema = tpch;

DROP TABLE if exists region;
CREATE TABLE REGION
(
        R_REGIONKEY  INT NOT NULL ,
        R_NAME       CHAR(25) NOT NULL ,
        R_COMMENT    VARCHAR(152)
)
with (orientation = column, COMPRESSION=MIDDLE)
distribute by replication;

DROP TABLE if exists nation;
CREATE TABLE NATION
(
        N_NATIONKEY  INT NOT NULL,
        N_NAME       CHAR(25) NOT NULL,
        N_REGIONKEY  INT NOT NULL,
        N_COMMENT    VARCHAR(152)
)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by replication;

DROP TABLE if exists supplier;
CREATE TABLE SUPPLIER
```



```
(
        S_SUPPKEY     BIGINT NOT NULL,
        S_NAME        CHAR(25) NOT NULL,
        S_ADDRESS     VARCHAR(40) NOT NULL,
        S_NATIONKEY   INT NOT NULL,
        S_PHONE       CHAR(15) NOT NULL,
        S_ACCTBAL     DECIMAL(15,2) NOT NULL,
        S_COMMENT     VARCHAR(101) NOT NULL
)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by hash(S_SUPPKEY);

DROP TABLE if exists customer;
CREATE TABLE CUSTOMER
(
        C_CUSTKEY     BIGINT NOT NULL,
        C_NAME        VARCHAR(25) NOT NULL,
        C_ADDRESS     VARCHAR(40) NOT NULL,
        C_NATIONKEY   INT NOT NULL,
        C_PHONE       CHAR(15) NOT NULL,
        C_ACCTBAL     DECIMAL(15,2)   NOT NULL,
        C_MKTSEGMENT  CHAR(10) NOT NULL,
        C_COMMENT     VARCHAR(117) NOT NULL
)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by hash(C_CUSTKEY);

DROP TABLE if exists part;
CREATE TABLE PART
(
        P_PARTKEY     BIGINT NOT NULL,
        P_NAME        VARCHAR(55) NOT NULL,
        P_MFGR        CHAR(25) NOT NULL,
        P_BRAND       CHAR(10) NOT NULL,
        P_TYPE        VARCHAR(25) NOT NULL,
        P_SIZE        BIGINT NOT NULL,
        P_CONTAINER   CHAR(10) NOT NULL,
        P_RETAILPRICE DECIMAL(15,2) NOT NULL,
        P_COMMENT     VARCHAR(23) NOT NULL
)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by hash(P_PARTKEY);

DROP TABLE if exists partsupp;
CREATE TABLE PARTSUPP
(
        PS_PARTKEY     BIGINT NOT NULL,
        PS_SUPPKEY     BIGINT NOT NULL,
        PS_AVAILQTY    BIGINT NOT NULL,
        PS_SUPPLYCOST  DECIMAL(15,2)  NOT NULL,
        PS_COMMENT     VARCHAR(199) NOT NULL
)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by hash(PS_PARTKEY);

DROP TABLE if exists orders;
CREATE TABLE ORDERS
(
        O_ORDERKEY       BIGINT NOT NULL,
        O_CUSTKEY        BIGINT NOT NULL,
        O_ORDERSTATUS    CHAR(1) NOT NULL,
        O_TOTALPRICE     DECIMAL(15,2) NOT NULL,
        O_ORDERDATE      DATE NOT NULL ,
        O_ORDERPRIORITY  CHAR(15) NOT NULL,
        O_CLERK          CHAR(15) NOT NULL ,
        O_SHIPPRIORITY   BIGINT NOT NULL,
        O_COMMENT        VARCHAR(79) NOT NULL
)
with (orientation = column,COMPRESSION=MIDDLE)
```



```
distribute by hash(O_ORDERKEY);

DROP TABLE if exists lineitem;
CREATE TABLE LINEITEM
(
        L_ORDERKEY    BIGINT NOT NULL,
        L_PARTKEY     BIGINT NOT NULL,
        L_SUPPKEY     BIGINT NOT NULL,
        L_LINENUMBER  BIGINT NOT NULL,
        L_QUANTITY    DECIMAL(15,2) NOT NULL,
        L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL,
        L_DISCOUNT    DECIMAL(15,2) NOT NULL,
        L_TAX         DECIMAL(15,2) NOT NULL,
        L_RETURNFLAG  CHAR(1) NOT NULL,
        L_LINESTATUS  CHAR(1) NOT NULL,
        L_SHIPDATE    DATE NOT NULL,
        L_COMMITDATE  DATE NOT NULL ,
        L_RECEIPTDATE DATE NOT NULL,
        L_SHIPINSTRUCT CHAR(25) NOT NULL,
        L_SHIPMODE     CHAR(10) NOT NULL,
        L_COMMENT      VARCHAR(44) NOT NULL
)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by hash(L_ORDERKEY);
```

Passo 2 Crie uma tabela estrangeira, que é usada para identificar e associar os dados de origem no OBS.

AVISO

- *<obs_bucket_name>* indica o nome do bucket do OBS. Apenas algumas regiões são suportadas. Para obter detalhes sobre as regiões suportadas e os nomes dos bucket do OBS, consulte **Regiões suportadas**. Os clusters do GaussDB(DWS) não oferecem suporte ao acesso entre regiões aos dados do bucket do OBS.
- Nesta prática, a região **CN-Hong Kong** é usada como exemplo. Digite **dws-demo-ap-southeast-1** e substitua *<Access_Key_Id>* e *<Secret_Access_Key>* pelo valor obtido em **Fazer preparações**.
- // AK e SK codificados rigidamente ou em texto não criptografado são arriscados. Para fins de segurança, criptografe seu AK e SK e armazene-os no arquivo de configuração ou nas variáveis de ambiente.
- Se a mensagem"ERROR: schema "*xxx*" does not exist Position" for exibida quando você criar uma tabela estrangeira, o esquema não existe. Execute a etapa anterior para criar um esquema.

```
CREATE SCHEMA tpchobs;
SET current_schema='tpchobs';
DROP FOREIGN table if exists region;
CREATE FOREIGN TABLE REGION
(
        like tpch.region
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/region.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
```



```
);

DROP FOREIGN table if exists nation;
CREATE FOREIGN TABLE NATION
(
        like tpch.nation
)
SERVER gsmpp_server
OPTIONS (
         encoding 'utf8',
         location 'obs://<obs_bucket_name>/tpch/nation.tbl',
         format 'text',
         delimiter '|',
         access_key '<Access_Key_Id>',
         secret_access_key '<Secret_Access_Key>',
         chunksize '64',
         IGNORE_EXTRA_DATA 'on'
);

DROP FOREIGN table if exists supplier;
CREATE FOREIGN TABLE SUPPLIER
(
        like tpch.supplier
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/supplier.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);

DROP FOREIGN table if exists customer;
CREATE FOREIGN TABLE CUSTOMER
(
        like tpch.customer
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/customer.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
DROP FOREIGN table if exists part;
CREATE FOREIGN TABLE PART
(
        like tpch.part

)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/part.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
```



```
DROP FOREIGN table if exists partsupp;
CREATE FOREIGN TABLE PARTSUPP
(
        like tpch.partsupp
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/partsupp.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
DROP FOREIGN table if exists orders;
CREATE FOREIGN TABLE ORDERS
(
        like tpch.orders
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/orders.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
DROP FOREIGN table if exists lineitem;
CREATE FOREIGN TABLE LINEITEM
(
        like tpch.lineitem
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/tpch/lineitem.tbl',
        format 'text',
        delimiter '|',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on'
);
```

Passo 3 Copie e execute as seguintes instruções para importar os dados da tabela estrangeira para a tabela do banco de dados correspondente.

Execute o comando **insert** para importar os dados na tabela estrangeira do OBS para a tabela do banco de dados do GaussDB(DWS). O kernel do banco de dados importa simultaneamente os dados do OBS em alta velocidade para o GaussDB(DWS).

```
INSERT INTO tpch.lineitem SELECT * FROM tpchobs.lineitem;
INSERT INTO tpch.part SELECT * FROM tpchobs.part;
INSERT INTO tpch.partsupp SELECT * FROM tpchobs.partsupp;
INSERT INTO tpch.customer SELECT * FROM tpchobs.customer;
INSERT INTO tpch.supplier SELECT * FROM tpchobs.supplier;
INSERT INTO tpch.nation SELECT * FROM tpchobs.nation;
INSERT INTO tpch.region SELECT * FROM tpchobs.region;
INSERT INTO tpch.orders SELECT * FROM tpchobs.orders;
```

Demora 10 minutos para importar dados.

----Fim

Passo 2: realizar análise de de várias tabelas e análise de temas

A seguir, a consulta TPC-H padrão é usada como exemplo para demonstrar como executar a consulta básica de dados no GaussDB(DWS).

Antes de consultar dados, execute o comando **Analyze** para gerar estatísticas relacionadas à tabela do banco de dados. Os dados de estatísticas são armazenados na tabela do sistema PG_STATISTIC e são úteis quando você executa o planejador, o que fornece um plano de execução de consulta eficiente.

A seguir estão exemplos de consulta:

- **Consulta de receita de um fornecedor em uma região (TPCH-Q5)**

  Ao executar a instrução de consulta TPCH-Q5, você pode consultar as estatísticas de receita de um fornecedor de peças de reposição em uma região. A receita é calculada com base em **sum(l_extendedprice * (1 - l_discount))**. As estatísticas podem ser usadas para determinar se um centro de alocação local precisa ser estabelecido em uma determinada região.

  Copie e execute a seguinte instrução TPCH-Q5 para consulta. Essa instrução apresenta consulta de associação de várias tabelas com **GROUP BY**, **ORDER BY** e **AGGREGATE**.

  ```
  SET current_schema='tpch';
  SELECT
  n_name,
  sum(l_extendedprice * (1 - l_discount)) as revenue
  FROM
  customer,
  orders,
  lineitem,
  supplier,
  nation,
  region
  where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and l_suppkey = s_suppkey
  and c_nationkey = s_nationkey
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'ASIA'
  and o_orderdate >= '1994-01-01'::date
  and o_orderdate < '1994-01-01'::date + interval '1 year'
  group by
  n_name
  order by
  revenue desc;
  ```

- **Consulta de relações entre peças de reposição e fornecedores (TPCH-Q16)**

  Ao executar a instrução de consulta TPCH-Q16, você pode obter o número de fornecedores que podem fornecer peças de reposição com as condições de contribuição especificadas. Esta informação pode ser usada para determinar se há fornecedores suficientes quando a quantidade do pedido é grande e a tarefa é urgente.

  Copie e execute a seguinte instrução TPCH-Q16 para consulta. A instrução apresenta operações de conexão de várias tabelas com subconsulta de group by, sort by, aggregate, deduplicate e NOT IN.

  ```
  SET current_schema='tpch';
  SELECT
  p_brand,
  p_type,
  p_size,
  count(distinct ps_suppkey) as supplier_cnt
  ```



```
FROM
partsupp,
part
where
p_partkey = ps_partkey
and p_brand <> 'Brand#45'
and p_type not like 'MEDIUM POLISHED%'
and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
and ps_suppkey not in (
        select
        s_suppkey
        from
        supplier
        where
        s_comment like '%Customer%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size
limit 100;
```

- **Consulta de perda de receita de pequenos pedidos (TPCH-Q17)**

  Você pode consultar a perda média de receita anual se não houver pedidos pequenos. Filtre os pequenos pedidos que são inferiores a 20% do volume médio de fornecimento e calcule o valor total desses pequenos pedidos para descobrir a perda média de receita anual.

  Copie e execute a seguinte instrução TPCH-Q17 para consulta. A instrução apresenta operações de conexão de várias tabelas com subconsulta de aggregate e aggregate.

```
SET current_schema='tpch';
SELECT
sum(l_extendedprice) / 7.0 as avg_yearly
FROM
lineitem,
part
where
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container = 'MED BOX'
and l_quantity < (
        select 0.2 * avg(l_quantity)
        from lineitem
        where l_partkey = p_partkey
);
```

6.3 Análise de status de operações de uma loja de departamento de varejo

Conhecimento de fundo

Nesta prática, os dados de negócios diários de cada loja de varejo são carregados do OBS para a tabela correspondente no cluster de armazém de dados para resumir e consultar KPIs. Esses dados incluem a rotatividade da loja, o fluxo de clientes, a classificação mensal de vendas, a taxa de conversão mensal do fluxo de clientes, índice mensal de preço-aluguel e as vendas por unidade de área. Este exemplo demonstra a consulta e análise multidimensional do GaussDB(DWS) no cenário de varejo.

NOTA

Os dados de amostra foram carregados na pasta **retail-data** em um bucket do OBS, e todas as contas da Huawei Cloud receberam a permissão somente leitura para acessar o bucket do OBS.

Procedimento geral

Essa prática leva cerca de 60 minutos. O procedimento é os seguintes:

1. **Preparativos**
2. **Passo 1: importar dados de amostra da loja de departamento de varejo**
3. **Passo 2: executar análise de status de operações**

Regiões suportadas

Tabela 6-7 descreve as regiões onde os dados do OBS foram carregados.

Tabela 6-7 Regiões e nomes de bucket do OBS

Região	Bucket de OBS
CN North-Beijing1	dws-demo-cn-north-1
CN North-Beijing2	dws-demo-cn-north-2
CN North-Beijing4	dws-demo-cn-north-4
CN North-Ulanqab1	dws-demo-cn-north-9
CN East-Shanghai1	dws-demo-cn-east-3
CN East-Shanghai2	dws-demo-cn-east-2
CN South-Guangzhou	dws-demo-cn-south-1
CN South-Guangzhou-InvitationOnly	dws-demo-cn-south-4
CN-Hong Kong	dws-demo-ap-southeast-1
AP-Singapore	dws-demo-ap-southeast-3
AP-Bangkok	dws-demo-ap-southeast-2
LA-Santiago	dws-demo-la-south-2
AF-Johannesburg	dws-demo-af-south-1
LA-Mexico City1	dws-demo-na-mexico-1
LA-Mexico City2	dws-demo-la-north-2
RU-Moscow2	dws-demo-ru-northwest-2
LA-Sao Paulo1	dws-demo-sa-brazil-1

Preparativos

- Você registrou uma conta do GaussDB(DWS), e a conta não está em atraso ou congelada.
- Você obteve o AK e SK da conta.
- Um cluster foi criado e conectado usando o Data Studio. Para mais detalhes, veja **Passo 1: criar um cluster** e **Passo 2: usar o Data Studio para conectar-se a um cluster**.

Passo 1: importar dados de amostra da loja de departamento de varejo

Depois de se conectar ao cluster usando a ferramenta de cliente SQL, execute as seguintes operações na ferramenta de cliente SQL para importar os dados de amostra de lojas de departamento de varejo e executar consultas.

Passo 1 Execute a instrução a seguir para criar o banco de dados **retail**:

```
CREATE DATABASE retail encoding 'utf8' template template0;
```

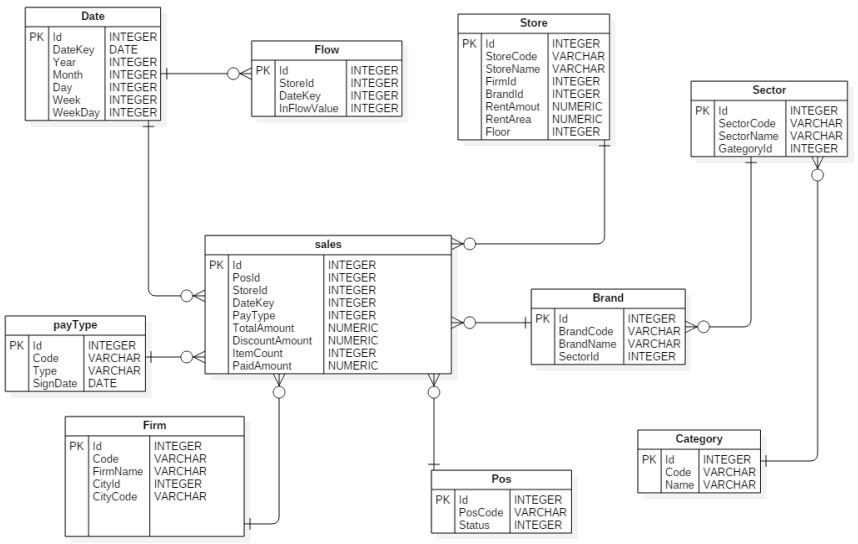
Passo 2 Execute as seguintes etapas para alternar para o novo banco de dados:

1. Na janela **Object Browser** do cliente do Data Studio, clique com o botão direito do mouse na conexão de banco de dados e selecione **Refresh** no menu de atalho. Em seguida, o novo banco de dados é exibido.
2. Clique com o botão direito do mouse no nome do novo banco de dados **retail** e escolha **Connect to DB** no menu de atalho.
3. Clique com o botão direito do mouse no nome do novo banco de dados **retail** e escolha **Open Terminal** no menu de atalho. A janela de comando SQL para conexão com o banco de dados especificado é exibida. Execute os seguintes passos na janela.

Passo 3 Crie uma tabela de banco de dados.

Os dados de amostra consistem em 10 tabelas de banco de dados cujas associações são mostradas em **Figura 6-2**.

Figura 6-2 Tabelas de dados de amostra de lojas de departamento de varejo



Copie e execute as seguintes instruções para alternar para criar uma tabela de banco de dados de informações de lojas de departamento de varejo.


```
CREATE SCHEMA retail_data;
SET current_schema='retail_data';

DROP TABLE IF EXISTS STORE;
CREATE TABLE STORE (
        ID INT,
        STORECODE VARCHAR(10),
        STORENAME VARCHAR(100),
        FIRMID INT,
        FLOOR INT,
        BRANDID INT,
        RENTAMOUNT NUMERIC(18,2),
        RENTAREA NUMERIC(18,2)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS POS;
CREATE TABLE POS(
        ID INT,
        POSCODE VARCHAR(20),
        STATUS INT,
        MODIFICATIONDATE DATE
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS BRAND;
CREATE TABLE BRAND (
        ID INT,
        BRANDCODE VARCHAR(10),
        BRANDNAME VARCHAR(100),
        SECTORID INT
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS SECTOR;
CREATE TABLE SECTOR(
        ID INT,
        SECTORCODE VARCHAR(10),
        SECTORNAME VARCHAR(20),
        CATEGORYID INT
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS CATEGORY;
CREATE TABLE CATEGORY(
        ID INT,
        CODE VARCHAR(10),
        NAME VARCHAR(20)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS FIRM;
CREATE TABLE FIRM(
        ID INT,
        CODE VARCHAR(4),
        NAME VARCHAR(40),
        CITYID INT,
        CITYNAME VARCHAR(10),
        CITYCODE VARCHAR(20)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS DATE;
CREATE TABLE DATE(
        ID INT,
        DATEKEY DATE,
        YEAR INT,
        MONTH INT,
        DAY INT,
```



```
        WEEK INT,
        WEEKDAY INT
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS PAYTYPE;
CREATE TABLE PAYTYPE(
        ID INT,
        CODE VARCHAR(10),
        TYPE VARCHAR(10),
        SIGNDATE DATE
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY REPLICATION;

DROP TABLE IF EXISTS SALES;
CREATE TABLE SALES(
         ID INT,
         POSID INT,
         STOREID INT,
         DATEKEY INT,
         PAYTYPE INT,
         TOTALAMOUNT NUMERIC(18,2),
         DISCOUNTAMOUNT NUMERIC(18,2),
         ITEMCOUNT INT,
         PAIDAMOUNT NUMERIC(18,2)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY HASH(ID);

DROP TABLE IF EXISTS FLOW;
CREATE TABLE FLOW (
         ID INT,
         STOREID INT,
         DATEKEY INT,
         INFLOWVALUE INT
)
WITH (ORIENTATION = COLUMN, COMPRESSION=MIDDLE) DISTRIBUTE BY HASH(ID);
```

Passo 4 Crie uma tabela estrangeira, que é usada para identificar e associar os dados de origem no OBS.

AVISO

- <*obs_bucket_name*> indica o nome do bucket do OBS. Apenas algumas regiões são suportadas. Para obter detalhes sobre as regiões suportadas e os nomes dos bucket do OBS, consulte **Regiões suportadas**. Os clusters do GaussDB(DWS) não oferecem suporte ao acesso entre regiões aos dados do bucket do OBS.
- Nesta prática, a região **CN-Hong Kong** é usada como exemplo. Digite **dws-demo-ap-southeast-1** e substitua <*Access_Key_Id*> e <*Secret_Access_Key*> pelo valor obtido em **Preparativos**.
- // AK e SK codificados rigidamente ou em texto não criptografado são arriscados. Para fins de segurança, criptografe seu AK e SK e armazene-os no arquivo de configuração ou nas variáveis de ambiente.
- Se a mensagem"ERROR: schema "*xxx*" does not exist Position" for exibida quando você criar uma tabela estrangeira, o esquema não existe. Execute a etapa anterior para criar um esquema.

```
CREATE SCHEMA retail_obs_data;
SET current_schema='retail_obs_data';
DROP FOREIGN table if exists SALES_OBS;
CREATE FOREIGN TABLE SALES_OBS
(
```



```
        like retail_data.SALES
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/sales',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);

DROP FOREIGN table if exists FLOW_OBS;
CREATE FOREIGN TABLE FLOW_OBS
(
        like retail_data.flow
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/flow',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);

DROP FOREIGN table if exists BRAND_OBS;
CREATE FOREIGN TABLE BRAND_OBS
(
        like retail_data.brand
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/brand',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);


DROP FOREIGN table if exists CATEGORY_OBS;
CREATE FOREIGN TABLE CATEGORY_OBS
(
       like retail_data.category
)
SERVER gsmpp_server
OPTIONS (
       encoding 'utf8',
       location 'obs://<obs_bucket_name>/retail-data/category',
       format 'csv',
       delimiter ',',
       access_key '<Access_Key_Id>',
       secret_access_key '<Secret_Access_Key>',
       chunksize '64',
       IGNORE_EXTRA_DATA 'on',
       header 'on'
);
```



```
DROP FOREIGN table if exists DATE_OBS;
CREATE FOREIGN TABLE DATE_OBS
(
        like retail_data.date
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/date',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);

DROP FOREIGN table if exists FIRM_OBS;
CREATE FOREIGN TABLE FIRM_OBS
(
        like retail_data.firm
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/firm',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);


DROP FOREIGN table if exists PAYTYPE_OBS;
CREATE FOREIGN TABLE PAYTYPE_OBS
(
        like retail_data.paytype
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/paytype',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);


DROP FOREIGN table if exists POS_OBS;
CREATE FOREIGN TABLE POS_OBS
(
        like retail_data.pos
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/pos',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
```



```
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);

DROP FOREIGN table if exists SECTOR_OBS;
CREATE FOREIGN TABLE SECTOR_OBS
(
        like retail_data.sector
)
SERVER gsmpp_server
OPTIONS (
        encoding 'utf8',
        location 'obs://<obs_bucket_name>/retail-data/sector',
        format 'csv',
        delimiter ',',
        access_key '<Access_Key_Id>',
        secret_access_key '<Secret_Access_Key>',
        chunksize '64',
        IGNORE_EXTRA_DATA 'on',
        header 'on'
);


DROP FOREIGN table if exists STORE_OBS;
CREATE FOREIGN TABLE STORE_OBS
(
         like retail_data.store
)
SERVER gsmpp_server
OPTIONS (
         encoding 'utf8',
         location 'obs://<obs_bucket_name>/retail-data/store',
         format 'csv',
         delimiter ',',
         access_key '<Access_Key_Id>',
         secret_access_key '<Secret_Access_Key>',
         chunksize '64',
         IGNORE_EXTRA_DATA 'on',
         header 'on'
);
```

Passo 5 Copie e execute as seguintes instruções para importar os dados da tabela externa para o cluster:

```
INSERT INTO retail_data.store SELECT * FROM retail_obs_data.STORE_OBS;
INSERT INTO retail_data.sector SELECT * FROM retail_obs_data.SECTOR_OBS;
INSERT INTO retail_data.paytype SELECT * FROM retail_obs_data.PAYTYPE_OBS;
INSERT INTO retail_data.firm SELECT * FROM retail_obs_data.FIRM_OBS;
INSERT INTO retail_data.flow SELECT * FROM retail_obs_data.FLOW_OBS;
INSERT INTO retail_data.category SELECT * FROM retail_obs_data.CATEGORY_OBS;
INSERT INTO retail_data.date SELECT * FROM retail_obs_data.DATE_OBS;
INSERT INTO retail_data.pos SELECT * FROM retail_obs_data.POS_OBS;
INSERT INTO retail_data.brand SELECT * FROM retail_obs_data.BRAND_OBS;
INSERT INTO retail_data.sales SELECT * FROM retail_obs_data.SALES_OBS;
```

Leva algum tempo para importar dados.

Passo 6 Copie e execute a instrução a seguir para criar a exibição **v_sales_flow_details**:

```
SET current_schema='retail_data';
CREATE VIEW v_sales_flow_details AS
SELECT
FIRM.ID FIRMID, FIRM.NAME FIRNAME, FIRM. CITYCODE,
CATEGORY.ID CATEGORYID, CATEGORY.NAME CATEGORYNAME,
SECTOR.ID SECTORID, SECTOR.SECTORNAME,
BRAND.ID BRANDID, BRAND.BRANDNAME,
STORE.ID STOREID, STORE.STORENAME, STORE.RENTAMOUNT, STORE.RENTAREA,
DATE.DATEKEY, SALES.TOTALAMOUNT, DISCOUNTAMOUNT, ITEMCOUNT, PAIDAMOUNT,
INFLOWVALUE
```



```
FROM SALES
INNER JOIN STORE ON SALES.STOREID = STORE.ID
INNER JOIN FIRM ON STORE.FIRMID = FIRM.ID
INNER JOIN BRAND ON STORE.BRANDID = BRAND.ID
INNER JOIN SECTOR ON BRAND.SECTORID = SECTOR.ID
INNER JOIN CATEGORY ON SECTOR.CATEGORYID = CATEGORY.ID
INNER JOIN DATE ON SALES.DATEKEY = DATE.ID
INNER JOIN FLOW ON FLOW.DATEKEY = DATE.ID AND FLOW.STOREID = STORE.ID;
```

----Fim

Passo 2: executar análise de status de operações

O seguinte usa a consulta padrão de informações de varejo de lojas de departamento como um exemplo para demonstrar como executar a consulta básica de dados no GaussDB(DWS).

Antes de consultar dados, execute o comando **Analyze** para gerar estatísticas relacionadas à tabela do banco de dados. Os dados de estatísticas são armazenados na tabela do sistema PG_STATISTIC e são úteis quando você executa o planejador, o que fornece um plano de execução de consulta eficiente.

A seguir estão exemplos de consulta:

- **Consultar a receita mensal de vendas de cada loja**

  Copie e execute as seguintes instruções para consultar a receita total de cada loja em um determinado mês:

  ```
  SET current_schema='retail_data';
  SELECT DATE_TRUNC('month',datekey)
  AT TIME ZONE 'UTC' AS __timestamp,
  SUM(paidamount)
  AS sum__paidamount
  FROM v_sales_flow_details
  GROUP BY DATE_TRUNC('month',datekey) AT TIME ZONE 'UTC'
  ORDER BY SUM(paidamount) DESC;
  ```

- **Consultar a receita de vendas e a relação preço-aluguel de cada loja**

  Copie e execute a seguinte instrução para consultar a receita de vendas e a relação de preço-aluguel de cada loja:

  ```
  SET current_schema='retail_data';
  SELECT firname AS firname,
  storename AS storename,
  SUM(paidamount)
  AS sum__paidamount,
  AVG(RENTAMOUNT)/SUM(PAIDAMOUNT)
  AS rentamount_sales_rate
  FROM v_sales_flow_details
  GROUP BY firname, storename
  ORDER BY SUM(paidamount) DESC;
  ```

- **Analisar a receita de vendas de cada cidade**

  Copie e execute a seguinte instrução para analisar e consultar a receita de vendas de todas as províncias:

  ```
  SET current_schema='retail_data';
  SELECT citycode AS citycode,
  SUM(paidamount)
  AS sum__paidamount
  FROM v_sales_flow_details
  GROUP BY citycode
  ORDER BY SUM(paidamount) DESC;
  ```

- **Analisar e comparando a relação de preço-aluguel e a taxa de conversão do fluxo de clientes de cada loja**


```
SET current_schema='retail_data';
SELECT brandname AS brandname,
firname AS firname,
SUM(PAIDAMOUNT)/AVG(RENTAREA) AS sales_rentarea_rate,
SUM(ITEMCOUNT)/SUM(INFLOWVALUE) AS poscount_flow_rate,
AVG(RENTAMOUNT)/SUM(PAIDAMOUNT) AS rentamount_sales_rate
FROM v_sales_flow_details
GROUP BY brandname,  firname
ORDER BY sales_rentarea_rate DESC;
```

- **Analisar marcas no setor de varejo**

```
SET current_schema='retail_data';
SELECT categoryname AS categoryname,
brandname AS brandname,
SUM(paidamount) AS sum__paidamount
FROM v_sales_flow_details
GROUP BY categoryname,
brandname
ORDER BY sum__paidamount DESC;
```

- **Consultar informações diárias de vendas de cada marca**

```
SET current_schema='retail_data';
SELECT brandname AS brandname,
DATE_TRUNC('day', datekey) AT TIME ZONE 'UTC' AS __timestamp,
SUM(paidamount) AS sum__paidamount
FROM v_sales_flow_details
WHERE datekey >= '2016-01-01 00:00:00'
AND datekey <= '2016-01-30 00:00:00'
GROUP BY brandname,
DATE_TRUNC('day', datekey) AT TIME ZONE 'UTC'
ORDER BY sum__paidamount ASC
LIMIT 50000;
```


7 Gerenciamento de segurança

7.1 Controle de acesso baseado em função (RBAC)

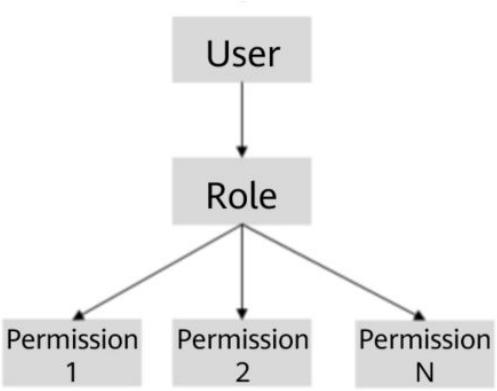
O que é o RBAC?

- O controle de acesso baseado em função (RBAC) é conceder permissões a funções e permitir que os usuários obtenham permissões associando-se a funções.
- Uma função é um conjunto de permissões.
- O RBAC simplifica muito o gerenciamento de permissões.

O que é o modelo RBAC?

Atribua permissões apropriadas às funções.

Associe usuários às funções.



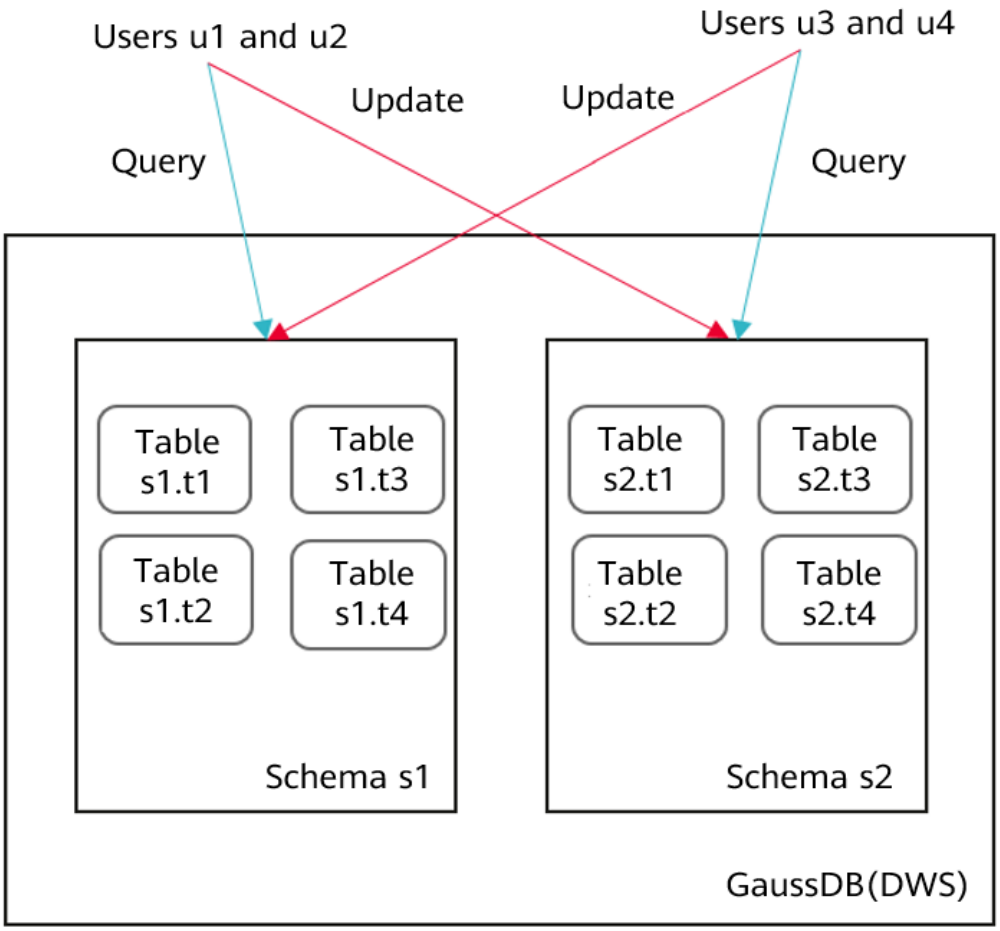
Cenários

Suponha que existam dois esquemas, **s1** e **s2**.

Existem dois grupos de usuários:

- Os usuários **u1** e **u2** podem consultar todas as tabelas em **s1** e atualizar todas as tabelas em **s2**.

- Os usuários **u3** e **u4** podem consultar todas as tabelas em **s2** e atualizar todas as tabelas em **s1**.



Procedimento para conceder permissões

Passo 1 Conecte-se ao banco de dados do DWS como usuário **dbadmin**.

Passo 2 Execute as instruções a seguir para criar os esquemas **s1** e **s2** e os usuários de **u1** a **u4**:

NOTA

Substitua *{password}* pela senha real.

```
CREATE SCHEMA s1;
CREATE SCHEMA s2;
CREATE USER u1 PASSWORD '{password}';
CREATE USER u2 PASSWORD '{password}';
CREATE USER u3 PASSWORD '{password}';
CREATE USER u4 PASSWORD '{password}';
```

Passo 3 Copie e execute as seguintes instruções para criar as tabelas **s1.t1** e **s2.t1**:

```
CREATE TABLE s1.t1 (c1 int, c2 int);
CREATE TABLE s2.t1 (c1 int, c2 int);
```

Passo 4 Execute a instrução a seguir para inserir dados nas tabelas:

```
INSERT INTO s1.t1 VALUES (1,2);
INSERT INTO s2.t1 VALUES (1,2);
```


Passo 5 Execute as seguintes instruções para criar quatro funções, cada uma com a permissão de consulta ou atualização da tabela **s1** ou **s2**:

```
CREATE ROLE rs1_select PASSWORD disable;  -- Permission to query s1
CREATE ROLE rs1_update PASSWORD disable; -- Permission to update s1
CREATE ROLE rs2_select PASSWORD disable; -- Permission to query s2
CREATE ROLE rs2_update PASSWORD disable; -- Permission to update s2
```

Passo 6 Execute as seguintes instruções para conceder as permissões de acesso dos esquemas **s1** e **s2** às funções:

```
GRANT USAGE ON SCHEMA s1, s2 TO rs1_select, rs1_update,rs2_select, rs2_update;
```

Passo 7 Execute as seguintes instruções para conceder permissões específicas às funções:

```
GRANT SELECT ON ALL TABLES IN SCHEMA s1 TO rs1_select; -- Grant the query
permission on all the tables in s1 to the rs1_select role.
GRANT SELECT,UPDATE ON ALL TABLES IN SCHEMA s1 TO rs1_update;  -- Grant the query
and update permissions on all the tables in s1 to the rs1_update role.
GRANT SELECT ON ALL TABLES IN SCHEMA s2 TO rs2_select;  -- Grant the query
permission on all the tables in s2 to the rs2_select role.
GRANT SELECT,UPDATE ON ALL TABLES IN SCHEMA s2 TO rs2_update;  -- Grant the query
and update permissions on all the tables in s2 to the rs2_update role.
```

Passo 8 Execute as seguintes instruções para conceder atribuições aos usuários:

```
GRANT rs1_select, rs2_update TO u1, u2;  -- Users u1 and u2 have the permissions
to query s1 and update s2.
GRANT rs2_select, rs1_update TO u3, u4;  -- Users u3 and u4 have the permissions
to query s2 and update s1.
```

Passo 9 Execute a instrução a seguir para exibir a função vinculada a um usuário específico:

```
\du u1;
```

```
test_lhy=> \du u1
                   List of roles
 Role name | Attributes |          Member of
-----------+------------+-------------------------
 u1        |            | {rs1_select,rs2_update}
```

Passo 10 Comece outra sessão. Conecte-se ao banco de dados como o usuário **u1**.

```
gsql -d gaussdb -h GaussDB(DWS)_EIP -U u1 -p 8000 -r -W {password};
```

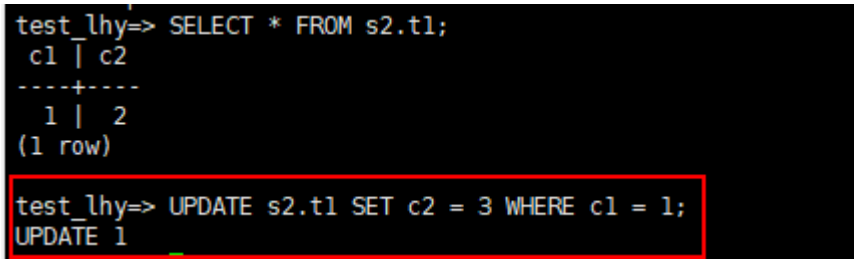
Passo 11 Execute as seguintes instruções na nova sessão para verificar se o usuário **u1** pode consultar, mas não pode atualizar **s1.t1**:

```
SELECT * FROM s1.t1;
UPDATE s1.t1 SET c2 = 3 WHERE c1 = 1;
```

```
test_lhy=> UPDATE s1.t1 SET c1 = 2 WHERE c2 = 2;
ERROR:  Distributed key column can't be updated in current version
test_lhy=> SELECT * FROM s1.t1;
 c1 | c2
----+----
  1 |  2
(1 row)

test_lhy=> UPDATE s1.t1 SET c2 = 3 WHERE c1 = 1;
ERROR:  permission denied for relation t1
```

Passo 12 Execute as seguintes instruções na nova sessão para verificar se o usuário **u1** pode atualizar **s2.t1**:


```
SELECT * FROM s2.t1;
UPDATE s2.t1 SET c2 = 3 WHERE c1 = 1;
```



----Fim

7.2 Criptografia e descriptografia de colunas de dados

A criptografia de dados é amplamente utilizada em vários sistemas de informação como uma tecnologia para efetivamente impedir o acesso não autorizado e evitar o vazamento de dados. Como o núcleo do sistema de informação, o armazém de dados de GaussDB(DWS) também fornece funções de criptografia de dados, incluindo encritação transparente e encritação usando funções SQL. Esta seção descreve a encritação de função SQL.

NOTA

Atualmente, o GaussDB(DWS) não suporta descriptografar dados criptografados em bancos de dados de Oracle, Teradata e MySQL. A criptografia e descriptografia dos bancos de dados de Oracle, Teradata e MySQL são diferentes das do GaussDB(DWS). O GaussDB(DWS) só pode descriptografar dados não criptografados migrados de bancos de dados de Oracle, Teradata e MySQL.

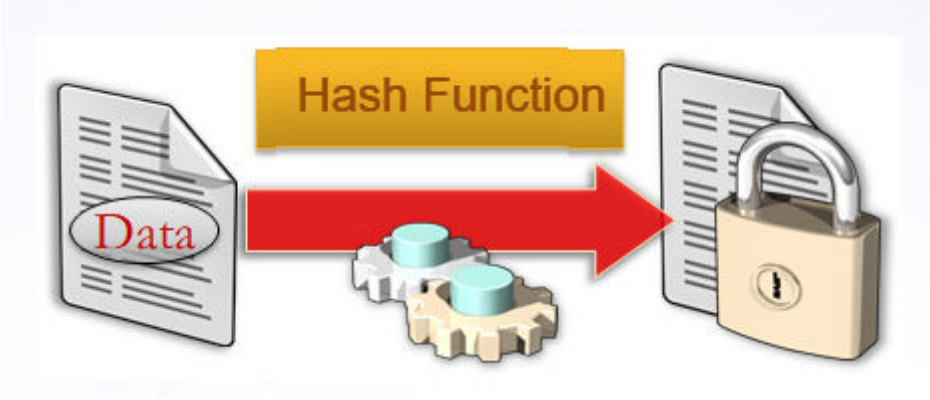
Conhecimento de fundo

- Funções hash

  A função hash também é chamada de algoritmo de resumo. Ela mapeia dados de entrada de um comprimento arbitrário para uma saída de comprimento fixo. Por exemplo, Hash(data)=result. Este processo é irreversível. Ou seja, a função hash não tem uma função inversa, e os dados não podem ser obtidos a partir do resultado. Em cenários em que as senhas de texto não criptografado não devem ser armazenadas (senhas são sensíveis) ou conhecidas pelos administradores de sistema, os algoritmos de hash devem ser usados para armazenar valores de hash unidirecional de senhas.

  Em uso real, os valores de sal e iteração são adicionados para evitar os mesmos valores de hash gerados pelas mesmas senhas, portanto, para evitar ataques à tabela arco-íris.

Figura 7-1 Funções hash



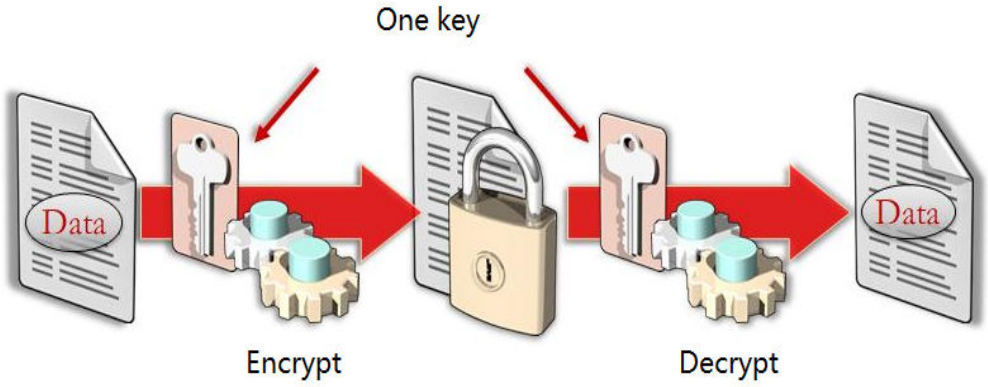
- Algoritmos de criptografia simétrica

  Algoritmos de criptografia simétrica usam a mesma chave para criptografar e descriptografar dados. Existem duas subcategorias de algoritmos de criptografia simétrica: cifras de bloco e cifras de fluxo.

  As cifras de bloco quebram o texto não criptografado em grupos de bits de comprimento fixo conhecidos como blocos e cada bloco é criptografado como uma unidade. E se não houver dados suficientes para preencher completamente um bloco, o "padding" é usado para garantir que os blocos atendam aos requisitos de comprimento fixo. Devido ao preenchimento, o comprimento do texto cifrado obtido por cifras de bloco é maior do que o do texto não criptografado.

  Em cifras de fluxo, as partes de criptografia e descriptografia usam o mesmo fluxo de dados criptografado pseudo-aleatório como chaves, e os dados de texto simples são criptografados sequencialmente por essas chaves. Na prática, os dados são criptografados um bit de cada vez usando uma operação XOR. As cifras de fluxo não precisam ser acolchoadas. Portanto, o comprimento do texto cifrado obtido é o mesmo que o comprimento do texto simples.

Figura 7-2 Algoritmos de criptografia simétrica



Detalhes técnicos

GaussDB(DWS) fornece funções hash e algoritmos criptográficos simétricos para criptografar e descriptografar colunas de dados. As funções hash suportam sha256, sha384, sha512 e SM3. Os algoritmos criptográficos simétricos suportam AES128, AES192, AES256 e SM4.

- Funções hash
  - md5(string)

    Use MD5 para criptografar cadeia e retornar um valor hexadecimal. MD5 é inseguro e não é recomendado.
  - gs_hash(hashstr, hashmethod)

    Obtém a cadeia de resumo de uma cadeia **hashstr** baseada no algoritmo especificado pelo **hashmethod**. **hashmethod** pode ser**sha256**, **sha384**, **sha512** ou **sm3**.
- Algoritmos de criptografia simétrica
  - gs_encrypt(encryptstr, keystr, cryptotype, cryptomode, hashmethod)

    Criptografa uma cadeia **encryptstr** usando a chave **keystr** com base no algoritmo de criptografia especificado por **cryptotype** e **cryptomode** e o algoritmo HMAC especificado por **hashmethod**, e retorna a cadeia criptografada.
  - gs_decrypt(decryptstr, keystr, cryptotype, cryptomode, hashmethod)

    Descriptografa uma cadeia **decryptstr** usando a chave **keystr** com base no algoritmo de criptografia especificado por **cryptotype** e **cryptomode** e o algoritmo HMAC especificado por **hashmethod**, e retorna a cadeia descriptografada. O **keystr** usado para descriptografia deve ser consistente com o usado para criptografia.
  - gs_encrypt_aes128(encryptstr,keystr)

    Criptografa cadeias **encryptstr** usando **keystr** como chave e retorna cadeias criptografadas. O comprimento do **keystr** varia de 1 a 16 bytes.
  - gs_decrypt_aes128(decryptstr,keystr)

    Descriptografa uma cadeia **decryptstr** usando a chave **keystr** e retorna a cadeia descriptografada. O **keystr** usado para descriptografia deve ser consistente com o usado para criptografia. **keystr** não pode estar vazio.

  Para obter mais informações sobre funções, consulte **Uso de funções para criptografia e descriptografia**.

Exemplos

Passo 1 Conecte-se ao banco de dados.

Para obter detalhes, consulte **Uso do cliente da CLI gsql para conectar-se a um cluster**.

Passo 2 Crie a tabela **student** com os atributos **id**, **name** e **score**. Em seguida, use funções hash para criptografar e salvar nomes e use algoritmos criptográficos simétricos para salvar pontuações.

```
CREATE TABLE student (id int, name text, score text, subject text);
INSERT INTO student VALUES (1, gs_hash('alice', 'sha256'), gs_encrypt('95',
'12345', 'aes128', 'cbc', 'sha256'),gs_encrypt_aes128('math', '1234'));
INSERT INTO student VALUES (2, gs_hash('bob', 'sha256'), gs_encrypt('92',
'12345', 'aes128', 'cbc', 'sha256'),gs_encrypt_aes128('english', '1234'));
INSERT INTO student VALUES (3, gs_hash('peter', 'sha256'), gs_encrypt('98',
'12345', 'aes128', 'cbc', 'sha256'),gs_encrypt_aes128('science', '1234'));
```

Passo 3 Consulte a tabela **student** sem usar chaves. O resultado da consulta mostra que os dados criptografados nas colunas nome e pontuação não podem ser visualizados mesmo que você tenha a permissão **SELECT**.

```
select * from student;
 id |                                name
|
score                                                                 |
```



```
                                  subject
----+-------------------------------------------------------------------
+-------------------------------------------------------------------------------------
-----------------------------------------+-----------
-------------------------------------------------------------------------------------
-
  1 | 2bd806c97f0e00af1a1fc3328fa763a9269723c8db8fac4f93af71db186d6e90 |
AAAAAAAAAABAuUC3VQ+MvPCDAaTUyS11e2gGLr4/ATdCUjTEvova3cb/Ba3ZKqIn1yNVGEFBvJnTq/
3sLF4//Gm8qG7AyfNbbqdW3aYErLVpbE/QWFX9Ig== | aFEWQR2gkj
iu6sfsAad+dHzfFDHePZ6xd44zyekh+qVFlh9FODZ0DoaFAJXctwUsiqaiitTxW8cCSEaNjS/E7Ke1ruY=
  2 | 81b637d8fcd2c6da6359e6963113a1170de795e4b725b84d1e0b4cfd9ec58ce9 |
AAAAAAAAAABAuUC3VQ
+MvPCDAaTUyS11taXxAoDqE793hgyCJvC0ESdAX5Mtgdq2LXI1f5ZxraQ73WIJVtIBX8oe3gTDxoXGlHbH
ht4kzM4U8dOwr5rjgg== | aFEWQR2gkj
iu6sfsAad+dM8tPTDo/Pds6ZmqdmjGiKxf39+Wzx5NoQ6c8FrzihnRzgc0fycWSu5YGWNOKYWhRsE84Ac=
  3 | 026ad9b14a7453b7488daa0c6acbc258b1506f52c441c7c465474c1a564394ff |
AAAAAAAAAACnyusORPeApqMUgh56ucQu3uso/
Llw5MbPFMkOXuspEzhhnc9vErwOFe6cuGtx8muEyHCX7V5yXs
+8FxhNh3n5L3419LDWJJLY2O4merHpSg== | zomphRfHV4
H32hTtgkio1PyrobVO8N+hN7kAKwtygKP2E7Aaf1vsjmtLHcL88jyeJNe1lxe0fAvodzPJAxAuV3UJN4M=
(3 rows)
```

Passo 4 Consulte a tabela **student** usando chaves. O resultado da consulta mostra que os dados são descriptografados pela função **gs_decrypt** (correspondente a **gs_encrypt**) e podem ser visualizados.

```
select id, gs_decrypt(score, '12345', 'aes128', 'cbc',
'sha256'),gs_decrypt_aes128(subject, '1234') from student;
 id | gs_decrypt | gs_decrypt_aes128
----+------------+-------------------
  1 | 95         | math
  2 | 92         | english
  3 | 98         | science
(3 rows)
```

----Fim

7.3 Gerenciamento e controle de permissões de dados por meio de exibições

Use modos de exibição para conceder a usuários diferentes a permissão para consultar dados diferentes na mesma tabela, fornecendo gerenciamento e segurança de permissões de dados.

Cenário

Depois de se conectar ao cluster como usuário dbadmin, crie um cliente de tabela de exemplo.

```
CREATE TABLE customer (id bigserial NOT NULL, province_id bigint NOT NULL,
user_info varchar, primary key (id)) DISTRIBUTE BY HASH(id);
```

Insira os dados de teste no cliente da tabela de exemplo.

```
INSERT INTO customer(province_id,user_info) VALUES (1,'Alice'),(1,'Jack'),
(2,'Jack'),(3,'Matu');
INSERT 0 4
```

Consulte a tabela do cliente.

```
SELECT * FROM customer;
 id | province_id | user_info
----+-------------+-----------
  3 |           2 | Jack
  1 |           1 | Alice
  2 |           1 | Jack
```



```
  4 |           3 | Matu
(4 rows)
```

Requisito: o usuário u1 pode visualizar apenas os dados da província 1 (province_id=1) e o usuário u2 pode visualizar apenas os dados da província 2 (province_id=2).

Implementação

Você pode criar uma exibição para atender aos requisitos no cenário anterior. O procedimento é o seguinte:

Passo 1 Depois de se conectar ao cluster como usuário dbadmin, crie as exibições v1 e v2 para as províncias 1 e 2 no modo dbadmin.

Execute a instrução CREATE VIEW para criar a exibição v1 para consultar os dados da província 1.

```
CREATE VIEW v1 AS
    SELECT * FROM customer WHERE province_id=1;
```

Execute a instrução CREATE VIEW para criar a exibição v2 para consultar os dados da província 2.

```
CREATE VIEW v2 AS
    SELECT * FROM customer WHERE province_id=2;
```

Passo 2 Crie os usuários u1 e u2.

```
CREATE USER u1 PASSWORD '*********';
CREATE USER u2 PASSWORD '*********';
```

Passo 3 Execute a instrução GRANT para conceder a permissão de consulta de dados para o usuário de destino.

Conceda a permissão na exibição do esquema correspondente a u1 e u2.

```
GRANT USAGE ON schema dbadmin TO u1,u2;
```

Conceda a u1 a permissão para consultar os dados da província 1 na exibição v1.

```
GRANT SELECT ON v1 TO u1;
```

Conceda a u2 a permissão para consultar os dados da província 2 na exibição V2.

```
GRANT SELECT ON v2 TO u2;
```

----Fim

Verificar o resultado da consulta

- Alterne para a conta u1 para se conectar ao cluster.

  ```
  SET ROLE u1 PASSWORD '*********';
  ```

  Essa interface é usada para consultar a exibição v1. u1 pode consultar apenas os dados da exibição v1.

  ```
  SELECT * FROM dbadmin.v1;
   id | province_id | user_info
  ----+-------------+-----------
    1 |           1 | Alice
    2 |           1 | Jack
  (2 rows)
  ```

  Se u1 tentar consultar dados no modo de exibição v2, as seguintes informações de erro serão exibidas:

  ```
  SELECT * FROM dbadmin.v2;
  ERROR:  SELECT permission denied to user "u1" for relation "dbadmin.v2"
  ```


O resultado mostra que o usuário u1 pode visualizar apenas os dados da província 1 (province_id=1).

- Use a conta u2 para se conectar ao cluster.

```
SET ROLE u2 PASSWORD '*********';
```

Essa interface é usada para consultar a exibição v2. u2 pode consultar apenas os dados da exibição v2.

```
SELECT * FROM dbadmin.v2;
 id | province_id | user_info
----+-------------+-----------
  3 |           2 | Jack
(1 row)
```

Se u2 tentar consultar dados no modo de exibição v1, as seguintes informações de erro serão exibidas:

```
SELECT * FROM dbadmin.v1;
ERROR:  SELECT permission denied to user "u2" for relation "dbadmin.v1"
```

O resultado mostra que o usuário u2 pode visualizar apenas os dados da província 2 (province_id=2).

