Data Warehouse Service (DWS)
8.1.3.333

Best Practices

Issue	01
Date	2024-08-09

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1 Import and Export

1.1 Best Practices for Data Import

Importing Data from OBS in Parallel

- Splitting a data file into multiple files

  Importing a huge amount of data takes a long period of time and consumes many computing resources.

  To improve the performance of importing data from OBS, split a data file into multiple files as evenly as possible before importing it to OBS. The preferred number of split files is an integer multiple of the DN quantity.
- Verifying data files before and after an import

  When importing data from OBS, first import your files to your OBS bucket, and then verify that the bucket contains all the correct files, and only those files.

  After the import is complete, run the **SELECT** statement to verify that the required files have been imported.
- Ensuring no Chinese characters are contained in paths used for importing data to or exporting data from OBS.

Using GDS to Import Data

- Data skew causes the query performance to deteriorate. Before importing all the data from a table containing over 10 million records, you are advised to import some of the data and check whether there is data skew and whether the distribution keys need to be changed. Troubleshoot the data skew if any. It is costly to address data skew and change the distribution keys after a large amount of data has been imported. For details, see "Checking for Data Skew" in the *Data Warehouse Service (DWS) Developer Guide*.
- To speed up the import, you are advised to split files and use multiple Gauss Data Service (GDS) tools to import data in parallel. An import task can be split into multiple concurrent import tasks. If multiple import tasks use the same GDS, you can specify the **-t** parameter to enable GDS multi-thread concurrent import. To prevent physical I/O and network bottleneck, you are advised to mount GDSs to different physical disks and NICs.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- If the GDS I/O and NICs do not reach their physical bottlenecks, you can enable SMP on GaussDB(DWS) for acceleration. SMP will multiply the pressure on GDSs. Note that SMP adaptation is implemented based on the GaussDB(DWS) CPU pressure rather than the GDS pressure. For more information about SMP, see "SMP Manual Optimization Suggestions" in *Data Warehouse Service (DWS) Developer Guide*.
- For the proper communication between GDSs and GaussDB(DWS), you are advised to use 10GE networks. 1GE networks cannot bear the high-speed data transmission, and, as a result, cannot ensure proper communication between GDSs and GaussDB(DWS). To maximize the import rate of a single file, ensure that a 10GE network is used and the data disk group I/O rate is greater than the upper limit of the GDS single-core processing capability (about 400 MB/s).
- Similar to the single-table import, ensure that the I/O rate is greater than the maximum network throughput in the concurrent import.
- You are advised to deploy one or two GDSs on a RAID of a data server.
- It is recommended that the ratio of GDS quantity to DN quantity be in the range of 1:3 to 1:6.
- To improve the efficiency of importing data in batches to column-store partitioned tables, the data is buffered before being written into a disk. You can specify the number of buffers and the buffer size by setting **partition_mem_batch** and **partition_max_cache_size**, respectively. Smaller values indicate the slower the batch import to column-store partitioned tables. The larger the values, the higher the memory consumption.

Using INSERT to Insert Multiple Rows

If the COPY statement cannot be used during data import, you can use multi-row inserts to insert data in batches. Multi-row inserts improve performance by batching up a series of inserts.

The following example inserts three rows into a three-column table using a single **INSERT** statement. This is still a small insert, shown simply to illustrate the syntax of a multi-row insert.

To insert multiple rows of data to the table **customer_t1**, run the following statement:

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

For more details and examples, see "INSERT" in the *Data Warehouse Service (DWS) SQL Syntax Reference*.

Using the COPY Statement to Import Data

The **COPY** statement imports data from local and remote databases in parallel. **COPY** imports large amounts of data more efficiently than **INSERT** statements.

For how to use the **COPY** command, see "Running the COPY FROM STDIN Statement to Import Data" in the *Data Warehouse Service (DWS) Developer Guide*.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Using a gsql Meta-Command to Import Data

The **\copy** command can be used to import data after you log in to a database through any **gsql** client. Compared with the **COPY** command, the **\copy** command directly reads or writes local files instead of reading or writing files on the database server.

Data read or written using the **\copy** command is transferred through the connection between the server and the client and may not be efficient than the **SQL COPY** command. The **COPY** statement is recommended when the amount of data is large.

For how to use the **\copy** command, see "Using a gsql Meta-Command to Import Data" in the *Data Warehouse Service (DWS) Developer Guide*.

NOTE

\copy only applies to small-batch data import with uniform formats but poor error tolerance capability. GDS or **COPY** is preferred for data import.

1.2 Migrating Data from OBS Buckets to a GaussDB(DWS) Cluster

Overview

This practice demonstrates how to upload sample data to OBS and import OBS data to the target table on GaussDB(DWS), helping you quickly learn how to import data from OBS to a GaussDB(DWS) cluster.

You can import data in TXT, CSV, ORC, PARQUET, CARBONDATA, or JSON format from OBS to a GaussDB(DWS) cluster for query.

This tutorial uses the CSV format as an example to perform the following operations:

- Generate data files in CSV format.
- Create an OBS bucket in the same region as the GaussDB(DWS) cluster, and upload data files to the OBS bucket.
- Create a foreign table to import data from the OBS bucket to GaussDB(DWS) clusters.
- Start GaussDB(DWS), create a table, and import data from OBS to the table.
- Analyze import errors based on the information in the error table and correct these errors.

Estimated time: 30 minutes

Preparing Source Data Files

- Data file **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.
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Data file **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.
```

- Data file **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."
```

Step 1 Create a text file, open it using a local editing tool (for example, Visual Studio Code), and copy the sample data to the text file.

Step 2 Choose **Format** > **Encode in UTF-8 without BOM**.

Step 3 Choose **File** > **Save as**.

Step 4 In the displayed dialog box, enter the file name, set the file name extension to .csv, and click **Save**.

----End

Uploading Data to OBS

Step 1 Store the three CSV source data files in the OBS bucket.

1. Log in to the OBS management console.

   Click **Service List** and choose **Object Storage Service** to open the OBS management console.

2. Create a bucket.

   For how to create an OBS bucket, see "Creating a Bucket" in the *Object Storage Service User Guide*.

   For example, create two buckets named **mybucket** and **mybucket02**.

   NOTICE

   Ensure that the two buckets and the GaussDB(DWS) cluster are in the same region.

3. Create a folder.

   For details, see "Creating a Folder" in the *Object Storage Service User Guide*

   Examples:

   - Create a folder named **input_data** in the **mybucket** OBS bucket.
   - Create a folder named **input_data** in the **mybucket02** OBS bucket.

4. Upload the files.

   For details, see "Uploading a File" in the *Object Storage Service User Guide*.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Examples:

- Upload the following data files to the **input_data** folder in the **mybucket** OBS bucket:
  ```
  product_info0.csv
  product_info1.csv
  ```
- Upload the following data file to the **input_data** folder in the **mybucket02** OBS bucket:
  ```
  product_info2.csv
  ```

Step 2 Grant the OBS bucket read permission for the user who will import data.

When importing data from OBS to a cluster, the user must have the read permission for the OBS buckets where the source data files are located. You can configure the ACL for the OBS buckets to grant the read permission to a specific user.

For details, see "Configuring a Bucket ACL" in the *Object Storage Service Usage Guide*.

----End

Creating a Foreign Table

Step 1 Connect to the GaussDB(DWS) database.

Step 2 Create a foreign table.

NOTE

- ACCESS_KEY and SECRET_ACCESS_KEY

  These parameters specify the AK and SK used to access OBS by a user. Replace them with the actual AK and SK.

  For how to obtain an access key, see section "Data Import" > "Importing Data from OBS in Parallel" > "Creating Access Keys (AK and SK)" in the *Data Warehouse Service (DWS) Developer Guide*.

- // Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and store them in the configuration file or environment variables.

```
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',
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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';
```

If the following information is displayed, the foreign table has been created:

```
CREATE FOREIGN TABLE
```

----End

Importing Data

Step 1 Create a table named **product_info** in the GaussDB(DWS) database to store the data imported from 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);
```

Step 2 Run **INSERT** to import data from OBS to the target table **product_info** through the foreign table **product_info_ext**.

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

Step 3 Run **SELECT** to view the data imported from OBS to GaussDB(DWS).

```
SELECT * FROM product_info;
```

The following information is displayed at the end of the query result:

```
(20 rows)
```

Step 4 Run **VACUUM FULL** on the **product_info** table.

```
VACUUM FULL product_info;
```

Step 5 Update statistics of the **product_info** table.

```
ANALYZE product_info;
```

----End

Deleting Resources

Step 1 If you have performed queries after importing data, run the following statement to delete the target table:

```
DROP TABLE product_info;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

If the following output is displayed, the foreign table has been deleted:

```
DROP TABLE
```

Step 2 Run the following statement to delete the foreign table:

```
DROP FOREIGN TABLE product_info_ext;
```

If the following output is displayed, the foreign table has been deleted:

```
DROP FOREIGN TABLE
```

----End

1.3 Using GDS to Import Table Data from a Remote Server to a GaussDB(DWS) Cluster

Overview

This practice demonstrates how to use General Data Service (GDS) to import data from a remote server to GaussDB(DWS).

GaussDB(DWS) allows you to import data in TXT, CSV, or FIXED format.

In this tutorial, you will:

- Generate the source data files in CSV format to be used in this tutorial.
- Upload the source data files to a data server.
- Create foreign tables used for importing data from a data server to GaussDB(DWS) through GDS.
- Start GaussDB(DWS), create a table, and import data to the table.
- Analyze import errors based on the information in the error table and correct these errors.

Preparing an ECS as the GDS Server

For details about how to create a Linux ECS, see "Creating an ECS" in the *Elastic Cloud Server User Guide*. After the creation, log in to the ECS by referring to **Logging In to a Linux ECS**.

NOTE

- The ECS OS must be supported by the GDS package.
- The ECS and GaussDB(DWS) are in the same region, VPC, and subnet.
- The ECS security group rule must allow access to the GaussDB(DWS) cluster, that is, the inbound rule of the security group is as follows:
  - **Protocol**: **TCP**
  - **Port**: **5000**
  - **Source**: Select **IP Address** and enter the IP address of the GaussDB(DWS) cluster, for example, **192.168.0.10/32**.
- If the firewall is enabled in the ECS, ensure that the listening port of GDS is enabled on the firewall:
  ```
  iptables  -I INPUT -p tcp -m tcp --dport <gds_port> -j ACCEPT
  ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Downloading the GDS Package

Step 1 Log in to the GaussDB(DWS) console.

Step 2 In the navigation tree on the left, choose **Management** > **Client Connections**.

Step 3 Select the GDS client of the corresponding version from the drop-down list of **CLI Client**.

Select a version based on the cluster version and the OS where the client is installed.

NOTE

The CPU architecture of the client must be the same as that of the cluster. If the cluster uses the x86 specifications, select the x86 client.

Step 4 Click **Download**.

----End

Preparing Source Data Files

- Data file **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.
```

- Data file **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.
```

- Data file **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."
```

Step 1 Create a text file, open it using a local editing tool (for example, Visual Studio Code), and copy the sample data to the text file.

Step 2 Choose **Format** > **Encode in UTF-8 without BOM**.

Step 3 Choose **File** > **Save as**.

Step 4 In the displayed dialog box, enter the file name, set the file name extension to .csv, and click **Save**.

Step 5 Log in to the GDS server as user **root**.

Step 6 Create the **/input_data** directory for storing the data file.

```
mkdir -p /input_data
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 7 Use MobaXterm to upload source data files to the created directory.

----End

Installing and Starting GDS

Step 1 Log in to the GDS server as user **root** and create the **/opt/bin/dws** directory for storing the GDS package.

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

Step 2 Upload the GDS package to the created directory.

For example, upload the **dws_client_8.1.*x*_redhat_x64.zip** package to the created directory.

Step 3 Go to the directory and decompress the package.

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

Step 4 Create a user (**gds_user**) and the user group (**gdsgrp**) to which the user belongs. This user is used to start GDS and must have the permission to read the source data file directory.

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

Step 5 Change the owner of the GDS package and source data file directory to **gds_user** and change the user group to **gdsgrp**.

```
chown -R gds_user:gdsgrp
chown -R gds_user:gdsgrp /input_data
```

Step 6 Switch to user **gds_user**.

```
su - gds_user
```

If the current cluster version is 8.0.*x* or earlier, skip **Step 7** and go to **Step 8**.

If the current cluster version is 8.1.*x* or later, go to the next step.

Step 7 Execute the script on which the environment depends (applicable only to 8.1.*x*).

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

Step 8 Start 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
```

Replace the italic parts as required.

- **-d** *dir*: directory for storing data files that contain data to be imported. This practice uses **/input_data/** as an example.
- **-p** *ip:port*: listening IP address and port for GDS. The default value is **127.0.0.1**. Replace it with the IP address of a 10GE network that can communicate with GaussDB(DWS). The port number ranges from 1024 to 65535. The default value is **8098**. This practice uses **192.168.0.90:5000** as an example.
- **-H** *address_string*: hosts that are allowed to connect to and use GDS. The value must be in CIDR format. Set this parameter to enable a GaussDB(DWS) cluster to access GDS for data import. Ensure that the network segment covers all hosts in a GaussDB(DWS) cluster.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **-l** *log_file*: GDS log directory and log file name. This practice uses **/opt/bin/dws/gds/gds_log.txt** as an example.
- **-D**: GDS in daemon mode. This parameter is used only in Linux.

----End

Creating a Foreign Table

Step 1 Use an SQL client to connect to the GaussDB(DWS) database.

Step 2 Create the following foreign table:

> ⚠ CAUTION
>
> **LOCATION**: Replace it with the actual GDS address and port number.

```
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';
```

If the following information is displayed, the foreign table has been created:

```
CREATE FOREIGN TABLE
```

----End

Importing Data

Step 1 Run the following statements to create the **product_info** table in GaussDB(DWS) to store imported data:

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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);
```

Step 2 Import data from source data files to the **product_info** table through the foreign table **product_info_ext**.

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

If the following information is displayed, the data is successfully imported:

```
INSERT 0 20
```

Step 3 Run the **SELECT** statement to view the data imported to GaussDB(DWS).

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

If the following information is displayed, the data has been imported:

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

Step 4 Run **VACUUM FULL** on the **product_info** table.

```
VACUUM FULL product_info
```

Step 5 Update statistics of the **product_info** table.

```
ANALYZE product_info;
```

----End

Stopping GDS

Step 1 Log in to the data server where GDS is installed as user **gds_user**.

Step 2 Perform the following operations to stop GDS:

1. Query the GDS process ID. The GDS process ID is **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. Run the **kill** command to stop GDS. **128954** indicates the GDS process ID.

   ```
   kill -9 128954
   ```

----End

Deleting Resources

Step 1 Run the following command to delete the target table **product_info**:

```
DROP TABLE product_info;
```

If the following information is displayed, the table has been deleted:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
DROP TABLE
```

Step 2 Run the following command to delete the foreign table **product_info_ext**:

```
DROP FOREIGN TABLE product_info_ext;
```

If the following information is displayed, the table has been deleted:

```
DROP FOREIGN TABLE
```

----End

1.4 Migrating Data Between GaussDB(DWS) Clusters Using Foreign Tables

In the era of big data convergent analysis, GaussDB(DWS) clusters in the same region can communicate with each other. This practice demonstrates how to import data from a remote GaussDB(DWS) cluster to the local GaussDB(DWS) cluster using foreign tables.

The demonstration procedure is as follows: Install the gsql database client on an ECS, connect to GaussDB(DWS) using gsql, and import data from the remote GaussDB(DWS) using a foreign table.

General Procedure

This practice takes about 40 minutes. The basic process is as follows:

1. **Preparations**
2. **Creating an ECS**
3. **Creating a Cluster and Downloading the Tool Package**
4. **Importing Data Sources Using GDS**
5. **Importing Remote GaussDB(DWS) Data Using a Foreign Table**

Preparations

Log in to ManageOne Operation Portal using a VDC account.

Creating an ECS

For details, see "Creating an ECS" in the *Elastic Cloud Server User Guide*. When the ECS is created, log in to the ECS. For details, see "Remotely Logging In to a Linux ECS Using a Password (SSH)".

NOTICE

When creating an ECS, ensure that the ECS and the GaussDB(DWS) clusters to be created are in the same VPC subnet and in the same region and AZ . The ECS OS is the same as that of the gsql client or GDS (CentOS 7.6 is used as an example), and the password is used for login.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Creating a Cluster and Downloading the Tool Package

Step 1 Log in to the ManageOne operation plane.

Step 2 Choose **Service List** > **EI Enterprise Intelligence** > **Data Warehouse Service**. On the page that is displayed, click **Create Cluster** in the upper right corner.

Step 3 Configure the parameters according to **Table 1-1**.

Table 1-1 Software configuration

Parameter	Configuration
Region	Select a region.
AZ	AZ2
Resource	Standard data warehouse
Compute Resource	ECS
Storage Type	Cloud SSD
CPU Architecture	x86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD)
Hot Storage	100 GB/node
Nodes	3
Cluster Name	dws-demo01
Administrator Account	dbadmin
Administrator Password	User-defined password
Confirm Password	password
Database Port	8000
VPC	vpc-default

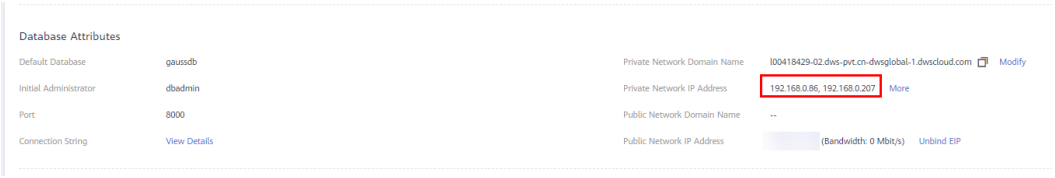
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Configuration
Subnet	subnet-default(192.168.0.0/24) **NOTICE** Ensure that the cluster and the ECS are in the same VPC subnet.
Security Group	Automatic creation
EIP	Automatically assign
Bandwidth	1 Mbit/s
Advanced Settings	Default

Step 4 Confirm the information, click **Create Now**, and then click **Submit**.

Step 5 Wait for about 10 minutes. After the cluster is created, click the cluster name to go to the **Basic Information** page. Choose **Network**, click a security group name, and verify that a security group rule has been added. In this example, the client IP address is 192.168.0.*x* (the private network IP address of the ECS where gsql is located is 192.168.0.90). Therefore, you need to add a security group rule in which the IP address is 192.168.0.0/24 and port number is 8000.

Step 6 Return to the **Basic Information** tab of the cluster and record the value of **Private Network IP Address**.



Step 7 Return to the home page of the GaussDB(DWS) console. In the navigation tree on the left, choose **Management** > **Client Connections**, select the appropriate ECS OS (such as **Redhat x86_64** for **CentOS 7.6**), and click **Download** to save the tool package to your local PC. The tool package contains the gsql client and GDS.

Step 8 Repeat **Step 1** to **Step 6** to create a second GaussDB(DWS) cluster and set its name to **dws-demo02**.

----End

Preparing Source Data

Step 1 Create the following three CSV files in the specified directory on the local PC:

- Data file **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.
```

- Data file **product_info1.csv**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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.
```

- Data file **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."
```

Step 2 Log in to the created ECS as user **root** and run the following command to create a data source file directory:

mkdir -p /input_data

Step 3 Use a file transfer tool to upload the preceding data files to the **/input_data** directory of the ECS.

----End

Importing Data Sources Using GDS

Step 1 Log in to the ECS as user **root** and use a file transfer tool to upload the downloaded tool package in **Step 7** to the **/opt** directory.

Step 2 Decompress the tool package in the **/opt** directory.

cd /opt

unzip dws_client_8.1.x_redhat_x64.zip

Step 3 Create a GDS user and change the owners of the data source and GDS directories.

groupadd gdsgrp

useradd -g gdsgrp gds_user

chown -R gds_user:gdsgrp /opt/gds

chown -R gds_user:gdsgrp /input_data

Step 4 Switch to user **gds_user**.

su - gds_user

Step 5 Import the GDS environment variables.

NOTE

This step is required only for 8.1.*x* or later. For earlier versions, skip this step.

cd /opt/gds/bin

source gds_env

Step 6 Start GDS.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

/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*: directory for storing data files that contain data to be imported. This practice uses **/input_data/** as an example.
- **-p** *ip:port*: listening IP address and port for GDS. Set this parameter to the private network IP address of the ECS where GDS is installed so that GDS can communicate with GaussDB(DWS). In this example, **192.168.0.90:5000** is used.
- **-H** *address_string*: hosts that are allowed to connect to and use GDS. The value must be in CIDR format. In this example, the network segment of the GaussDB(DWS) private network IP address is used.
- **-l** *log_file*: GDS log directory and log file name. In this example, **/opt/gds/gds_log.txt** is used.
- **-D**: GDS in daemon mode.

Step 7 Connect to the first GaussDB(DWS) cluster using gsql.

1. Run the **exit** command to switch to user **root**, go to the **/opt** directory of the ECS, and import the environment variables of gsql.

   exit

   cd /opt

   source gsql_env.sh

2. Go to the **/opt/bin** directory and connect to the first GaussDB(DWS) cluster using gsql.

   cd /opt/bin

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

   - **-d**: name of the connected database. In this example, the default database **gaussdb** is used.
   - **-h**: private network IP address of the connected GaussDB(DWS) database queried in **Step 6**. In this example, **192.168.0.8** is used.
   - **-p**: GaussDB(DWS) port. The value is **8000**.
   - **-U**: database administrator. The value defaults to **dbadmin**.
   - **-W**: administrator password, which is set during cluster creation in **Step 3**. In this example, replace *password* with your actual password.

Step 8 Create a common user **leo** and grant the user the permission for creating foreign tables.

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

Step 9 Switch to user **leo** and create a GDS foreign table.

NOTE

Set **LOCATION** to the GDS listening IP address and port number obtained in **Step 6**, for example, **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,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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';
```

Step 10 Create a local table.

```
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);
```

Step 11 Import data from the GDS foreign table and check whether the data is successfully imported.

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

----End

Importing Remote GaussDB(DWS) Data Using a Foreign Table

Step 1 Connect to the second cluster on the ECS by referring to **Step 7**. Change the connection address to the address of the second cluster. In this example, **192.168.0.86** is used.

Step 2 Create a common user **jim** and grant the user the permission for creating foreign tables and servers. The value of **FOREIGN DATA WRAPPER** is **gc_fdws**.

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 3 Switch to user **jim** and create a server.

```
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**: private network IP addresses and port number of the first cluster obtained in **Step 6**. In this example, **192.168.0.8:8000** and **192.168.0.158:8000** are used.
- **dbname**: database name of the first connected cluster. In this example, **gaussdb** is used.
- **username**: username of the first connected cluster. In this example, **leo** is used.
- **password**: user password

Step 4 Create a foreign table.

NOTICE

The columns and constraints of the foreign table must be consistent with those of the table to be accessed.

```
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**: name of the server created in the previous step. In this example, **server_remote** is used.
- **schema_name**: schema name of the first cluster to be accessed. In this example, **leo** is used.
- **table_name**: table name of the first cluster to be accessed obtained in **Step 10**. In this example, **product_info** is used.
- **encoding**: The value must be the same as that of the first cluster obtained in **Step 9**. In this example, **utf8** is used.

Step 5 View the created server and foreign table.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

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

Step 6 Create a local table.

NOTICE

The columns and constraints of the table must be consistent with those of the table to be accessed.

```
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);
```

Step 7 Import data to the local table using the foreign table.

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

Step 8 Query the foreign table without importing data.

```
SELECT * FROM region;
```

----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2 Data Development

2.1 Cutting Costs by Switching Between Cold and Hot Data Storage in GaussDB(DWS)

Scenarios

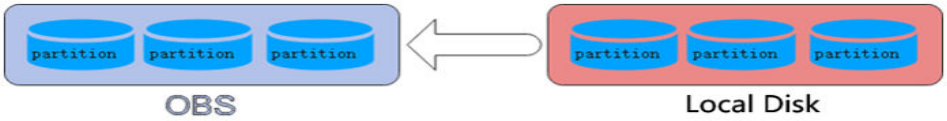
In massive big data scenarios, with the growing of data, data storage and consumption increase rapidly. The need for data may vary in different time periods, therefore, data is managed in a hierarchical manner, improving data analysis performance and reducing service costs. In some data usage scenarios, data can be classified into hot data and cold data by accessing frequency.

Hot and cold data is classified based on the data access frequency and update frequency.

- Hot data: Data that is frequently accessed and updated and requires fast response.
- Cold data: Data that cannot be updated or is seldom accessed and does not require fast response

You can define cold and hot management tables to switch cold data that meets the specified rules to OBS for storage. Cold and hot data can be automatically determined and migrated by partition.

Figure 2-1 Hot and cold data management



The hot and cold partitions can be switched based on LMT (Last Modify Time) and HPN (Hot Partition Number) policies. LMT indicates that the switchover is performed based on the last update time of the partition, and HPN indicates that the switchover is performed based on the number of reserved hot partitions.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **LMT**: Switch the hot partition data that is not updated in the last *[day]* days to the OBS tablespace as cold partition data. *[day]* is an integer ranging from 0 to 36500, in days.
- HPN: indicates the number of hot partitions to be reserved. During the cold and hot switchover, data needs to be migrated to OBS. HPN is an integer ranging from 0 to 1600.

Constraints

- If a table has both cold and hot partitions, the query becomes slow because cold data is stored on OBS and the read/write speed are lower than those of local queries.
- Currently, cold and hot tables support only column-store partitioned tables of version 2.0. Foreign tables do not support cold and hot partitions.
- Only hot data can be switched to cold data. Cold data cannot be switched to hot data.

Procedure

This practice takes about 30 minutes. The basic process is as follows:

1. **Creating a cluster**.
2. **Using the gsql CLI Client to Connect to a Cluster**.
3. **Creating Hot and Cold Tables**.
4. **Hot and Cold Data Switchover**.
5. **Viewing Data Distribution in Hot and Cold Tables**.

Creating a cluster

Step 1 Log in to the ManageOne operation plane.

Step 2 Choose **Service List** > **EI Enterprise Intelligence** > **Data Warehouse Service**. On the page that is displayed, click **Create Cluster** in the upper right corner.

Step 3 Configure the parameters according to **Table 2-1**.

Table 2-1 Software configuration

Parameter	Configuration
Region	Select a region.
AZ	AZ2
Product	Standard data warehouse
CPU Architecture	X86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD)

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Configuration
Nodes	3
Cluster Name	dws-demo
Administrator Account	dbadmin
Administrator Password	N/A
Confirm Password	N/A
Database Port	8000
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24)
Security Group	Automatic creation
EIP	Automatically assign
Bandwidth	1Mbit/s
Advanced Settings	Default

Step 4 Confirm the information, click **Create Now**, and then click **Submit**.

Step 5 Wait about 6 minutes. After the cluster is created, click next to the cluster name. On the displayed cluster information page, record the value of **Public Network Address**.

----End

Using the gsql CLI Client to Connect to a Cluster

Step 1 On the left of the GaussDB(DWS) console, choose **Management** > **Client Connections**. In the **CLI Client** area, select the corresponding version and download the gsql client.

Step 2 Decompress the client.

```
cd <Path_for_storing_the_client> unzip dws_client_8.1.x_redhat_x64.zip
```

Where,

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- <*Path_for_storing_the_client*>: Replace it with the actual path.
- *dws_client_8.1.x_redhat_x64.zip*: This is the client tool package name of **RedHat x64**. Replace it with the actual name.

Step 3 Configure the GaussDB(DWS) client.

```
source gsql_env.sh
```

If the following information is displayed, the gsql client is successfully configured:

```
All things done.
```

Step 4 Use the gsql client to connect to a GaussDB(DWS) database (using the password you defined when creating the cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

If the following information is displayed, the connection succeeded:

```
gaussdb=>
```

----End

Creating Hot and Cold Tables

Create a column-store cold and hot data management table **lifecycle_table** and set the hot data validity period LMT to 100 days.

```
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;
```

Hot and Cold Data Switchover

Switch cold data to the OBS tablespace.

- Automatic switchover: The scheduler automatically triggers the switchover at 00:00 every day.

  You can use the **pg_obs_cold_refresh_time(table_name, time)** function to customize the automatic switchover time. For example, set the automatic triggering time to 06:30 every morning.

  ```
  SELECT * FROM pg_obs_cold_refresh_time('lifecycle_table', '06:30:00');
  pg_obs_cold_refresh_time
  --------------------------
   SUCCESS
  (1 row)
  ```

- Manual

  Run the **ALTER TABLE** statement to manually switch a single table.

  ```
  ALTER TABLE lifecycle_table refresh storage;
  ALTER TABLE
  ```

  Use the **pg_refresh_storage()** function to switch all hot and cold tables in batches.

  ```
  SELECT pg_catalog.pg_refresh_storage();
   pg_refresh_storage
  --------------------
   (1,0)
  (1 row)
  ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Viewing Data Distribution in Hot and Cold Tables

- View the data distribution in a single table:

```
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)
```

- View data distribution in all hot and cold tables:

```
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)
```

2.2 Cutting Partition Maintenance Costs for the E-commerce and IoT Industries by Leveraging GaussDB(DWS)'s Automatic Partition Management Feature

Scenarios

For partition tables whose partition columns are time, the automatic partition management function can be added to automatically create partitions and delete expired partitions, reducing partition table maintenance costs and improving query performance. To facilitate data query and maintenance, the time column is often used as the partition column of a partitioned table that stores time-related data, such as e-commerce order information and real-time IoT data. When the time-related data is imported to a partitioned table, the table should have partitions of the corresponding time ranges. Common partition tables do not automatically create new partitions or delete expired partitions. Therefore, maintenance personnel need to periodically create new partitions and delete expired partitions, leading to increased O&M costs.

GaussDB(DWS) has introduced an automatic partition management feature to address this issue. You can set the table-level parameters **period** and **ttl** to enable the automatic partition management function, which automatically creates partitions and deletes expired partitions, reducing partition table maintenance costs and improving query performance.

period: interval for automatically creating partitions. The default value is 1 day. The value range is 1 hour ~ 100 years.

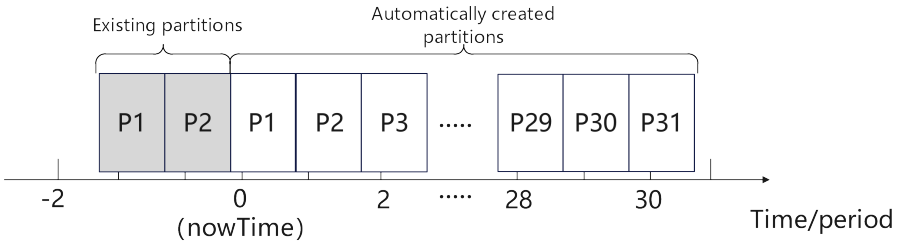
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

ttl: time for automatically eliminate partitions. The value range is 1 hour ~ 100 years. Partition elimination occurs when nowtime - Partition boundary > ttl, resulting in the removal of qualifying partitions.

- Automatic partition creation

  One or more partitions are automatically created at the interval specified by **period** to make the maximum partition boundary time greater than nowTime + 30 x period. As long as there is an automatically created partition, real-time data will not fail to be imported within the next 30 periods.

  Figure 2-2 Automatic partition creation

- Automatically deleting expired partitions

  Partitions whose boundary time is earlier than **nowTime-ttl** are considered expired partitions. The automatic partition management function traverses all partitions and deletes expired partitions after each **period**. If all partitions are expired partitions, the system retains one partition and truncates the table.

Constraints

When using the partition management function, ensure that the following requirements are met:

- It cannot be used on midrange servers, acceleration clusters, or stand-alone clusters.
- It can be used in clusters of version 8.1.3 or later.
- It can only be used for row-store range partitioned tables, column-store range partitioned tables, time series tables, and cold and hot tables.
- The partition key must be unique and its type must be timestamp, timestamptz, or date.
- The maxvalue partition is not supported.
- The value of (nowTime - boundaryTime)/period must be less than the maximum number of partitions. **nowTime** indicates the current time, and **boundaryTime** indicates the earliest partition boundary time.
- The values of **period** and **ttl** range from 1 hour to 100 years. In addition, in a database compatible with Teradata or MySQL, if the partition key type is date, the value of period cannot be less than 1day.
- The table-level parameter **ttl** cannot exist independently. You must set **period** in advance or at the same time, and the value of **ttl** must be greater than or equal to that of **period**.
- During online cluster scale-out, partitions cannot be automatically added. Partitions reserved each time partitions are added will ensure that services are not affected.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Creating an ECS

For details, see "Creating an ECS" in the *Elastic Cloud Server User Guide*. When the ECS is created, log in to the ECS. For details, see "Remotely Logging In to a Linux ECS Using a Password (SSH)".

NOTICE

When creating an ECS, ensure that the ECS is in the same region, AZ, and VPC subnet as the stream data warehouse. Select the OS used by the gsql client (CentOS 7.6 is used as an example) as the ECS OS, and select using passwords to log in.

Creating a cluster

Step 1 Log in to the ManageOne operation plane.

Step 2 Choose **Service List** > **EI Enterprise Intelligence** > **Data Warehouse Service**. On the page that is displayed, click **Create Cluster** in the upper right corner.

Step 3 Configure the parameters according to **Table 2-2**.

Table 2-2 Software configuration

Parameter	Configuration
Region	Select a region.
AZ	AZ2
Product	Standard data warehouse
CPU Architecture	X86
Node Flavor	dws2.m6.4xlarge.8 (16 vCPUs | 128 GB | 2000 GB SSD)
Nodes	3
Cluster Name	dws-demo
Administrator Account	dbadmin
Administrator Password	N/A
Confirm Password	N/A

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Configuration
Database Port	8000
VPC	vpc-default
Subnet	subnet-default(192.168.0.0/24)
Security Group	Automatic creation
EIP	Automatically assign
Bandwidth	1Mbit/s
Advanced Settings	Default

Step 4 Confirm the information, click **Create Now**, and then click **Submit**.

Step 5 Wait about 6 minutes. After the cluster is created, click next to the cluster name. On the displayed cluster information page, record the value of **Public Network Address**.

----End

Using the gsql CLI Client to Connect to a Cluster

Step 1 On the left of the GaussDB(DWS) console, choose **Management** > **Client Connections**. In the **CLI Client** area, select the corresponding version and download the gsql client.

Step 2 Decompress the client.

```
cd <Path_for_storing_the_client> unzip dws_client_8.1.x_redhat_x64.zip
```

Where,

- <*Path_for_storing_the_client*>: Replace it with the actual path.
- *dws_client_8.1.x_redhat_x64.zip*: This is the client tool package name of **RedHat x64**. Replace it with the actual name.

Step 3 Configure the GaussDB(DWS) client.

```
source gsql_env.sh
```

If the following information is displayed, the gsql client is successfully configured:

```
All things done.
```

Step 4 Use the gsql client to connect to a GaussDB(DWS) database (using the password you defined when creating the cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

If the following information is displayed, the connection succeeded:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
gaussdb=>
```

----End

Automatic partition management

The partition management function is bound to the table-level parameters **period** and **ttl**. Automatic partition creation is enabled with the enabling of **period**, and automatic partition deletion is enabled with the enabling of **ttl**. 30 seconds after **period** or **ttl** is set, the automatic partition creation or deletion works for the first time.

You can enable the partition management function in either of the following ways:

- Specify **period** and **ttl** when creating a table.

  This way is applicable when you create a partition management table. There are two syntaxes for creating a partition management table. One specifies partitions, and the other does not.

  If partitions are specified when a partition management table is created, the syntax rules are the same as those for creating a common partitioned table. The only difference is that the syntax specifies the table-level parameters **period** and **ttl**.

  The following example shows how to create a partition management table **CPU1** and specify partitions.

  ```
  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')
  );
  ```

  When creating a partition management table, you can specify only the partition key but not partitions. In this case, two default partitions will be created with **period** as the partition time range. The boundary time of the first default partition is the first hour, day, week, month, or year past the current time. The time unit is selected based on the maximum unit of PERIOD. The boundary time of the second default partition is the boundary time of the first partition plus PERIOD. Assume that the current time is 2023-02-17 16:32:45, and the boundary of the first default partition is described in the following table.

  Table 2-3 Description of the period parameter

period	Maximum PERIOD Unit	Boundary of First Default Partition
1hour	Hour	2023-02-17 17:00:00
1day	Day	2023-02-18 00:00:00
1month	Month	2023-03-01 00:00:00

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

period	Maximum PERIOD Unit	Boundary of First Default Partition
13months	Year	2024-01-01 00:00:00

Run the following command to create the partition management table **CPU2** with no partitions specified:

```
CREATE TABLE CPU2(
    id integer,
    IP text,
    time timestamp
) with (TTL='7 days',PERIOD='1 day')
partition by range(time);
```

- Run the **ALTER TABLE RESET** command to set **period** and **ttl**.

  This method is used to add the partition management function to an ordinary partitioned table that meets the partition management constraints.

  - Run the following command to create an ordinary partition table **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')
    );
    ```

  - To enable the automatic partition creation and deletion functions, run the following command:

    ```
    ALTER TABLE CPU3 SET (PERIOD='1 day',TTL='7 days');
    ```

  - To enable only the automatic partition creation function, run the following command:

    ```
    ALTER TABLE CPU3 SET (PERIOD='1 day');
    ```

  - To enable only the automatic partition deletion function, run the following command (If automatic partition creation is not enabled in advance, the operation will fail):

    ```
    ALTER TABLE CPU3 SET (TTL='7 days');
    ```

  - Modify the **period** and **ttl** parameters to modify the partition management function.

    ```
    ALTER TABLE CPU3 SET (TTL='10 days',PERIOD='2 days');
    ```

- Disabling the partition management function

  You can run the **ALTER TABLE RESET** command to delete the table-level parameters **period** and **ttl** to disable the partition management function.

  NOTE

  - The **period** cannot be deleted separately with **TTL**.
  - The time series table does not support **ALTER TABLE RESET**.

  - Run the following command to disable the automatic partition creation and deletion functions:

    ```
    ALTER TABLE CPU1 RESET (PERIOD,TTL);
    ```

  - To disable only the automatic partition deletion, run the following command:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
ALTER TABLE CPU3 RESET (TTL);
```

– To disable only the automatic partition creation function, run the following command (If the table contains the **ttl** parameter, the operation will fail):

```
ALTER TABLE CPU3 RESET (PERIOD);
```

2.3 Improving Development Efficiency by Leveraging GaussDB(DWS)'s View Decoupling and Rebuilding Function

Base table objects cannot be modified independently due to view and table dependency. To solve this problem, GaussDB(DWS) supports view decoupling and rebuilding. This document describes when and how to use the automatic view rebuilding function.

Scenario

GaussDB(DWS) uses object identifiers (OIDs) to store reference relationships between objects. When a view is defined, the OID of the database object on which the view depends is bound to it. No matter how the view name changes, the dependency does not change. If you modify some columns in the base table, an error will be reported because the columns are strongly bound some objects. If you want to delete a table column or the entire table, you need to use the **cascade** keyword to delete the associated views. After the table column is deleted or the table is re-created, you need to re-create the views of different levels one by one. This increases the workload and deteriorates the usability.

To solve this problem, GaussDB(DWS) 8.1.0 decouples views from their dependent base tables or other database objects (views, synonyms, functions, and table columns), so that these objects can be deleted independently. After the base table is rebuilt, you can run the **ALTER VIEW REBUILD** command to rebuild the dependency. As a development, the version 8.1.1 supports automatic rebuilding. Dependencies can be automatically rebuilt without user awareness. After automatic rebuilding is enabled, lock conflicts may occur. Therefore, you are advised not to enable automatic rebuilding.

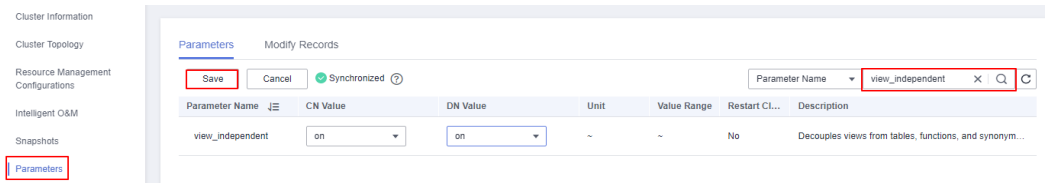
Usage

Step 1 Create a cluster on the management console. For details, see section "Creating a Dedicated GaussDB(DWS) Cluster" in the *Data Warehouse Service (DWS) User Guide*.

Step 2 Enable the GUC parameter **view_independent**.

The GUC parameter **view_independent** controls whether to decouple a view from its objects. This parameter is disabled by default. You need to manually enable the parameter. To enable the **view_independent** parameter, log in to the management console and click the cluster name. On the displayed **Cluster Details** page, click the **Parameters** tab, search for **view_independent**, modify the parameter, and save the modification.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.



Step 3 Use the gsql client to connect to a GaussDB(DWS) database (using the password you defined when creating the cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

If the following information is displayed, the connection succeeded:

```
gaussdb=>
```

Step 4 Create a sample table **t1** and insert data into the table.

```
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');
```

Step 5 Create view **v1** that depends on table **t1**, and create view **v11** that depends on view **v1**. Query view **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)
```

Step 6 After table **t1** is deleted, an error is reported when you query the view **v11**. However, the views still exist.

GaussDB(DWS) provides the **GS_VIEW_INVALID** view to query all invalid views visible to the user. If the base table, function, or synonym that the view depends on is abnormal, the **validtype** column of the view is displayed as "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)
```

Step 7 After the table **t1** is recreated in a cluster of a version earlier than 8.3.0, the view is automatically recreated. The views are automatically refreshed only when they are used.

```
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;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
 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)
```

Step 8 After the table t1 is recreated for a cluster of version 8.3.0 or later, the view is not automatically recreated. The view can be automatically refreshed only after the ALTER VIEW REBUILD operation is performed.

```
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
--------+------------+----------+-----------+------------------------------+-----------
 213563 | public     | v1       | dbadmin   | SELECT a, b FROM public.t1; | invalid
 213567 | public     | v11      | dbadmin   | SELECT a FROM public.v1;    | invalid
(1 row)

ALTER VIEW ONLY v1 REBUILD;

SELECT * FROM gs_view_invalid;
  oid   | schemaname | viewname | viewowner |        definition         | validtype
--------+------------+----------+-----------+---------------------------+-----------
 213567 | public     | v11      | dbadmin   | SELECT a FROM public.v1; | invalid
(1 rows)
```

----End

2.4 Best Practices of GIN Index

A GIN index is a data structure that pairs a key with its posting list. The key indicates a specific value, and the posting list tracks all the locations that this key occurs. For example, 'hello', '14:2 23:4' indicates that **hello** is found at the locations **14:2** and **23:4**. A GIN index efficiently locates tuples with specific keywords, making it ideal for searching elements within multi-valued fields. This section describes how to use GIN indexes to search through array and JSONB types, as well as how to conduct full-text searches.

Using a GIN Index to Search Through the Array Type

Create a GIN index to speed up tag searches.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 1 Create a cluster on the management console. For details, see section "Creating a Dedicated GaussDB(DWS) Cluster" in the *Data Warehouse Service (DWS) User Guide*.

Step 2 Use the gsql client to connect to a GaussDB(DWS) database (using the password you defined when creating the cluster).

```
gsql -d gaussdb -p 8000 -h 192.168.0.86 -U dbadmin -W password -r
```

If the following information is displayed, the connection succeeded:

```
gaussdb=>
```

Step 3 Create the **books** table. The **tags** column stores the tag information of **books** using the array type.

```
CREATE TABLE books (id SERIAL PRIMARY KEY, title VARCHAR(100), tags TEXT[]);
```

Step 4 Insert data.

```
INSERT INTO books (title, tags)
VALUES   ('Book 1', ARRAY['fiction', 'adventure']),
  ('Book 2', ARRAY['science', 'fiction']),
  ('Book 3', ARRAY['romance', 'fantasy']),
  ('Book 4', ARRAY['adventure']);
```

Step 5 Create a GIN index.

```
CREATE INDEX idx_books_tags_gin ON books USING GIN (tags);
```

Step 6 Use the GIN index to perform a search query to find books that contain a specific tag in the **tags** column. Search for books containing the tag "fiction":

```
SELECT * FROM books WHERE tags @> ARRAY['fiction'];
 id | title  |        tags
----+--------+---------------------
  1 | Book 1 | {fiction,adventure}
  2 | Book 2 | {science,fiction}
(2 rows)
```

Step 7 Use the GIN index to search for books that contain both the "fiction" and "adventure" tags.

```
SELECT * FROM books WHERE tags @> ARRAY['fiction', 'adventure'];
 id | title  |        tags
----+--------+---------------------
  1 | Book 1 | {fiction,adventure}
(1 row)
```

----End

Using a GIN Index to Search Through the JSONB Type

When using the JSONB type to store and query JSON data, you can use GIN indexes to improve query performance. GIN indexes are suitable for querying JSONB columns that contain a large number of different key-value pairs.

Step 1 Create the **my_table** table. The **data** column stores information about each person using the JSONB type.

```
CREATE TABLE my_table (id SERIAL PRIMARY KEY, data JSONB);
```

Step 2 Insert data.

```
INSERT INTO my_table (data)
 VALUES ('{"name": "John", "age": 30, "address": {"career": "announcer", "state": "NY"}}'),
     ('{"name": "Alice", "age": 25, "address": {"career": "architect", "state": "CA"}}'),
     ('{"name": "Bob", "age": 35, "address": {"career": "dentist", "state": "WA"}}');
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 3 Create a GIN index to accelerate the query of JSONB columns.

```
CREATE INDEX my_table_data_gin_index ON my_table USING GIN (data);
```

Step 4 Use the GIN index to perform queries on JSONB columns. For example, search for a person whose occupation is dentist::

```
SELECT * FROM my_table WHERE data @> '{"address": {"career": "dentist"}}';
 id |                                 data
----+-----------------------------------------------------------------------------
  3 | {"age": 35, "name": "Bob", "address": {"state": "WA", "career": "dentist"}}
(1 row)
```

Step 5 GIN indexes can also be queried on keys of JSONB columns. For example, search for people who are 30 years old or older:

```
SELECT * FROM my_table WHERE data ->> 'age' >= '30';
 id |                                  data
----+-------------------------------------------------------------------------------
  3 | {"age": 35, "name": "Bob", "address": {"state": "WA", "career": "dentist"}}
  1 | {"age": 30, "name": "John", "address": {"state": "NY", "career": "announcer"}}
(2 rows)
```

----End

Using a GIN Index for Full-Text Search

When using GIN indexes for full-text search, you can use the tsvector and tsquery data types and related functions.

> **NOTE**
>
> To build a tsquery object, you need to use the **to_tsquery** function and provide the search criteria and the corresponding text search configuration (english in this case). Other text search functions and operators can also be used for more complex full-text searches, such as **plainto_tsquery** and **ts_rank**. The specific usage depends on your needs.

Step 1 Create an **articles** table in which the **content** column stores the article content.

```
CREATE TABLE articles (id SERIAL PRIMARY KEY,title VARCHAR(100),content TEXT);
```

Step 2 Insert data.

```
INSERT INTO articles (title, content)
 VALUES ('Article 1', 'This is the content of article 1.'),
   ('Article 2', 'Here is the content for article 2.'),
   ('Article 3', 'This article discusses various topics.'),
   ('Article 4', 'The content of the fourth article is different.');
```

Step 3 Creates an auxiliary column **tsvector** for the **content** column that stores the processed text indexes.

```
ALTER TABLE articles ADD COLUMN content_vector tsvector;
```

Step 4 Update the value in the **content_vector** column and convert the text in the **content** column to the tsvector type.

```
UPDATE articles SET content_vector = to_tsvector('english', content);
```

Step 5 Create a GIN index.

```
CREATE INDEX idx_articles_content_gin ON articles USING GIN (content_vector);
```

Step 6 Perform a full-text search, using the tsquery type to specify the search criteria. For example, search for an article that contains the word "content":

```
SELECT * FROM articles WHERE content_vector @@ to_tsquery('english', 'content');
```

----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2.5 Encrypting and Decrypting Data Columns

Data encryption is widely used in various information systems as a technology to effectively prevent unauthorized access and prevent data leakage. As the core of the information system, the GaussDB(DWS) data warehouse also provides data encryption functions, including transparent encryption and encryption using SQL functions. This section describes SQL function encryption.

NOTE

Currently, GaussDB(DWS) does not support decrypting data encrypted in Oracle, Teradata, and MySQL databases. The encryption and decryption of Oracle, Teradata, and MySQL databases are different from those of GaussDB(DWS). GaussDB(DWS) can only decrypt unencrypted data migrated from Oracle, Teradata, and MySQL databases.

Background

- Hash Functions

  The hash function is also called the digest algorithm. It maps input data of an arbitrary length to an output of fixed length. For example, Hash(data)=result. This process is irreversible. That is, the hash function does not have an inverse function, and data cannot be obtained from the result. In scenarios where plaintext passwords should not be stored (passwords are sensitive) or known by system administrators, hash algorithms should be used to store one-way hash values of passwords.

  In actual use, salt values and iteration are added to prevent same hash values generated by same passwords, hence to prevent rainbow table attacks.

- Symmetric Encryption Algorithms

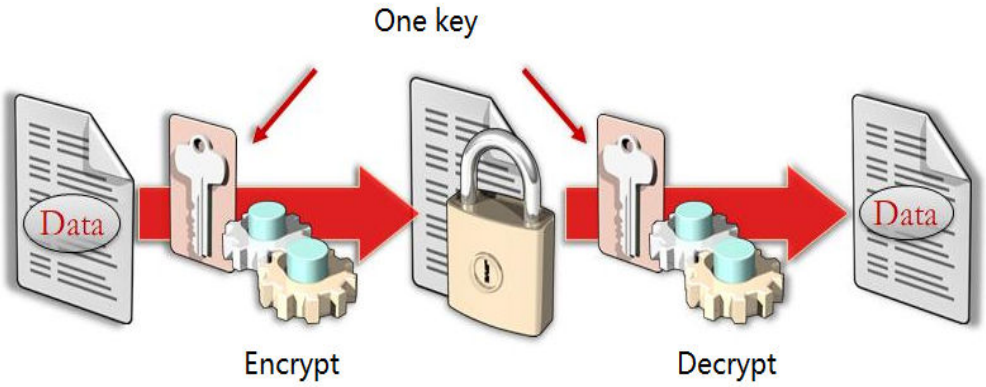
  Symmetric encryption algorithms use the same key to encrypt and decrypt data. There are two subcategories of symmetric encryption algorithms: block ciphers and stream ciphers.

  Block ciphers break the plaintext into fixed-length groups of bits known as blocks and Each block then gets encrypted as a unit. And if there's not enough data to completely fill a block, "padding" is then used to ensure that the blocks meet the fixed-length requirements. Due to padding, the length of the ciphertext obtained by block ciphers is greater than that of the plaintext.

  In stream ciphers, encryption and decryption parties use same pseudo-random encrypted data stream as keys, and plaintext data is sequentially encrypted by these keys. In practice, data is encrypted one bit at a time using an XOR operation. Stream cyphers do not need to be padded. Therefore the length of the obtained ciphertext is same as the length of the plaintext.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 2-3 Symmetric encryption algorithms



Technical Details

GaussDB(DWS) provides hash functions and symmetric cryptographic algorithms to encrypt and decrypt data columns. Hash functions support sha256, sha384, sha512, and SM3. Symmetric cryptographic algorithms support AES128, AES192, AES256, and SM4.

- Hash Functions
  - md5(string)

    Use MD5 to encrypt string and return a hexadecimal value. MD5 is insecure and is not recommended.
  - gs_hash(hashstr, hashmethod)

    Obtains the digest string of a **hashstr** string based on the algorithm specified by **hashmethod**. **hashmethod** can be **sha256**, **sha384**, **sha512**, or **sm3**.
- Symmetric Encryption Algorithms
  - gs_encrypt(encryptstr, keystr, cryptotype, cryptomode, hashmethod)

    Encrypts an **encryptstr** string using the **keystr** key based on the encryption algorithm specified by **cryptotype** and **cryptomode** and the HMAC algorithm specified by **hashmethod**, and returns the encrypted string.
  - gs_decrypt(decryptstr, keystr, cryptotype, cryptomode, hashmethod)

    Decrypts a **decryptstr** string using the **keystr** key based on the encryption algorithm specified by **cryptotype** and **cryptomode** and the HMAC algorithm specified by **hashmethod**, and returns the decrypted string. The **keystr** used for decryption must be consistent with that used for encryption.
  - gs_encrypt_aes128(encryptstr, keystr)

    Encrypts **encryptstr** strings using **keystr** as the key and returns encrypted strings. The length of **keystr** ranges from 1 to 16 bytes.
  - gs_decrypt_aes128(decryptstr, keystr)

    Decrypts a **decryptstr** string using the **keystr** key and returns the decrypted string. The **keystr** used for decryption must be consistent with that used for encryption. **keystr** cannot be empty.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

For details, see "Database Security Management > Sensitive Data Management > Using Functions for Encryption and Decryption" in *Data Warehouse Service (DWS) Developer Guide*.

Examples

Step 1 Connect to the database.

For details, see "Connecting to a GaussDB(DWS) Cluster"> "Using the CLI to Connect to a GaussDB(DWS) Cluster" in the *Data Warehouse Service (DWS) User Guide*.

Step 2 Create the table **student** with the columns **id**, **name**, and **score**. Then use hash functions to encrypt and save names, and use symmetric cryptographic algorithms to save scores.

```
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'));
```

Step 3 Query the table **student** without using keys. The query result shows that the encrypted data in the name and score columns cannot be viewed even if you have the **SELECT** permission.

```
SELECT * FROM student;
 id |                              name                              |
score                                                |
                          subject
----+----------------------------------------------------------------
+----------------------------------------------------------------------------------------------------------------------------------
+-----------
--------------------------------------------------------------------------------
  1 | 2bd806c97f0e00af1a1fc3328fa763a9269723c8db8fac4f93af71db186d6e90 | AAAAAAAAAABAuUC3VQ
+MvPCDAaTUySl1e2gGLr4/ATdCUjTEvova3cb/Ba3ZKqIn1yNVGEFBvJnTq/3sLF4//
Gm8qG7AyfNbbqdW3aYErLVpbE/QWFX9Ig== | aFEWQR2gkj
iu6sfsAad+dHzfFDHePZ6xd44zyekh+qVFlh9FODZ0DoaFAJXctwUsiqaiitTxW8cCSEaNjS/E7Ke1ruY=
  2 | 81b637d8fcd2c6da6359e6963113a1170de795e4b725b84d1e0b4cfd9ec58ce9 | AAAAAAAAAABAuUC3VQ
+MvPCDAaTUySl1taXxAoDqE793hgyCJvC0ESdAX5Mtgdq2LXI1f5ZxraQ73WIJVtIBX8oe3gTDxoXGlHbHht4kzM
4U8dOwr5rjgg== | aFEWQR2gkj
iu6sfsAad+dM8tPTDo/Pds6ZmqdmjGiKxf39+Wzx5NoQ6c8FrzihnRzgc0fycWSu5YGWNOKYWhRsE84Ac=
  3 | 026ad9b14a7453b7488daa0c6acbc258b1506f52c441c7c465474c1a564394ff |
AAAAAAAAAACnyusORPeApqMUgh56ucQu3uso/
Llw5MbPFMkOXuspEzhhnc9vErwOFe6cuGtx8muEyHCX7V5yXs+8FxhNh3n5L3419LDWJJLY2O4merHpSg== |
zomphRfHV4
H32hTtgkio1PyrobVO8N+hN7kAKwtygKP2E7Aaf1vsjmtLHcL88jyeJNe1lxe0fAvodzPJAxAuV3UJN4M=
(3 rows)
```

Step 4 Query the table **student** using keys. The query result shows that the data is decrypted by the function **gs_decrypt** (corresponding to **gs_encrypt**) and can be viewed.

```
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)
```

----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

3 Database Management

3.1 Role-based Access Control (RBAC)

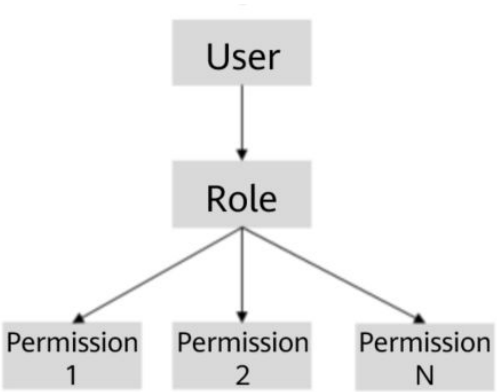
What is RBAC?

- Role-based access control (RBAC) is to grant permissions to roles and let users obtain permissions by associating with roles.
- A role is a set of permissions.
- RBAC greatly simplifies permissions management.

What is the RBAC Model?

Assign appropriate permissions to roles.

Associate users with the roles.



Scenarios

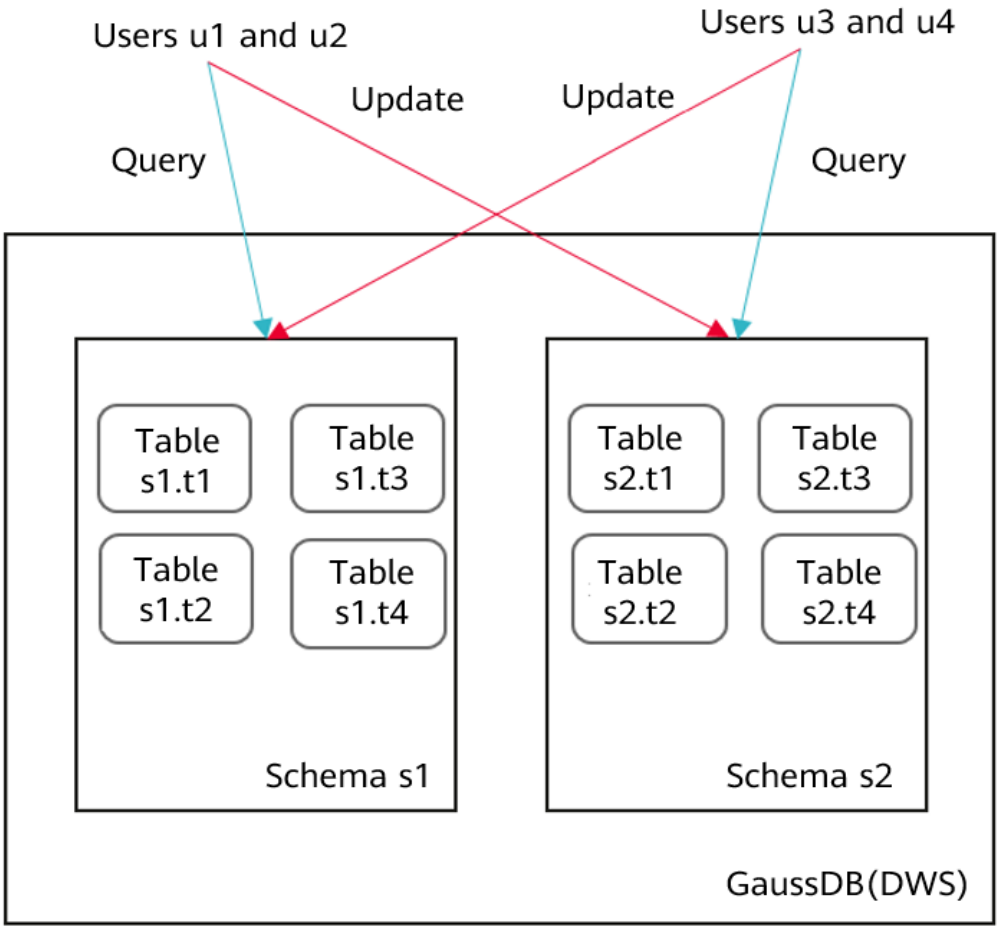
Assume there are two schemas, **s1** and **s2**.

There are two groups of users:

- Users **u1** and **u2** can query all the tables in **s1** and update all the tables in **s2**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Users **u3** and **u4** can query all the tables in **s2** and update all the tables in **s1**.



Granting Permissions

Step 1 Connect to the GaussDB(DWS) database as user **dbadmin**.

Step 2 Run the following statements to create schemas **s1** and **s2** and users **u1** to **u4**:

NOTE

Replace *{password}* with the actual password.

```
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}';
```

Step 3 Copy and run the following statements to create the **s1.t1** and **s2.t1** tables:

```
CREATE TABLE s1.t1 (c1 int, c2 int);
CREATE TABLE s2.t1 (c1 int, c2 int);
```

Step 4 Run the following statement to insert data to the tables:

```
INSERT INTO s1.t1 VALUES (1,2);
INSERT INTO s2.t1 VALUES (1,2);
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 5 Run the following statements to create four roles, each having the query or update permission of table **s1** or **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
```

Step 6 Run the following statements to grant the access permissions of schemas **s1** and **s2** to the roles:

```
GRANT USAGE ON SCHEMA s1, s2 TO rs1_select, rs1_update,rs2_select, rs2_update;
```

Step 7 Run the following statements to grant specific permissions to the roles:

```
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.
```

Step 8 Run the following statements to grant roles to users:

```
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.
```

Step 9 Run the following statement to view the role bound to a specific user:

```
\du u1;
```

```
test_lhy=> \du u1
                List of roles
 Role name | Attributes |        Member of
-----------+------------+-------------------------
 u1        |            | {rs1_select,rs2_update}
```

Step 10 Start another session. Connect to the database as user **u1**.

```
gsql -d gaussdb -h GaussDB(DWS)_EIP -U u1 -p 8000 -r -W {password};
```

Step 11 Run the following statements in the new session verify that user **u1** can query but cannot update **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
```

Step 12 Run the following statements in the new session to verify that user **u1** can update **s2.t1**:

```
SELECT * FROM s2.t1;
UPDATE s2.t1 SET c2 = 3 WHERE c1 = 1;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
test_lhy=> SELECT * FROM s2.t1;
 c1 | c2
----+----
  1 |  2
(1 row)

test_lhy=> UPDATE s2.t1 SET c2 = 3 WHERE c1 = 1;
UPDATE 1
```

----End

3.2 Configuring Read-Only Permissions

Context

If you need to assign different permissions to employees in your company to access your GaussDB(DWS) resources, you can use roles, policies, and user groups preset on the security management page of the ManageOne unified portal. The preset policies and domains bound to user groups are oriented to all resources. To implement domain-based management, you need to create policies and user groups. With rights- and domain-based management, you can use your cloud account to create users, and assign permissions to the users to control their access to specific resources.

- **Scenario 1**: Allow software developers in your enterprise to use GaussDB(DWS) resources, but do not allow them to delete the resources or perform any high-risk operations. To this end, you can create users for these developers and grant them only the permissions required for using GaussDB(DWS) resources.
- **Scenario 2**: Allow employees to use only GaussDB(DWS) resources, but not the resources of other services. To this end, grant them only the permissions for GaussDB(DWS).

Prerequisites

- You have logged in to the ManageOne unified portal.
- You have the permissions to manage users, user groups, and roles.

Tutorial 1: Read-Only Operations on Resource Space

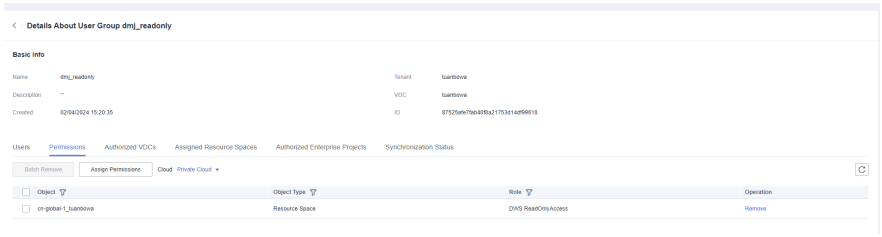
Step 1 Create a user group and assign permissions to it.

1. On the ManageOne unified portal homepage, choose **System** > **Permissions** > **User Groups** and click **Create** to create a user group.
2. Click the name of the created user group. On the details page that is displayed, switch to **Permissions**, and click **Assign Permissions** to grant the read-only permission **DWS ReadOnlyAccess** to GaussDB(DWS).

   NOTE

   Tag Management Service (TMS) in ManageOne is a global service. Tag-related permissions take effect only on all resource spaces.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.



Step 2 Create a user and add it to a user group.

On the ManageOne unified portal homepage, choose **Basic Info** > **Users** and click **Create** to create a user. Add the user to the user group created in **Step 1**.

Step 3 Log in and verify permissions.

Log in to the console by using the user created and verify the user permissions.

- Choose **Service List** > **Data Warehouse Service** to access the GaussDB(DWS) console, and click **Create GaussDB(DWS) Cluster** to create a GaussDB(DWS) cluster. If you cannot create one, **DWS ReadOnlyAccess** has taken effect.
- Choose any other service in **Service List**. If only the **DWS ReadOnlyAccess** policy is added and a message is displayed indicating that you have insufficient permission to access the service, **DWS ReadOnlyAccess** has taken effect.

----End

Tutorial 2: Read-Only Operations in an Enterprise Project

Step 1 Create a user group and assign permissions to it.

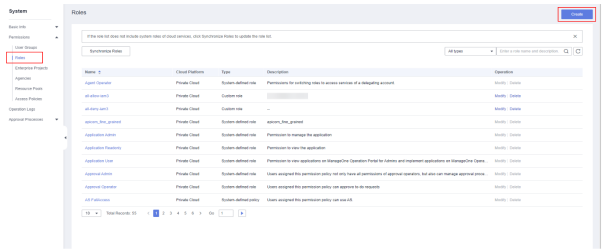
1. On the ManageOne unified portal homepage, choose **System** > **Permissions** > **User Groups** and click **Create** to create a user group.
2. Click the name of the created user group. On the details page that is displayed, switch to **Permissions**, and click **Assign Permissions** to grant the read-only permission **DWS ReadOnlyAccess** to GaussDB(DWS).

NOTE

- In the enterprise project view, the system still displays a message indicating that you lack the fine-grained permissions if you perform read-only operations irrelevant to resources. For example, fine-grained permissions related to events and alarms.
- Tag Management Service (TMS) in ManageOne is a global service. Tag-related permissions take effect only on all resource spaces.

Step 2 Create a role that has read-only permissions for events and alarms in the resource space.

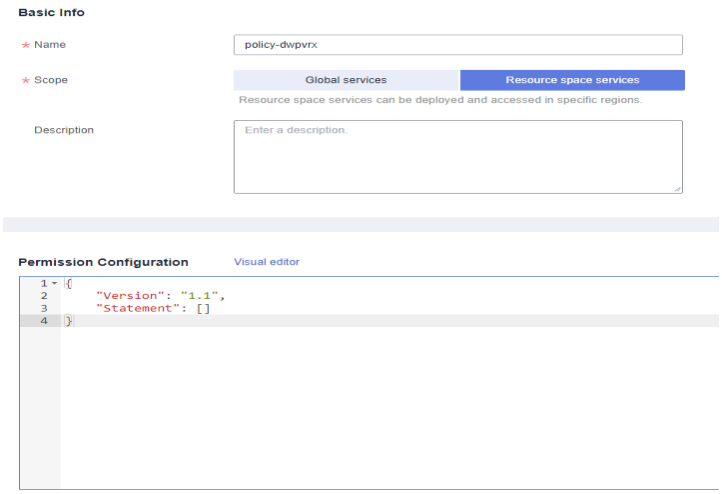
1. Choose **System** > **Permissions** > **Roles** and click **Create**.



 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. Select **Resource space services** for **Scope** and enter the following custom policy **readonly_event_alarm** in **Permission Configuration**:

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "dws:alarm*:list*",
            "dws:cluster*:list*",
            "dws:dms*:get*",
            "dws:event*:list*"
         ]
      }
   ]
}
```



3. Click the name of the created user group. On the details page that is displayed, switch to **Permissions**, and click **Assign Permissions** to grant the created custom policy to configure the read-only permission for the enterprise project.

Step 3 Create a user and add it to a user group.

On the ManageOne unified portal homepage, choose **Basic Info** > **Users** and click **Create** to create a user. Add the user to the user group created in **Step 1**.

Step 4 Log in and verify permissions.

Log in to the console by using the user created and verify the user permissions.

- Choose **Service List** > **Data Warehouse Service** to access the GaussDB(DWS) console, and click **Create GaussDB(DWS) Cluster** to create a GaussDB(DWS) cluster. If you cannot create one, **DWS ReadOnlyAccess** has taken effect.
- Choose any other service in **Service List**. If only the **DWS ReadOnlyAccess** policy is added and a message is displayed indicating that you have insufficient permission to access the service, **DWS ReadOnlyAccess** has taken effect.

----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

3.3 Excellent Practices for SQL Queries

Based on a large number of SQL execution mechanisms and practices, we can optimize SQL statements following certain rules to more quickly execute SQL statements and obtain correct results.

- Replacing **UNION** with **UNION ALL**

  UNION eliminates duplicate rows while merging two result sets but **UNION ALL** merges the two result sets without deduplication. Therefore, replace **UNION** with **UNION ALL** if you are sure that the two result sets do not contain duplicate rows based on the service logic.

- **Adding NOT NULL to the join column**

  If there are many **NULL** values in the **JOIN** columns, you can add the filter criterion **IS NOT NULL** to filter data in advance to improve the **JOIN** efficiency.

- Converting **NOT IN** to **NOT EXISTS**

  nestloop anti join must be used to implement **NOT IN**, and **Hash anti join** is required for **NOT EXISTS**. If no **NULL** value exists in the **JOIN** column, **NOT IN** is equivalent to **NOT EXISTS**. Therefore, if you are sure that no **NULL** value exists, you can convert **NOT IN** to **NOT EXISTS** to generate **hash joins** and to improve the query performance.

  As shown in the following figure, the **t2.d2** column does not contain null values (it is set to **NOT NULL**) and **NOT EXISTS** is used for the query.

  ```
  SELECT * FROM t1 WHERE  NOT EXISTS (SELECT * FROM t2 WHERE t1.c1=t2.d2);
  ```

  The generated execution plan is as follows:

  Figure 3-1 NOT EXISTS execution plan

  ```
   id |                operation
  ----+-----------------------------------------
    1 | ->  Streaming (type: GATHER)
    2 |    ->  Hash Right Anti Join (3, 5)
    3 |       ->  Streaming(type: REDISTRIBUTE)
    4 |          ->  Seq Scan on t2
    5 |       ->  Hash
    6 |          ->  Seq Scan on t1

  Predicate Information (identified by plan id)
  ---------------------------------------------
    2 --Hash Right Anti Join (3, 5)
          Hash Cond: (t2.d2 = t1.c1)
  (13 rows)
  ```

- Use **hashagg**.

  If a plan involving groupAgg and SORT operations generated by the **GROUP BY** statement is poor in performance, you can set **work_mem** to a larger value to generate a **hashagg** plan, which does not require sorting and improves the performance.

- Replace functions with **CASE** statements

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

The GaussDB(DWS) performance greatly deteriorates if a large number of functions are called. In this case, you can modify the pushdown functions to **CASE** statements.

- **Do not use functions or expressions for indexes.**

  Using functions or expressions for indexes stops indexing. Instead, it enables scanning on the full table.

- Do not use **!=** or **<>** operators, **NULL**, **OR**, or implicit parameter conversion in **WHERE** clauses.

- **Split complex SQL statements.**

  You can split an SQL statement into several ones and save the execution result to a temporary table if the SQL statement is too complex to be tuned using the solutions above, including but not limited to the following scenarios:

  – The same subquery is involved in multiple SQL statements of a task and the subquery contains large amounts of data.

  – Incorrect **Plan cost** causes a small hash bucket of subquery. For example, the actual number of rows is 10 million, but only 1000 rows are in hash bucket.

  – Functions such as **substr** and **to_number** cause incorrect measures for subqueries containing large amounts of data.

  – **BROADCAST** subqueries are performed on large tables in multi-DN environment.

For details, see "Typical SQL Optimization Methods" in the *Data Warehouse Service (DWS) Developer Guide*.

3.4 Excellent Practices for Data Skew Queries

3.4.1 Real-Time Detection of Storage Skew During Data Import

During the import, the system collects statistics on the number of rows imported on each DN. After the import is complete, the system calculates the skew ratio. If the skew ratio exceeds the specified threshold, an alarm is generated immediately. The skew ratio is calculated as follows: Skew ratio = (Maximum number of rows imported on a DN – Minimum number of rows imported on a DN)/Number of imported rows. Currently, data can be imported only by running **INSERT** or **COPY**.

NOTE

enable_stream_operator must be set to **on** so that DNs can return the number of imported rows at a time when a plan is delivered to them. Then, the skew ratio is calculated on the CN based on the returned values.

Usage

1. Set parameters table_skewness_warning_threshold (threshold for triggering a table skew alarm) and table_skewness_warning_rows (minimum number of rows for triggering a table skew alarm).

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

  - The value of **table_skewness_warning_threshold** ranges from **0** to **1**. The default value is **1**, indicating that the alarm is disabled. Other values indicate that the alarm is enabled.
  - The value of **table_skewness_warning_rows** ranges from **0** to **2147483647**. The default value is **100,000**. The alarm is triggered only when the following condition is met: Total number of imported rows > Value of **table_skewness_warning_rows** x Number of DNs involving in the import.

```
show table_skewness_warning_threshold;
set table_skewness_warning_threshold = xxx;
show table_skewness_warning_rows;
set table_skewness_warning_rows = xxx;
```

2. Use **INSERT** or **COPY** to import data.
3. Detect and handle alarms. The alarm information includes the table name, minimum number of rows, maximum number of rows, total number of rows, average number of rows, skew rate, and prompt information about data distribution or parameter modification.

```
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
```

3.4.2 Quickly Locating the Tables That Cause Data Skew

Currently, the following skew query APIs are provided: table_distribution(schemaname text, tablename text), table_distribution(), and PGXC_GET_TABLE_SKEWNESS. You can select one based on service requirements.

Scenario 1: Data Skew Caused by a Full Disk

First, use the pg_stat_get_last_data_changed_time(oid) function to query the tables whose data is changed recently. The last change time of a table is recorded only on the CN where **INSERT**, **UPDATE**, and **DELETE** operations are performed. Therefore, you need to query tables that are changed within the last day (the period can be changed in the function).

```
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;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Then, execute the table_distribution(schemaname text, tablename text) function to query the storage space occupied by the tables on each DN.

```
SELECT table_distribution(schemaname,relname) FROM get_last_changed_table();
```

Scenario 2: Routine Data Skew Inspection

- If the number of tables in the database is less than 10,000, use the **PGXC_GET_TABLE_SKEWNESS** view to query data skew of all tables in the database.
  ```
  SELECT * FROM pgxc_get_table_skewness ORDER BY totalsize DESC;
  ```
- If the number of tables in the database is no less than 10,000, you are advised to use the **table_distribution()** function instead of the PGXC_GET_TABLE_SKEWNESS view because the view takes a longer time (hours) due to the query of the entire database for skew columns. When you use the **table_distribution()** function, you can define the output based on PGXC_GET_TABLE_SKEWNESS, optimizing the calculation and reducing the output columns. For example:
  ```
  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;
  ```

Scenario 3: Querying Data Skew of a Table

Run the following SQL statement to query the data skew of a table. Replace **table_name** with the actual table name.

```
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;
```

The following is an example of the information returned. If the data distribution deviation on each DN is less than 10%, data is evenly distributed. If it is greater than 10%, data skew occurs.

```
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)
```

3.5 Best Practices for User Management

A GaussDB(DWS) cluster mainly consists of system administrators and common users. This section describes the permissions of system administrators and common users and describes how to create users and query user information.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

System Administrator

The user **dbadmin** created when you start a GaussDB(DWS) cluster is a system administrator. It has the highest system permission and can perform all operations, including operations on tablespaces, tables, indexes, schemas, functions, and custom views, as well as query for system catalogs and views.

To create a database administrator, connect to the database as an administrator and run the **CREATE USER** or **ALTER USER** statement with **SYSADMIN** specified.

Examples:

Create user **Jim** as a system administrator.

```
CREATE USER Jim WITH SYSADMIN password '{Password}';
```

Change user **Tom** to a system administrator. **ALTER USER** can be used only for existing users.

```
ALTER USER Tom SYSADMIN;
```

Common User

You can run the **CREATE USER** SQL statement to create a common user. A common user cannot create, modify, delete, or assign tablespaces, and needs to be assigned the permission for accessing tablespaces. A common user has all permissions for its own tables, schemas, functions, and custom views, creates indexes on its own tables, and queries only some system catalogs and views.

The database cluster has one or more named databases. Users are shared within the entire cluster, but their data is not shared.

Common user operations are as follows. Replace **password** with the actual password.

1. Creating a user
   ```
   CREATE USER Tom PASSWORD '{Password}';
   ```
2. Changing a user password

   Change the login password of user **Tom** from **password** to **newpassword**.
   ```
   ALTER USER Tom IDENTIFIED BY 'newpassword' REPLACE '{Password}';
   ```
3. Assigning permissions to a user
   - Add **CREATEDB** when you create a user that has the permission for creating a database.
     ```
     CREATE USER Tom CREATEDB PASSWORD '{Password}';
     ```
   - Add the **CREATEROLE** permission for a user.
     ```
     ALTER USER Tom CREATEROLE;
     ```
4. Revoking user permissions
   ```
   REVOKE ALL PRIVILEGES FROM Tom;
   ```
5. Locking or unlocking a user
   - Lock user **Tom**.
     ```
     ALTER USER Tom ACCOUNT LOCK;
     ```
   - Unlock user **Tom**.
     ```
     ALTER USER Tom ACCOUNT UNLOCK;
     ```
6. Deleting a user

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
DROP USER Tom CASCADE;
```

User Information Query

System views related to users, roles, and permissions include **ALL_USERS**, **PG_USER**, and **PG_ROLES**, and system catalogs include **PG_AUTHID** and **PG_AUTH_MEMBERS**.

- **ALL_USERS** displays all users in the database but does not show the details of them.
- **PG_USER** displays user information, including user IDs, the permission to create databases, and resource pools.
- **PG_ROLES** displays information about database roles.
- **PG_AUTHID** records information about database authentication identifiers (roles), including role permissions to log in or create databases.
- **PG_AUTH_MEMBERS** stores information of roles contained in a role group.

1. You can run **PG_USER** to query all users in the database. User ID (**USESYSID**) and permissions can also be queried.

```
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** displays all users in the database but does not show the details of them.

```
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** stores information about roles that have accessed the database.

```
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
---------+----------+------------+---------------+-------------+--------------+-------------+----------------
+---------------+----------------+--------------+-------------+---------------+-----
----------+--------------+-------------+-------------+-----------+-------+----------+---------+-----------
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
+--------------+---------------
 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. To view user properties, query the system catalog **PG_AUTHID**, which stores information about database authorization identifiers (roles). Each cluster, not each database, has only one **PG_AUTHID** system catalog. Only users with system administrator permissions can access the catalog.

```
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 |
sha256366f1e665be208e6015bc3c5795d13e4dc297a148dca6c60346018c80e5c04c9ba170384ce44609b
31baa741f09a3ea5bedc7dadb906286ca994067c3fbf672dc08c981929e326ca08c005d8df942994e146ed
3302af47000b36e9852b50e39dmd585de11aafebd90ec620b201fc36f07a5ecdficefade3a1456ec0aca9a0
ee01e3bf2971d1dbafd604e596149e2e2928be4060dec2bd8688776588b4cd8c64fd38f1b0beab1603129f
a396556ba8aa4c7d6e137a04623 |              |              | default_pool | t        |          0 |           |
n       |           0 |            |              |             |           |
 sysadmin | f       | t        | f          | f         | f          | t         | f           | f          | t
|          -1 |
sha256ecaa7f0ca4436143af43074f16cdd825783ad1a5d659fd94f5e2fa5124e7da44045ecf40bda1a9797
5fcf5920dca0c8be375be5c71b51cb1eeeba0851fb3648cfa49f55989f83fd9baf1a9d5853ce19125f4fc29a7
c709c095ed02d00638410dmd556d6e2dcc41594dc7ad8ee909ef81637ecdficefadefd7d9704ee06affef958
1cd6a50a546607f88891198e96a5e84e7e83dccf56c5cd20a500bbc5248e8ea51f0bca70c5a8dcf00953f8b
62c7a181368153abce760 |              |              | default_pool | f        |          0 |           | n
|           |            |           |          |
 Tom     | f       | t        | f          | t         | f          | t         | f           | f          | f
|          -1 |
sha256f43c4f52ac51e297bc4dbdbc751fcf05319c15681dbf5a9c5777d2edce45cb592a948b25457a728e9
9a3e0608592f33b0a4312eba6124936522304ba298caa2002a04578860fecb0286d7c7baec09365eafd049
b2b99f74f21a08864dd7d3f2amd515ee49f0b18ef8e7d0cd27d91ce2fa9decdficefade16bab5f05b6d7c86a
19ae6406cc59c437506c3f6187bfdf3eefc7a7c7033afa076361b255cc8b6ccb6e19d4767effaec654b3308cc
72cebb891d00a4a10362da |              |              | default_pool | f        |          0 |           | n
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
|           |           |           |           |
(3 rows)
```

User Resource Query

1. Querying the resource quota and usage of all users

   ```
   SELECT * FROM PG_TOTAL_USER_RESOURCE_INFO;
   ```

   Example of the resource usage of all users:

   ```
    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. Querying the resource quota and usage of a specified user

   ```
   SELECT * FROM GS_WLM_USER_RESOURCE_INFO('username');
   ```

   Example of the resource usage of user **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. Querying the I/O usage of a specified user

   ```
   SELECT * FROM pg_user_iostat('username');
   ```

   Example of the I/O usage of user **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)
   ```

3.6 Viewing Table and Database Information

Querying Table Information

- Querying information about all tables in a database using the **pg_tables** system catalog

  ```
  SELECT * FROM pg_tables;
  ```

- Querying the table structure using **\d+** command of the **gsql** tool.

  Example: Create a table **customer_t1** and insert data into the table.

  ```
  CREATE TABLE customer_t1
  (
      c_customer_sk             integer,
      c_customer_id             char(5),
      c_first_name              char(6),
      c_last_name               char(8)
  ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
)
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');
```

Query the table structure. If no schema is specified when you create a table, the schema of the table defaults to **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
```

NOTE

The options may vary in different versions but the difference does not affect services. The options here are for reference only. The actual options are subject to the version.

- Use **pg_get_tabledef** to query the table definition.

```
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)
```

- Querying all data in **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)
```

- Querying all data of a column in **customer_t1** using **SELECT**

```
SELECT c_customer_sk FROM customer_t1;
 c_customer_sk
---------------
          6885
          4321
          9527
(3 rows)
```

- Check whether a table has been analyzed. The time when the table was analyzed will be returned. If nothing is returned, it indicates that the table has not been analyzed.

```
SELECT pg_stat_get_last_analyze_time(oid),relname FROM pg_class where relkind='r';
```

Query the time when the **public** table was analyzed.

```
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';
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
 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)
```

- Quickly query the column information of a table. If a view in **information_schema** has a large number of objects in the database, it takes a long time to return the result. You can run the following SQL statement to quickly query the column information of one or more tables:

```
SELECT /*+ set (enable_hashjoin off) */T.table_schema AS tableschema,
    T.TABLE_NAME AS tablename,
    T.dtd_identifier AS srcAttrId,
    COLUMN_NAME AS fieldName,
    'N' AS isPrimaryKey,
    nvl ( nvl ( T.character_maximum_length, T.numeric_precision ), 0 ) AS fieldLength,
    T.udt_name AS fieldType
from (
 SELECT  /*+ indexscan(co) indexscan(nco) indexscan(a) indexscan(t) leading((nc c a)) leading((co
nco)) indexscan(bt) indexscan(nt) */
    nc.nspname AS table_schema,
    c.relname AS table_name,
    a.attname AS column_name,
    information_schema._pg_char_max_length(information_schema._pg_truetypid(a.*, t.*),
information_schema._pg_truetypmod(a.*, t.*))::information_schema.cardinal_number AS
character_maximum_length,
    information_schema._pg_numeric_precision(information_schema._pg_truetypid(a.*, t.*),
information_schema._pg_truetypmod(a.*, t.*))::information_schema.cardinal_number AS
numeric_precision,
    COALESCE(bt.typname, t.typname)::information_schema.sql_identifier AS udt_name,
    a.attnum AS dtd_identifier
   FROM pg_attribute a
   LEFT JOIN pg_attrdef ad ON a.attrelid = ad.adrelid AND a.attnum = ad.adnum
   JOIN (pg_class c
   JOIN pg_namespace nc ON c.relnamespace = nc.oid) ON a.attrelid = c.oid
   JOIN (pg_type t
   JOIN pg_namespace nt ON t.typnamespace = nt.oid) ON a.atttypid = t.oid
   LEFT JOIN (pg_type bt
   JOIN pg_namespace nbt ON bt.typnamespace = nbt.oid) ON t.typtype = 'd'::"char" AND
t.typbasetype = bt.oid
   LEFT JOIN (pg_collation co
   JOIN pg_namespace nco ON co.collnamespace = nco.oid) ON a.attcollation = co.oid AND
(nco.nspname <> 'pg_catalog'::name OR co.collname <> 'default'::name)
  WHERE NOT pg_is_other_temp_schema(nc.oid) AND a.attnum > 0 AND NOT a.attisdropped AND
(c.relkind = ANY (ARRAY['r'::"char", 'v'::"char", 'f'::"char"])) AND (pg_has_role(c.relowner,
'USAGE'::text) OR has_column_privilege(c.oid, a.attnum, 'SELECT, INSERT, UPDATE, REFERENCES'::text))

) t
WHERE
    1 = 1
    AND UPPER ( T.TABLE_NAME ) <> 'DIS_USER_DATARIGHT_IF_SPLIT_T'
    AND UPPER ( T.TABLE_NAME ) NOT LIKE'DIS_TMP_%'
    AND UPPER ( T.COLUMN_NAME ) <> '_DISAPP_AUTO_ID_'
    AND (  ( T.TABLE_NAME ),  ( T.table_schema ) ) IN ( ( lower ( 'table_name' )::name, lower
( 'schema_name' )::name ) );
```

  Quickly query the column information of the **customer_t1** table.

```
SELECT /*+ set (enable_hashjoin off) */T.table_schema AS tableschema,
    T.TABLE_NAME AS tablename,
    T.dtd_identifier AS srcAttrId,
    COLUMN_NAME AS fieldName,
    'N' AS isPrimaryKey,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    nvl ( nvl ( T.character_maximum_length, T.numeric_precision ), 0 ) AS fieldLength,
    T.udt_name AS fieldType
from (
 SELECT  /*+ indexscan(co) indexscan(nco) indexscan(a) indexscan(t) leading((nc c a)) leading((co
nco)) indexscan(bt) indexscan(nt) */
    nc.nspname AS table_schema,
    c.relname AS table_name,
    a.attname AS column_name,
    information_schema._pg_char_max_length(information_schema._pg_truetypid(a.*, t.*),
information_schema._pg_truetypmod(a.*, t.*))::information_schema.cardinal_number AS
character_maximum_length,
    information_schema._pg_numeric_precision(information_schema._pg_truetypid(a.*, t.*),
information_schema._pg_truetypmod(a.*, t.*))::information_schema.cardinal_number AS
numeric_precision,
    COALESCE(bt.typname, t.typname)::information_schema.sql_identifier AS udt_name,
    a.attnum AS dtd_identifier
  FROM pg_attribute a
  LEFT JOIN pg_attrdef ad ON a.attrelid = ad.adrelid AND a.attnum = ad.adnum
  JOIN (pg_class c
  JOIN pg_namespace nc ON c.relnamespace = nc.oid) ON a.attrelid = c.oid
  JOIN (pg_type t
  JOIN pg_namespace nt ON t.typnamespace = nt.oid) ON a.atttypid = t.oid
  LEFT JOIN (pg_type bt
  JOIN pg_namespace nbt ON bt.typnamespace = nbt.oid) ON t.typtype = 'd'::"char" AND
t.typbasetype = bt.oid
  LEFT JOIN (pg_collation co
  JOIN pg_namespace nco ON co.collnamespace = nco.oid) ON a.attcollation = co.oid AND
(nco.nspname <> 'pg_catalog'::name OR co.collname <> 'default'::name)
  WHERE NOT pg_is_other_temp_schema(nc.oid) AND a.attnum > 0 AND NOT a.attisdropped AND
(c.relkind = ANY (ARRAY['r'::"char", 'v'::"char", 'f'::"char"])) AND (pg_has_role(c.relowner,
'USAGE'::text) OR has_column_privilege(c.oid, a.attnum, 'SELECT, INSERT, UPDATE, REFERENCES'::text))

) t
WHERE
    1 = 1
    AND UPPER ( T.TABLE_NAME ) <> 'DIS_USER_DATARIGHT_IF_SPLIT_T'
    AND UPPER ( T.TABLE_NAME ) NOT LIKE'DIS_TMP_%'
    AND UPPER ( T.COLUMN_NAME ) <> '_DISAPP_AUTO_ID_'
    AND (  ( T.TABLE_NAME ),  ( T.table_schema ) ) IN ( ( lower ( 'promotion' )::name, lower
( 'public' )::name ) );
```

- Obtain the table definition by querying audit logs.

  Use the **pgxc_query_audit** function to query audit logs of all CNs. The syntax is as follows:

  ```
  pgxc_query_audit(timestamptz startime,timestamptz endtime)
  ```

  Query the audit records of multiple objects.

  ```
  SET audit_object_name_format TO 'all';
  SELECT object_name,result,operation_type,command_text FROM pgxc_query_audit('2024-05-26
  8:00:00','2024-05-26 22:55:00') where command_text like '%student%';
  ```

Querying the Table Size

- Querying the total size of a table (indexes and data included)

  ```
  SELECT pg_size_pretty(pg_total_relation_size('<schemaname>.<tablename>'));
  ```

  Example:

  First, create an index on **customer_t1**.

  ```
  CREATE INDEX index1 ON customer_t1 USING btree(c_customer_sk);
  ```

  Then, query the size of table **customer_t1** of **public**.

  ```
  SELECT pg_size_pretty(pg_total_relation_size('public.customer_t1'));
   pg_size_pretty
  ----------------
   264 kB
  (1 row)
  ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Querying the size of a table (indexes excluded)

```
SELECT pg_size_pretty(pg_relation_size('<schemaname>.<tablename>'));
```

Example: Query the size of table **customer_t1** of **public**.

```
SELECT pg_size_pretty(pg_relation_size('public.customer_t1'));
 pg_size_pretty
----------------
 208 kB
(1 row)
```

- Query all the tables, ranked by their occupied space.

```
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;
```

Example 1: Query the 15 tables that occupy the most space.

```
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)
```

Example 2: Query the top 20 tables with the largest space usage in the **public** schema.

```
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)
```

Quickly Querying the Space Occupied by All Tables in the Database

In a large cluster (8.1.3 or later) with a large amount of data (more than 1000 tables), you are advised to use the **pgxc_wlm_table_distribution_skewness** view to query all tables in the database. This view can be used to query the tablespace usage and data skew in the database. The unit of **total_size** and **avg_size** is byte.

```
SELECT *, pg_size_pretty(total_size) as tableSize FROM pgxc_wlm_table_distribution_skewness ORDER BY
total_size desc;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
 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
```

The query result shows that the **history_tbs_test_row_1** table occupies the largest space and data skew occurs.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

CAUTION

1. The **pgxc_wlm_table_distribution_skewness** view can be queried only when the GUC parameter **use_workload_manager and enable_perm_space** is enabled. In earlier versions, you are advised to use the **table_distribution()** function to query the entire database. If only the size of a table is queried, the **table_distribution(schemaname text, tablename text)** function is recommended.
2. In 8.2.1 and later cluster versions, GaussDB(DWS) supports the **pgxc_wlm_table_distribution_skewness** view, which can be directly used for query.
3. In the 8.1.3 cluster version, you can use the following definition to create a view and then perform query:

```
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;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Querying Database Information

- Querying the database list using the **\l** meta-command of the **gsql** tool.

```
\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)
```

  NOTE

  - If the parameters **LC_COLLATE** and **LC_CTYPE** are not specified during database installation, the default values of them are **C**.
  - If **LC_COLLATE** and **LC_CTYPE** are not specified during database creation, the sorting order and character classification of the template database are used by default.

    For details, see "CREATE DATABASE" in the *Data Warehouse Service (DWS) SQL Syntax Reference*.

- Querying the database list using the **pg_database** system catalog

```
SELECT datname FROM pg_database;
  datname
-----------
 template1
 template0
 gaussdb
(3 rows)
```

Querying the Database Size

Querying the size of databases

```
select datname,pg_size_pretty(pg_database_size(datname)) from pg_database;
```

Example:

```
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)
```

Querying the Size of a Table and the Size of the Corresponding Index in a Specified Schema

```
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
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
   ( 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;
```

3.7 Best Practices of Dynamic Load Management

This practice demonstrates how GaussDB(DWS) organizes workloads through memory resource management.

Dynamic load management has advantages over static load management because it supports memory control, in addition to concurrency control and CPU isolation. Memory management in dynamic load management has two core capabilities:

- Memory can be preempted based on the estimated memory usage of statements.
- One of the CN nodes serves as the CCN node. The CCN node manages the memory resources of the entire cluster.

During the estimation of statement memory usage, a range is provided first. The maximum value indicates the memory required for optimal statement running performance. The minimum value indicates the memory required for statement running when data spilling is allowed. The final estimation will be within this range.

Dynamic Load Management

Step 1 Query the total available memory resources of the cluster in the **PGXC_TOTAL_MEMORY_DETAIL** view. As shown in the query result, the maximum dynamic memory of all DNs is the total available memory resources of the cluster.

```
SELECT * FROM pgxc_total_memory_detail WHERE nodename like 'dn_%' and memorytype =
'max_dynamic_memory';
   nodename   |     memorytype     | memorymbytes
--------------+--------------------+--------------
 dn_6019_6020 | max_dynamic_memory |        24614
 dn_6003_6004 | max_dynamic_memory |        24614
 dn_6011_6012 | max_dynamic_memory |        24614
 dn_6027_6028 | max_dynamic_memory |        24614
 dn_6009_6010 | max_dynamic_memory |        24614
 dn_6013_6014 | max_dynamic_memory |        24614
 dn_6021_6022 | max_dynamic_memory |        24614
 dn_6025_6026 | max_dynamic_memory |        24614
 dn_6029_6030 | max_dynamic_memory |        24614
 dn_6005_6006 | max_dynamic_memory |        24614
 dn_6007_6008 | max_dynamic_memory |        24614
 dn_6015_6016 | max_dynamic_memory |        24614
 dn_6001_6002 | max_dynamic_memory |        24614
 dn_6023_6024 | max_dynamic_memory |        24614
 dn_6017_6018 | max_dynamic_memory |        24614
(15 rows)
```

Step 2 Classify workloads and create resource pools for them.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

- It is recommended that you limit the number of service categories to less than 5. When there are too many categories, the memory resources in each resource pool are reduced as they are already occupied.
- When creating a resource pool, you can leave the management and control policies unconfigured, such as the concurrency, memory, and CPU.
- The default concurrency of a resource pool is 10. You can set **active_statements** to **-1** to cancel concurrency control.

Run the **CREATE RESOURCE POOL** statement to create a resource pool.

```
CREATE RESOURCE POOL rp1 WITH (active_statements = -1);
CREATE RESOURCE POOL

SELECT * FROM pg_resource_pool;
 respool_name | mem_percent | cpu_affinity |    control_group    | active_statements | max_dop |
memory_limit | parentid | io_limits | io_priority |  nodegroup   | is_foreign | short_acc | except_rule
--------------+-------------+--------------+---------------------+-------------------+---------+--------------+----------
+-----------+-------------+--------------+------------+-----------+-------------
 default_pool |           0 |           -1 | DefaultClass:Medium |                -1 |      -1 | default      |        0 |         0
| None        | installation | f          | t          | None
 rp1          |           0 |           -1 | DefaultClass:Medium |                -1 |      -1 | default      |        0 |         0 |
None        | installation | f          | t          | None
(2 rows)
```

Associate workload users with the resource pool.

```
ALTER USER user1 WITH RESOURCE POOL 'rp1';
ALTER USER

SELECT * FROM pg_user;
  username  | usesysid | usecreatedb | usesuper | usecatupd | userepl |  passwd  | valbegin | valuntil |
respool    | parent | spacelimit | useconfig | nodegroup | tempspacelimit | spillspacelimit
------------+----------+-------------+----------+-----------+---------+----------+----------+----------+--------------
+--------+------------+-----------+-----------+-----------------+-----------------
 x00482400 |       10 | t           | t        | t         | t       | ******** |          |          | default_pool |      0 |
|          |          |               |
 user1     |    17362 | f           | f        | f         | f       | ******** |          |          | rp1          |      0 |
|          |          |               |
(2 rows)
```

Step 3 Allocate memory to the resource pool by setting **mem_percent**. In GaussDB(DWS) 8.1.2 and later versions, top SQL statement monitoring does not affect the statement execution performance. You can configure the following GUC parameters to enable top SQL statement monitoring:

```
enable_resource_track = on
resource_track_level = query
resource_track_cost = 0
resource_track_duration = 0
```

After top SQL statement monitoring is enabled, run normal workloads for a period of time to verify that the SQL monitoring information can be recorded in the TopSQL history tables. Run the following query statement to check the recorded SQL information:

```
select substr(query,0,100) as query,count(query) as count from dbms_om.gs_wlm_session_info group by
query order by count desc limit 100;
                        query                         | count
------------------------------------------------------+-------
 SELECT o.o_w_id, o.o_d_id, o.o_id, o.o_ol_cnt, COU |    19
 SELECT * FROM (SELECT o.o_w_id, o.o_d_id, o.sum_o_ |    19
 SELECT i_price, i_name, i_data FROM item WHERE i_i |    16
 SELECT o.xmin,  o.o_w_id, o.o_d_id, o.o_id, o.o_ol |    15
 explain performance select cd_gender, cd_marital_s |    14
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
SELECT i_price, i_name, i_data FROM item WHERE i_i |    14
SELECT i_price, i_name, i_data FROM item WHERE i_i |    14
explain performance with ws as (select d_year AS w |    14
explain performance with cross_items as (select i_ |    13
```

You can run the following statements to query the estimated and actual memory of each statement:

- Obtain the estimated average memory usage, estimated maximum memory usage, actual average memory usage, actual maximum memory usage of statements in each resource pool.

```
select resource_pool, avg(estimate_memory) as rp_avg_estimate_memory, max(estimate_memory) as
rp_max_estimate_memory, avg(average_peak_memory) as rp_avg_average_peak_memory,
max(average_peak_memory) as rp_max_average_peak_memory from dbms_om.gs_wlm_session_info
group  by  resource_pool;
 resource_pool | rp_avg_estimate_memory | rp_max_estimate_memory | rp_avg_average_peak_memory |
rp_max_average_peak_memory
---------------+------------------------+------------------------+----------------------------+----------------------------
 default_pool  |                     23 |                    111 |                         15 |                        170
 rp1           |                   25.6 |                   8186 |                       24.9 |                       9150
 root          |                    .67 |                      2 |                          4 |                          4
(3 rows)
```

- Obtain the top x% estimated memory usages and actual memory usages of statements in each resource pool.

```
select resource_pool,percentile_cont(0.99) within group(order by estimate_memory) as
rp_percentile_estimate_memory,percentile_cont(0.99) within group(order by average_peak_memory) as
rp_percentile_average_peak_memory from dbms_om.gs_wlm_session_info  group  by  resource_pool;
 resource_pool | rp_percentile_estimate_memory | rp_percentile_average_peak_memory
---------------+-------------------------------+-----------------------------------
 root          |                          1.84 |                                 4
 rp1           |                           298 |                               198
 default_pool  |                            70 |                               121
(3 rows)
```

NOTE

percentile_cont is set to **0.99**, indicating that the top 99% estimated memory usages and top 99% actual memory usages are returned. You can adjust the value of **percentile_cont** based on the number of statements in the resource pool. This parameter is used to check the estimated and actual memory usage distribution in the resource pool. The memory allocated to each resource pool is determined (using the parameter **mem_percent**) based on this distribution. After memory is allocated to resource pools, the available memory for statements reduces from the remaining system memory to the remaining memory of each resource pool. The performance of some statements will thus deteriorate. The purpose of memory management is to protect important statements from the impact of random or poor statements and to ensure overall performance, not to totally avoid performance deterioration.

When allocating memory to a resource pool, comply with the following principles to guarantee overall statement performance:

1. Ensure that the memory upper limit of each resource pool is larger than the total estimated memory usage of most statements in the resource pool, and larger than twice of the total of top x% estimated memory usage.
2. If the above principle cannot be met, there may be too many resource pools created. As a result, the memory allocated to each resource pool is insufficient. In this case, you need to combine resource pools.
3. If the estimated memory usage of most statements in a resource pool is less than 256 MB, you do not need to set **mem_percent** to reserve memory for workloads in the resource pool. Statements in the resource pool can directly use idle system resources. However, there may be statements with large

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

estimated memory usage in the resource pool in the future, which may impact the entire system. To avoid this, you are advised to set **memory_limit** for the resource pool to ensure that the estimated memory of all statements in the resource pool will not exceed the value of **memory_limit**.

4. Use **mem_percent** to allocate memory resources to each resource pool based on principle 1. If the system memory parameter **max_dynamic_memory** indicates that there is idle memory, you are advised to allocate the idle memory to resource pools based on the workload priority, impact scope, workload concurrency, and number of statements whose resource pool memory upper limit is lower than its estimated memory usage, handling high priority workloads first or minimizing performance impact first.

Step 4 Configure **memory_limit** to put an upper limit on the estimated memory usage of statements in a resource pool. Query TopSQL information and check the maximum actual memory usage and top x% actual memory usages of statements in a resource pool.

```
select resource_pool,max(average_peak_memory) as rp_max_average_peak_memory,percentile_cont(0.995)
within group(order by average_peak_memory) as rp_percentile_average_peak_memory from
dbms_om.gs_wlm_session_info  group  by resource_pool;
 resource_pool | rp_max_average_peak_memory | rp_percentile_average_peak_memory
---------------+----------------------------+-----------------------------------
 root          |                          4 |                                 4
 rp1           |                       9150 |                              1658
 default_pool  |                        170 |                               135
(3 rows)
```

During estimation of statement memory, a range is provided first. The minimum estimated memory is determined based on the statement execution plan and cannot be manually adjusted. The maximum estimated memory is the minimum value among the available memory of the system, available memory of the resource pool, and estimated memory upper limit of the resource pool. The available memory of a resource pool is controlled by the **mem_percent** parameter. After the **mem_percent** value is determined, the value of **memory_limit** can be set using the following principles:

1. The default estimated memory upper limit of statements in a resource is about half of the available memory of the resource pool, which is controlled by the parameter **mem_percent**. Generally, this value does not need to be modified, and can only be decreased. To increase it, ensure all the related settings comply with principle 1 for setting the parameter **mem_percent**.
2. Try to set **memory_limit** to a value that can meet the actual memory requirements of all statements in the resource pool. The parameter **memory_limit** is used to prevent statements with high memory usage from exhausting memory in the resource pool. If the number of concurrent requests in the resource pool is not limited, you are advised to set **memory_limit** to 20% of the upper limit of the resource pool memory.
3. Pay attention to the execution duration of statements with high estimated memory usage. If statements with long execution duration and large estimated memory usage are executed concurrently, the memory in the resource pool may be used up. You can decrease the value of **memory_limit** to a value smaller than the estimated memory usage of such statements to reduce their preempted memory, ensuring the performance of other statements.
4. Ensure **mem_percent** is set for every resource pool, or some statements may occupy too many system resources.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 5 Properly set the parameter **agg_max_mem**, which indicates the estimated memory of aggregation operators. Query the TopSQL information to check the estimated memory usage and actual memory usage of **GROUP BY** statements.

```
select substr(query,0,100) as query, count(query) as count, floor(avg(estimate_memory)) as
avg_estimate_memory, floor(avg(max_peak_memory)) as avg_max_peak_memory,
round((avg_estimate_memory+1) / (avg_max_peak_memory+1), 2) as factor from
dbms_om.gs_wlm_session_info where query like '%group by%' group by query  order by
avg_estimate_memory  desc  limit 100;
                                                query                                                | count | avg_estimate_memory |
avg_max_peak_memory |  factor
-----------------------------------------------------------------------------------------------------+-------
+---------------------+---------------------+----------
 insert into test1.rpt_wf_tmp2\r                                                                     |     1 |               60680
|               38800 |      1.56
 FETCH FORWARD 1000 FROM __QUERY5555__(CURSOR __DS1__ NO SCROLL FOR sel
|     1 |               46710 |               1392 |    33.53
 SELECT tt.* FROM (  select /*+parallel(8)*/                                                         |     2 |               29606
|                16 |  1741.59
 FETCH FORWARD 1000 FROM __QUERY5555__(CURSOR __DS1__ NO SCROLL FOR sel
|     1 |               29605 |                 16 |  1741.53
 FETCH FORWARD 1000 FROM __QUERY5556__(CURSOR __DS1__ NO SCROLL FOR sel
|     1 |               29605 |               1536 |    19.26
 insert into test1.w_f_tmp(version_id,l3_dept_code,rpt_scp_comp_en_name,biz_scr_co                   |     1
|               28684 |                656 |    43.66
 insert into test1.w_f_tmp(version_id,sub_account_code,je_category_en_name,subject                   |     1
|               28684 |                656 |    43.66
 insert into test1.w_f_tmp(version_id,common_prod_flag,company_rel_geo_pc_code,pro                   |     1
|               28684 |                656 |    43.66
 insert /* -I- */ into test1.w_f_tmp(version_id,common_prod_fl                                      |     1 |
28675 |                640 |    44.74
 insert /* -I- */ into test1.w_f_tmp(version_id,common_prod_fl                                      |     1 |
27789 |                624 |    44.46
```

If the estimated memory of the **GROUP BY** statements is large (that is, the values of **avg_estimate_memory** and **factor** are both large), you can set **agg_max_mem** to **2 GB** or **4 GB** (depending on the value of **max_dynamic_memory**) to limit the memory estimation of the **Agg** operators for **GROUP BY** statements that combine more than five columns, to prevent the estimation of the entire statement from being too large.

Step 6 Set the maximum available memory for statements at the system level through the parameter **query_max_mem**. Query the TopSQL views and check the maximum actual memory used by a statement on all DNs.

```
select resource_pool,max(max_peak_memory) as rp_max_max_peak_memory from
dbms_om.gs_wlm_session_info  group  by  resource_pool;
 resource_pool | rp_max_max_peak_memory
---------------+------------------------
 default_pool  |                    282
 rp1           |                   9482
 root          |                      4
(2 rows)
```

After the **mem_percent** and **memory_limit** parameters are set, the estimated memory and actual memory of statements can be controlled within a proper range. The memory management of statements is at operator granularity. GaussDB(DWS) can ensure that the memory of heavy-memory operators, such as **AGGREGATION**, **JOIN**, **MATERIALIZE**, **SCAN**, **SET OPERATION**, **SORT**, and **WINDOW**, is controlled. Memory management is not implemented for some non-heavy-memory operators, because they usually do not occupy a large amount of memory. Any statement that only has non-heavy-memory operators but occupies huge memory can be considered as an abnormal statement. Such a statement can be rewritten to reduce memory usage. You can set the parameter **query_max_mem** to prevent this type of abnormal statements. Exercise caution

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

when setting this parameter, because you will need to handle errors caused by statements whose actual memory usage exceeds its value. To check and handle high memory usage errors, you are advised to set **query_max_mem** to 40% of **max_dynamic_memory**.

Step 7 Another thing to be considered is the statements whose estimated memory is 0. Generally, this is because the tables queried by the statements have no statistics. If the estimated memory of a statement is 0, the memory of each operator in the statement is controlled by the parameter **work_mem**. If there are a large number of operators in the statement and **work_mem** is set to a large value, the statement will also occupy large memory. You are advised to set **work_mem** to a small value and to use TopSQL to check statements whose estimated memory is 0 but with a large actual memory usage.

Step 8 In GaussDB(DWS) 8.2.0 and later versions, a negative memory feedback mechanism is introduced with the adjustable parameter **wlm_memory_feedback_adjust**. Memory is preempted based on the estimated statement memory usage calculated on the CN. If the estimated memory usage of a statement is too high, it will preempt too many memory resources, causing subsequent jobs to be queued. With the negative memory feedback mechanism, if the cluster memory usage has been overestimated for a period of time, the CCN node will dynamically release some memory for subsequent jobs, improving resource utilization.

In this mechanism, two parameters can be set:

- Minimum time for triggering negative feedback: the time period that the cluster memory is overestimated before negative feedback is enabled. For example, if this parameter is set to **50**, the negative feedback mechanism is triggered if the memory usage has been overestimated for 50 seconds.
- Minimum estimated memory usage for triggering negative feedback: the minimum estimated memory usage that triggers negative feedback after it lasts for a period of time. For example, if this parameter is set to **40**, the negative feedback mechanism is triggered if the minimum estimated memory usage exceeds 40% of the available system memory.

----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

4 Performance Tuning

4.1 Optimizing Table Structure Design to Enhance GaussDB(DWS) Query Performance

4.1.1 Table Structure Design

Before you optimize a table, you need to understand the structure of the table. During database design, some key factors about table design will greatly affect the subsequent query performance of the database. Table design affects data storage as well. Scientific table design reduces I/O operations and minimizes memory usage, improving the query performance.

This section describes how to optimize table performance in GaussDB(DWS) by properly designing the table structure (for example, by configuring the table storage mode, compression level, distribution mode, distribution column, partitioned tables, and local clustering).

Selecting a Storage Mode

Selecting a model for table storage is the first step of table definition. Select a proper storage model for your service based on the table below.

Generally, if a table contains many columns (called a wide table) and its query involves only a few columns, column storage is recommended. If a table contains only a few columns and a query involves most of the columns, row storage is recommended.

Storage Model	Application Scenario
Row storage	Point query (simple index–based query that returns only a few records). Query involving many **INSERT**, **UPDATE**, and **DELETE** operations.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Storage Model	Application Scenario
Column storage	Statistical analysis queries. Queries with many groups and joins.

The row/column storage of a table is specified by the **orientation** attribute in the table definition. The value **row** indicates a row-store table and **column** indicates a column-store table. The default value is **row**.

Table Compression

Table compression can be enabled when a table is created. Table compression enables data in the table to be stored in compressed format to reduce memory usage.

In scenarios where I/O is large (much data is read and written) and CPU is sufficient (little data is computed), select a high compression ratio. In scenarios where I/O is small and CPU is insufficient, select a low compression ratio. Based on this principle, you are advised to select different compression ratios and test and compare the results to select the optimal compression ratio as required. Specify a compressions ratio using the **COMPRESSION** parameter. The supported values are as follows:

- The valid value of column-store tables is **YES**, **NO**, **LOW**, **MIDDLE**, or **HIGH**, and the default value is **LOW**.
- The valid values of row-store tables are **YES** and **NO**, and the default is **NO**. (The row-store table compression function is not put into commercial use. To use this function, contact technical support.)

The service scenarios applicable to each compression level are described in the following table.

Compression Level	Application Scenario
LOW	The system CPU usage is high and the disk storage space is sufficient.
MIDDLE	The system CPU usage is moderate and the disk storage space is insufficient.
HIGH	The system CPU usage is low and the disk storage space is insufficient.

Selecting a Distribution Mode

GaussDB(DWS) supports the following distribution modes: replication, hash, and Round-robin.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

Round-robin is supported in cluster 8.1.2 and later.

Policy	Description	Application Scenario	Advantages/ disadvantages
Replication	Full data in a table is stored on each DN in the cluster.	Small tables and dimension tables	• The advantage of replication is that each DN has full data of the table. During the join operation, data does not need to be redistributed, reducing network overheads and reducing plan segments (each plan segment starts a corresponding thread). • The disadvantage of replication is that each DN retains the complete data of the table, resulting in data redundancy. Generally, replication is only used for small dimension tables.
Hash	Table data is distributed on all DNs in the cluster.	Fact tables containing a large amount of data	• The I/O resources of each node can be used during data read/write, greatly improving the read/write speed of a table. • Generally, a large table (containing over 1 million records) is defined as a hash table.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Policy	Description	Application Scenario	Advantages/ disadvantages
Polling (Round-robin)	Each row in the table is sent to each DN in turn. Data can be evenly distributed on each DN.	Fact tables that contain a large amount of data and cannot find a proper distribution key in hash mode	• Round-robin can avoid data skew, improving the space utilization of the cluster. • Round-robin does not support local DN optimization like a hash table does, and the query performance of Round-robin is usually lower than that of a hash table. • If a proper distribution key can be found for a large table, use the hash distribution mode with better performance. Otherwise, define the table as a round-robin table.

Selecting a Distribution Key

If the hash distribution mode is used, a distribution key must be specified for the user table. If a record is inserted, the system performs hash computing based on values in the distribute column and then stores data on the related DN.

Select a hash distribution key based on the following principles:

1. **The values of the distribution key should be discrete so that data can be evenly distributed on each DN.** You can select the primary key of the table as the distribution key. For example, for a person information table, choose the ID number column as the distribution key.
2. **Do not select the column where a constant filter exists.** For example, if a constant constraint (for example, zqdh= '000001') exists on the **zqdh** column in some queries on the **dwcjk** table, you are not advised to use **zqdh** as the distribution key.
3. **With the above principles met, you can select join conditions as distribution keys**, so that join tasks can be pushed down to DNs for execution, reducing the amount of data transferred between the DNs.

   For a hash table, an improper distribution key may cause data skew or poor I/O performance on certain DNs. Therefore, you need to check the table to ensure that data is evenly distributed on each DN. You can run the following SQL statements to check for data skew:

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

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

xc_node_id corresponds to a DN. Generally, **over 5% difference between the amount of data on different DNs is regarded as data skew. If the difference is over 10%, choose another distribution key.**

4. You are not advised to add a column as a distribution key, especially add a new column and use the SEQUENCE value to fill the column. (Sequences may cause performance bottlenecks and unnecessary maintenance costs.)

Using Partitioned Tables

Partitioning refers to splitting what is logically one large table into smaller physical pieces based on specific schemes. The table based on the logic is called a partitioned table, and a physical piece is called a partition. Data is stored on these smaller physical pieces, namely, partitions, instead of the larger logical partitioned table. A partitioned table has the following advantages over an ordinary table:

1. High query performance: The system queries only the concerned partitions rather than the whole table, improving the query efficiency.
2. High availability: If a partition is faulty, data in the other partitions is still available.
3. Easy maintenance: You only need to fix the faulty partition.

The partitioned tables supported by GaussDB(DWS) include range partitioned tables and list partitioned tables. (List partitioned tables are supported only in cluster 8.1.3).

Using Partial Clustering

Partial Cluster Key is the column-based technology. It can minimize or maximize sparse indexes to quickly filter base tables. Partial cluster key can specify multiple columns, but you are advised to specify no more than two columns. Use the following principles to specify columns:

1. The selected columns must be restricted by simple expressions in base tables. Such constraints are usually represented by Col, Op, and Const. Col specifies the column name, Op specifies operators, (including =, >, >=, <=, and <) Const specifies constants.
2. Select columns that are frequently selected (to filter much more undesired data) in simple expressions.
3. List the less frequently selected columns on the top.
4. List the columns of the enumerated type at the top.

Selecting a Data type

You can use data types with the following features to improve efficiency:

1. **Data types that boost execution efficiency**

   Generally, the calculation of integers (including common comparison calculations, such as =, >, <, ≥, ≤, and ≠ and **GROUP BY**) is more efficient than that of strings and floating point numbers. For example, if you need to perform a point query on a column-store table whose **NUMERIC** column is

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

used as a filter criterion, the query will take over 10 seconds. If you change the data type from **NUMERIC** to **INT**, the query takes only about 1.8 seconds.

2. **Selecting data types with a short length**

   Data types with short length reduce both the data file size and the memory used for computing, improving the I/O and computing performance. For example, use **SMALLINT** instead of **INT**, and **INT** instead of **BIGINT**.

3. **Same data type for a join**

   You are advised to use the same data type for a join. To join columns with different data types, the database needs to convert them to the same type, which leads to additional performance overheads.

Using Indexes

- The purpose of creating indexes is to accelerate queries. Therefore, ensure that indexes can be used in some queries. If an index is not used by any query statement, the index is meaningless. Delete such an index.
- Do not create unnecessary secondary indexes. Useful secondary indexes can accelerate query. However, the space occupied by indexes increases with the number of indexes. Each time an index is added, an additional key-value pair needs to be added when a piece of data is inserted. Therefore, the more indexes, the slower the write speed, and the larger the space usage. In addition, too many indexes affect the optimizer running time, and inappropriate indexes mislead the optimizer. Having more indexes does not necessarily lead to better results.
- Create proper indexes based on service characteristics. In principle, indexes need to be created for columns required in a query to improve performance. Indexes can be created in the following scenarios:
  - For columns with high differentiation, indexes can significantly reduce the number of rows after filtering. For example, you are advised to create an index in the ID card number column, but not in the gender column.
  - If there are multiple query conditions, you can select a combination index. Note that the column of the equivalent condition must be placed before the combination index. For example, if your query is **SELECT * FROM t where c1 = 10 and c2 = 100 and c3 > 10;**, create a composite index **Index cidx (c1, c2, c3)** to optimize scanning.
- When an index column is used as a query condition, do not perform calculation, function, or type conversion on the index column. Otherwise, the optimizer cannot use the index.
- Ensure that the index column contains the query column. Do not always run the **SELECT *** statement to query all columns.
- Indexes are not utilized when **!= or NOT IN** are used in query conditions.
- When LIKE is used, if the condition starts with the wildcard %, the index cannot be used.
- If multiple indexes are available for a query condition but you know which index is the optimal one, you are advised to use the optimizer hint to force the optimizer to use the index. This prevents the optimizer from selecting an incorrect index due to inaccurate statistics or other problems.
- When the IN expression is used as the query condition, the number of matched conditions should not be too large. Otherwise, the execution efficiency is low.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

4.1.2 Table Optimization Overview

In this practice, you will learn how to optimize the design of your tables. You will start by creating tables without specifying their storage mode, distribution key, distribution mode, or compression mode. Load test data into these tables and test system performance. Then, follow excellent practices to create the tables again using new storage modes, distribution keys, distribution modes, and compression modes. Load the test data and test performance again. Compare the two test results to find out how table design affects the storage space, and the loading and query performance of the tables.

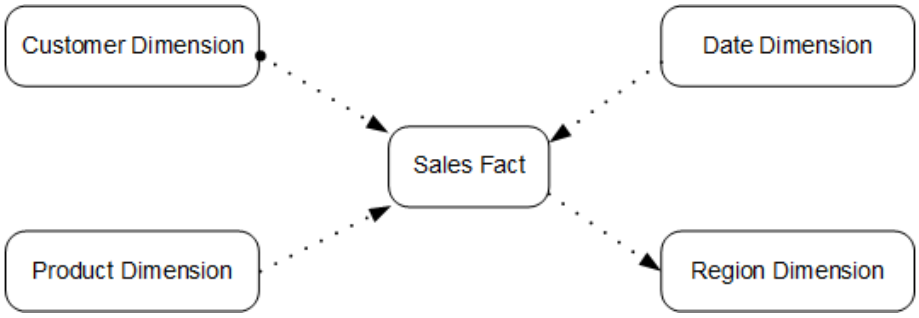
Estimated time: 60 minutes

4.1.3 Selecting a Table Model

The most common types of data warehouse schemas are star and snowflake schemas. Consider service and performance requirements when you choose a schema for your tables.

- In the star schema, a central fact table contains the core data for the database and several dimension tables provide descriptive attribute information for the fact table. The primary key of a dimension table associates a foreign key in a fact table, as shown in **Figure 4-1**.
  - All facts must have the same granularity.
  - Different dimensions are not associated.

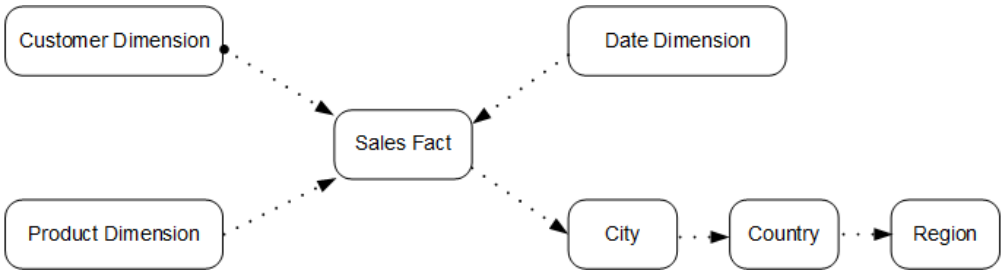
  Figure 4-1 Star schema

- The snowflake schema is developed based on the star schema. In this schema, each dimension can be associated with multiple dimensions and split into tables of different granularities based on the dimension level, as shown in **Figure 4-2**.
  - Dimension tables can be associated as needed, and the data stored in them is reduced.
  - This schema has more dimension tables to maintain than the star schema does.

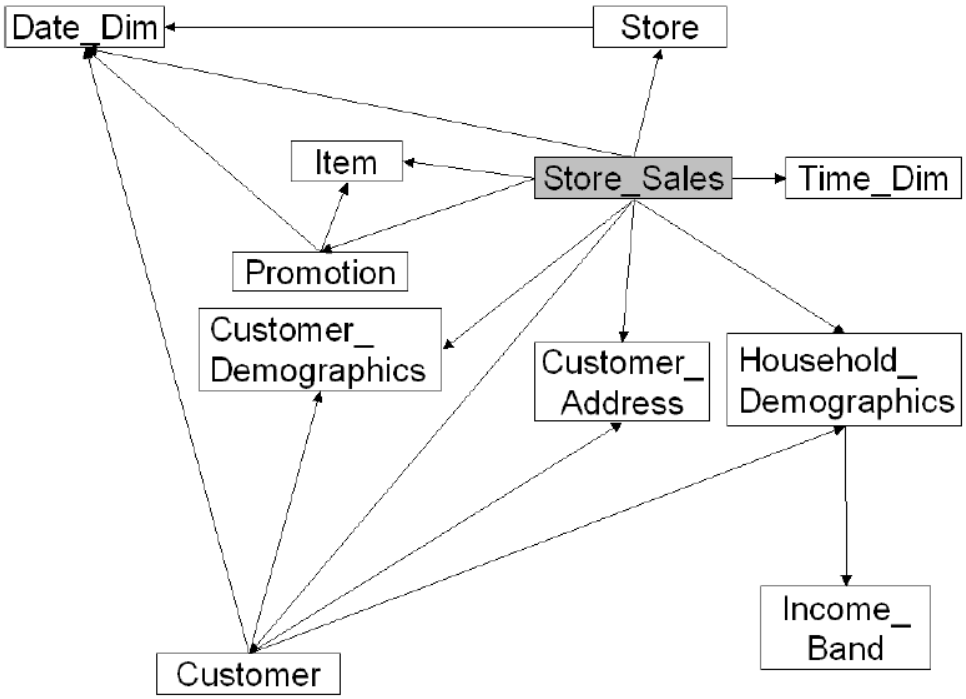
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 4-2 Snowflake schema



This practice verifies performance using the Store Sales (SS) model of TPC-DS. The model uses the snowflake schema. **Figure 4-3** illustrates its structure.

Figure 4-3 TPC-DS Store Sales ER-Diagram



For details about the **store_sales** fact table and dimension tables in the model, see the official document of TPC-DS at **http://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp**.

4.1.4 Step 1: Creating an Initial Table and Loading Sample Data

Create a group of tables without specifying their storage modes, distribution keys, distribution modes, or compression modes. Load sample data into these tables.

Step 1 (Optional) Create a cluster.

If a cluster is available, skip this step. For how to create a cluster, see section "Creating a DWS 2.0 Cluster" in the *Data Warehouse Service (DWS) User Guide*.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Furthermore, connect to the cluster and test the connection. For details, see section "Methods of Connecting to a Cluster" in the *Data Warehouse Service (DWS) User Guide*.

This practice uses an 8-node cluster as an example. You can also use a four-node cluster to perform the test.

Step 2 Create an SS test table **store_sales**.

NOTE

If SS tables already exist in the current database, run the **DROP TABLE** statement to delete these tables first.

For example, delete the **store_sales** table.

```
DROP TABLE store_sales;
```

Do not configure the storage mode, distribution key, distribution mode, or compression mode when you create this table.

Run the **CREATE TABLE** command to create the 11 tables in **Figure 4-3**. This section only provides the syntax for creating the **store_sales** table. To create all tables, copy the syntax in **4.1.10.2 Creating an Initial Table**.

```
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)
) ;
```

Step 3 Load sample data into these tables.

An OBS bucket provides sample data used for this practice. In HUAWEI CLOUD Stack, the sample data is for reference only. You need to upload your own sample data to OBS. The bucket can be read by all authenticated cloud users. Perform the following operations to load the sample data:

1. Create a foreign table for each table.

   GaussDB(DWS) uses the foreign data wrappers (FDWs) provided by PostgreSQL to import data in parallel. To use FDWs, create FDW tables first (also called foreign tables). This section only provides the syntax for creating the **obs_from_store_sales_001** foreign table corresponding to the **store_sales**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

table. To create all foreign tables, copy the syntax in **4.1.10.4 Creating a Foreign Table**.

NOTE

- Note that *<obs_bucket_name>* in the following statement indicates the OBS bucket name. Only some regions are supported. GaussDB(DWS) clusters do not support cross-region access to OBS bucket data.
- The columns of the foreign table must be the same as that of the corresponding ordinary table. In this example, **store_sales** and **obs_from_store_sales_001** should have the same columns.
- The foreign table syntax obtains the sample data used for this practice from the OBS bucket. To load other sample data, modify **SERVER gsmpp_server OPTIONS** as needed. For details, see "About Parallel Data Import from OBS" in the *Data Warehouse Service (DWS) Developer Guide*.
- Hardcoded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store them in the configuration file or environment variables.

```
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.example.com/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. Set **ACCESS_KEY** and **SECRET_ACCESS_KEY** parameters as needed in the foreign table creation statement, and run this statement in a client tool to create a foreign table.

   For the values of **ACCESS_KEY** and **SECRET_ACCESS_KEY**, see "Data Import > Concurrently Importing Data from OBS > Creating Access Keys (AK and SK)"in the *Data Warehouse Service (DWS) Developer Guide*.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

3. Import data.

   Create the **insert.sql** script containing the following statements and execute it:

```
\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
```

   The returned result is as follows:

```
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. Calculate the total time spent in creating the 11 tables. The result will be recorded as the loading time in the benchmark table in **Step 1** in the next section.

5. Run the following command to verify that each table is loaded correctly and records lines into the table:

```
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;
```

   The number of rows in each SS table is as follows:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table name	Number of Rows
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

Step 4 Run the **ANALYZE** command to update statistics.

```
ANALYZE;
```

If **ANALYZE** is returned, the execution is successful.

```
ANALYZE
```

The **ANALYZE** statement collects statistics about table content in databases, which will be stored in the **PG_STATISTIC** system catalog. Then, the query optimizer uses the statistics to work out the most efficient execution plan.

After executing batch insertions and deletions, you are advised to run the **ANALYZE** statement on the table or the entire library to update statistics.

----End

4.1.5 Step 2: Testing System Performance of the Initial Table and Establishing a Baseline

Before and after tuning table structures, test and record the following information to compare differences in system performance:

- Load time
- Storage space occupied by tables
- Query performance

The examples in this practice are based on a dws.d2.xlarge cluster consisting of eight nodes. Because system performance is affected by many factors, clusters of the same flavor may have different results.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 4-1 Cluster specifications

Model	dws.d2.xlarge VM
CPU	4*CPU E5-2680 v2 @ 2.80GHZ
Memory	32 GB
Network	1 GB
Disk	1.63 TB
Number of Nodes	8

Record the results using the following benchmark table.

Table 4-2 Recording results

Benchmark	Before	After
Loading time (11 tables)	341584 ms	-
Occupied storage space		
Store_Sales	-	-
Date_Dim	-	-
Store	-	-
Item	-	-
Time_Dim	-	-
Promotion	-	-
Customer_Demographics	-	-
Customer_Address	-	-
Household_Demographics	-	-
Customer	-	-
Income_Band	-	-
Total storage space	-	-
Query execution time		
Query 1	-	-
Query 2	-	-
Query 3	-	-

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Benchmark	Before	After
Total execution time	-	-

Perform the following steps to test the system performance before tuning to establish a benchmark:

Step 1 Enter the cumulative load time for all the 11 tables in the benchmarks table in the **Before** column.

Step 2 Record the storage space usage of each table.

Determine how much disk space is used for each table using the **pg_size_pretty** function and record the results in base tables.

```
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);
```

The following information is displayed:

```
        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)
```

Step 3 Test query performance.

Run the following queries and record the time spent on each query. The execution durations of the same query can be different, depending on the OS cache during execution. You are advised to perform several rounds of tests and select a group with average values.

```
\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
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
  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;
```

----End

After the preceding statistics are collected, the benchmark table is as follows:

Benchmark	Before	After
Loading time (11 tables)	341584 ms	-
Occupied storage space		
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	-

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Benchmark	Before	After
Customer	441 MB	-
Income_Band	88 KB	-
Total storage space	42 GB	-
Query execution time		
Query 1	14552.05 ms	-
Query 2	27952.36 ms	-
Query 3	17721.15 ms	-
Total execution time	60225.56 ms	-

4.1.6 Step 3: Optimizing a Table

Selecting a Storage Mode

Sample tables used in this practice are typical multi-column TPC-DS tables where many statistical analysis queries are performed. Therefore, the column storage mode is recommended.

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

Selecting a Compression Level

No compression ratio is specified in **4.1.4 Step 1: Creating an Initial Table and Loading Sample Data**, and the low compression ratio is selected by GaussDB(DWS) by default. Specify **COMPRESSION** to **MIDDLE**, and compare the result to that when **COMPRESSION** is set to **LOW**.

The following is an example of selecting a storage mode and the **MIDDLE** compression ratio for a table.

```
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)                  ,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
  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);
```

Selecting a Distribution Mode

Based on table sizes provided in **4.1.5 Step 2: Testing System Performance of the Initial Table and Establishing a Baseline**, set the distribution mode as follows.

Table Name	Number of Rows	Distribution Mode
Store_Sales	287997024	Hash
Date_Dim	73049	Replication
Store	402	Replication
Item	204000	Replication
Time_Dim	86400	Replication
Promotion	1000	Replication
Customer_Demographics	1920800	Hash
Customer_Address	1000000	Hash
Household_Demographics	7200	Replication
Customer	1981703	Hash
Income_Band	20	Replication

Selecting a Distribution Key

If your table is distributed using hash, choose a proper distribution key. You are advised to select a distribution key according to **Selecting a Distribution Key**.

Select the primary key of each table as the distribution key of the hash table.

Table Name	Number of Records	Distribution Mode	Distribution Key
Store_Sales	287997024	Hash	ss_item_sk
Date_Dim	73049	Replication	-
Store	402	Replication	-
Item	204000	Replication	-

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table Name	Number of Records	Distribution Mode	Distribution Key
Time_Dim	86400	Replication	-
Promotion	1000	Replication	-
Customer_Demographics	1920800	Hash	cd_demo_sk
Customer_Address	1000000	Hash	ca_address_sk
Household_Demographics	7200	Replication	-
Customer	1981703	Hash	c_customer_sk
Income_Band	20	Replication	-

4.1.7 Step 4: Creating Another Table and Loading Data

After selecting a storage mode, compression level, distribution mode, and distribution key for each table, use these attributes to create tables and reload data. Compare the system performance before and after the table recreation.

Step 1 Delete the tables created before.

```
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;
```

Step 2 Create tables and specify storage and distribution modes for them.

Only the syntax for recreating the **store_sales** table is provided for simplicity. To recreate all the other tables, copy the syntax in **4.1.10.3 Creating a Another Table After Design Optimization**.

```
CREATE TABLE store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer               not null,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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);
```

Step 3 **Load sample data into these tables.**

Step 4 Record the loading time in the benchmark tables.

Benchmark	Before	After
Loading time (11 tables)	341584 ms	257241 ms
Occupied storage space		
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	-
Total storage space	42 GB	-
Query execution time		
Query 1	14552.05 ms	-
Query 2	27952.36 ms	-

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Benchmark	Before	After
Query 3	17721.15 ms	-
Total execution time	60225.56 ms	-

Step 5 Run the **ANALYZE** command to update statistics.

```
ANALYZE;
```

If **ANALYZE** is returned, the execution is successful.

```
ANALYZE
```

Step 6 Check for data skew.

For a hash table, an improper distribution key may cause data skew or poor I/O performance on certain DNs. Therefore, you need to check the table to ensure that data is evenly distributed on each DN. You can run the following SQL statements to check for data skew:

```
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 corresponds to a DN. Generally, **over 5% difference between the amount of data on different DNs is regarded as data skew. If the difference is over 10%, choose another distribution key.** In GaussDB(DWS), you can select multiple distribution keys to distribute data evenly.

----End

4.1.8 Step 5: Testing System Performance in the New Table

After recreating the test data set with the selected storage modes, compression levels, distribution modes, and distribution keys, you will retest the system performance.

Step 1 Record the storage space usage of each table.

Determine how much disk space is used for each table using the **pg_size_pretty** function and record the results in base tables.

```
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)
```

Step 2 Test the query performance and record the performance data in the benchmark table.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Execute the following queries again and record the time spent on each query.

```
\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;
```

The following benchmark table shows the validation results of the cluster used in this tutorial. Your results may vary based on a number of factors, but the relative results should be similar. The execution durations of queries having the same table structure can be different, depending on the OS cache during execution. You are advised to perform several rounds of tests and select a group with average values.

Benchmark	Before	After
Loading time (11 tables)	341584 ms	257241 ms

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Benchmark	Before	After
Occupied storage space		
Store_Sales	42 GB	14 GB
Date_Dim	11 MB	27 MB
Store	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
Customer	441 MB	111 MB
Income_Band	88 KB	896 KB
Total storage space	42 GB	15 GB
Query execution time		
Query 1	14552.05 ms	1783.353 ms
Query 2	27952.36 ms	14247.803 ms
Query 3	17721.15 ms	11441.659 ms
Total execution time	60225.56 ms	27472.815 ms

Step 3 If you have higher expectations for the performance after the table design, you can run the **EXPLAIN PERFORMANCE** command to view the execution plan for tuning.

For more details about execution plans and query tuning, see "SQL Execution Plan" in the *Data Warehouse Service (DWS) Developer Guide* and "Query Performance Tuning Overview" in the *Data Warehouse Service (DWS) Developer Guide*.

----End

4.1.9 Step 6: Evaluating the Performance of the Optimized Table

Compare the loading time, storage space usage, and query execution time before and after the table tuning.

The following table shows the example results of the cluster used in this tutorial. Your results will be different, but should show similar improvement.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Benchmark	Before	After	Change	Percentage (%)
Loading time (11 tables)	341584 ms	257241 ms	-84343 ms	-24.7%
Occupied storage space			-	-
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%
Total storage space	42 GB	15 GB	-27 GB	-64.3%
Query execution time			-	-
Query 1	14552.05 ms	1783.353 ms	-12768.697 ms	-87.7%
Query 2	27952.36 ms	14247.803 ms	-13704.557 ms	-49.0%
Query 3	17721.15 ms	11441.659 ms	-6279.491 ms	-35.4%
Total execution time	60225.56 ms	27472.815 ms	-32752.745 ms	-54.4%

Evaluating the Table After Optimization

- The loading time was reduced by 24.7%.

  The distribution mode has obvious impact on loading data. The hash distribution mode improves the loading efficiency. The replication distribution mode reduces the loading efficiency. When the CPU and I/O are sufficient, the compression level has little impact on the loading efficiency. Typically, the

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

efficiency of loading a column-store table is higher than that of a row-store table.

- The storage usage space was reduced by 64.3%.

  The compression level, column storage, and hash distribution can save the storage space. A replication table increases the storage usage, but reduces the network overhead. Using the replication mode for small tables is a positive way to use small space for performance.

- The query performance (speed) increased by 54.4%, indicating that the query time decreased by 54.4%.

  The query performance is improved by optimizing storage modes, distribution modes, and distribution keys. In a statistical analysis query on multi-column tables, column storage can improve query performance. In a hash table, I/O resources on each node can be used during I/O read/write, which improves the read/write speed of a table.

  Often, query performance can be improved further by rewriting queries and configuring workload management (WLM).

You can adapt the operations in **4.1 Optimizing Table Structure Design to Enhance GaussDB(DWS) Query Performance** to further improve the distribution of tables and the performance of data loading, storage, and query.

Deleting Resources

After this practice is completed, delete the cluster.

To retain the cluster and delete the SS tables, run the following command:

```
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;
```

4.1.10 Appendix: Table Creation Syntax

4.1.10.1 Usage

This section provides SQL test statements used in this tutorial. You are advised to copy the SQL statements in each section and save them as an .sql file. For example, create a file named **create_table_fir.sql** file and paste the SQL statements in section **4.1.10.2 Creating an Initial Table** to the file. Executing the file on an SQL client tool is efficient, and the total elapsed time of test cases is easy to calculate. Execute the **.sql** file using **gsql** as follows:

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

Replace the italic parts in the example with actual values in GaussDB(DWS). For example:

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Replace the following information in the example based on the site requirements:

- **postgres**: indicates the name of the database to be connected.
- **10.10.0.1**: cluster connection address.
- **dbadmin**: username of the cluster database. The default administrator is **dbadmin**.
- **8000**: database port set during cluster creation.
- **password**: password set during cluster creation.

4.1.10.2 Creating an Initial Table

This section contains the table creation syntax used when you create a table for the first time in this tutorial. Tables are created without specifying their storage modes, distribution keys, distribution modes, or compression modes.

```
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                       ,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
(
    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,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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
) ;
```

4.1.10.3 Creating a Another Table After Design Optimization

This section contains the syntax of creating another table after the storage modes, compression levels, distribution modes, and distribution keys are selected in this practice.

```
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)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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;
```

4.1.10.4 Creating a Foreign Table

This section contains the syntax of foreign tables for obtaining sample data used in this tutorial. The sample data is stored in an OBS bucket accessible to all authenticated cloud users.

NOTE

- Note that *<obs_bucket_name>* in the following statement indicates the OBS bucket name. Only some regions are supported. GaussDB(DWS) clusters do not support cross-region access to OBS bucket data.
- You can replace **ACCESS_KEY** and **SECRET_ACCESS_KEY** with your own credentials in this example.
- When an OBS foreign table is created, only the mapping relationship is created, and data is not pulled to the GaussDB(DWS) disk.

```
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.example.com/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;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
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.example.com/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)                      ,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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.example.com/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.example.com/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                     ,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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.example.com/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.example.com/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
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
OPTIONS (
LOCATION 'obs://obs.example.com/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.example.com/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.example.com/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,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    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.example.com/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.example.com/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;
```

4.2 Analyzing SQL Statements That Are Being Executed to Handle GaussDB(DWS) Performance Issues

During development, developers often encounter problems such as excessive SQL connections, long SQL query time, and SQL query blocking. You can use the **PG_STAT_ACTIVITY** and **PGXC_THREAD_WAIT_STATUS** views to analyze and locate SQL problems. This section describes some common locating methods.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 4-3 Some PG_STAT_ACTIVITY fields

Name	Type	Description
usename	name	Name of the user logging in to the backend
client_addr	inet	IP address of the client connected to the backend **null** indicates either that the client is connected via a Unix socket on the server machine or that this is an internal process such as autovacuum.
application_name	text	Name of the application connected to the backend
state	text	Overall state of the backend. The value can be: • **active**: The backend is executing queries. • **idle**: The backend is waiting for new client commands. • **idle in transaction**: The backend is in a transaction, but there is no statement being executed in the transaction. • **idle in transaction (aborted)**: The backend is in a transaction, but there are statements failed in the transaction. • **fastpath function call**: The backend is executing a **fast-path** function. • **disabled**: This state is reported if **track_activities** is disabled in this backend. **NOTE** Common users can view only the session status of their own accounts. That is, the state information of other accounts is empty.
waiting	boolean	If the back end is currently waiting for a lock, the value is **t**. Otherwise, the value is **f**. • **t** stands for true. • **f** stands for false.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Name	Type	Description
enqueue	text	Queuing status of a statement. Its value can be: • **waiting in global queue**: The statement is queuing in the global concurrency queue. The number of concurrent statements exceeds the value of **max_active_statements** configured for a single CN. • **waiting in respool queue**: The statement is queuing in the resource pool and the concurrency of simple jobs is limited. The main reason is that the concurrency of simple jobs exceeds the upper limit **max_dop** of the fast track. • **waiting in ccn queue**: The job is in the CCN queue, which may be global memory queuing, slow lane memory queuing, or concurrent queuing. The scenarios are: 1. The available global memory exceeds the upper limit, the job is queuing in the global memory queue. 2. Concurrent requests on the slow lane in the resource pool exceed the upper limit, which is specified by **active_statements**. 3. The slow lane memory of the resource pool exceeds the upper limit, that is, the estimated memory of concurrent jobs in the resource pool exceeds the upper limit specified by **mem_percent**. • Empty or **no waiting queue**: The statement is running.
pid	bigint	ID of the backend thread.

Viewing Connection Information

- Set **track_activities** to **on**.

  ```
  SET track_activities = on;
  ```

  The database collects the running information about active queries only if this parameter is set to **on**.

- You can run the following SQL statements to check the current connection user, connection address, connection application, status, whether to wait for a lock, queuing status, and thread ID.

  ```
  SELECT usename,client_addr,application_name,state,waiting,enqueue,pid FROM PG_STAT_ACTIVITY
  WHERE DATNAME='database name';
  ```

  The following command output is displayed:

  ```
  usename |  client_addr  | application_name | state  | waiting | enqueue |       pid
  ---------+---------------+------------------+--------+---------+---------+-----------------
  ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
 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)
```

- End a session (only the system administrator has the permission).

```
SELECT PG_TERMINATE_BACKEND(pid);
```

Viewing SQL Running Information

- Run the following command to obtain all SQL information that the current user has permission to view (if the current user has administrator or preset role permission, all user query information can be displayed):

```
SELECT usename,state,query FROM PG_STAT_ACTIVITY WHERE DATNAME='database name';
```

  If the value of **state** is **active**, the **query** column indicates the SQL statement that is being executed. In other cases, the **query** column indicates the previous query statement. If the value of **state** is **idle**, the connection is idle and waits for the user to enter a command. The following command output is displayed:

```
 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)
```

- Run the following command to view the information about the SQL statements that are not in the idle state:

```
SELECT datname,usename,query FROM PG_STAT_ACTIVITY WHERE state != 'idle' ;
```

Viewing Time-Consuming Statements

- Check the SQL statements that take a long time to execute.

```
SELECT current_timestamp - query_start as runtime, datname, usename, query FROM
PG_STAT_ACTIVITY WHERE state != 'idle' order by 1 desc;
```

  Query statements are returned and sorted by execution time length in descending order. The first record is the query statement that takes the longest time to execute.

```
    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)
```

- Alternatively, you can set **current_timestamp - query_start** to be greater than a threshold to identify query statements that are executed for a duration longer than this threshold.

```
SELECT query from PG_STAT_ACTIVITY WHERE current_timestamp - query_start > interval '2 days';
```

Querying Blocked Statements

- Run the following command to view blocked query statements:

```
SELECT pid, datname, usename, state, query FROM PG_STAT_ACTIVITY WHERE state <> 'idle' and
waiting=true;
```

  Run the following statement to end the blocked SQL session:

```
SELECT PG_TERMINATE_BACKEND(pid);
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

- In most cases, blocking is caused by internal locks and **waiting=true** is displayed. You can view the blocking in the **pg_stat_activity** view.
- The blocked statements about file write and event schedulers cannot be viewed in the **pg_stat_activity** view.

- View information about the blocked query statements, tables, and schemas.

```
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;
```

The command output includes a session ID, user information, query status, and table or schema that caused the block.

After finding the blocked table or schema information, end the faulty session.

```
SELECT PG_TERMINATE_BACKEND(pid);
```

If information similar to the following is displayed, the session is successfully terminated:

```
PG_TERMINATE_BACKEND
----------------------
t
(1 row)
```

If information similar to the following is displayed, the user is attempting to terminate the session, but the session will be reconnected rather than terminated.

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

NOTE

If the **PG_TERMINATE_BACKEND** function is used by the gsql client to terminate the background threads of the session, the client will be reconnected automatically rather than be terminated.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.