Data Warehouse Service (DWS)
8.1.3.333

Quick Start

Issue 01

Date 2024-08-09

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

1 Before You Start

This section describes how to quickly create databases and tables, insert data to tables, and query data in tables. Later sections in this chapter will elaborate on common operations.

Basic Database Operations

Step 1 Create a database user.

By default, only administrators that are generated during cluster creation can access the initial database. They need to create user accounts and grant permissions to let other users access the database.

```
CREATE USER joe WITH PASSWORD 'password';
```

If the following information is displayed, the resource pool is created.

```
CREATE USER
```

The password of user **joe** is user-defined.

When you create a user with the **CREATE USER** command, they are automatically granted default permissions to log in to the database, create tables, views, indexes, and manipulate any objects they create.

Step 2 Create a database.

```
CREATE DATABASE mydatabase;
```

For details about database management, see **2 Creating and Managing Databases**.

Step 3 (Optional) Create a schema.

Schemas allow multiple users to use the same database without interfering with each other.

Create a schema.

```
CREATE SCHEMA myschema;
```

If the following information is displayed, the **myschema** schema is created:

```
CREATE SCHEMA
```


After a schema is created, you can create its objects. Before creating an object, use either of the following methods to create the object in the corresponding schema:

1. Set **search_path** of the database to the schema.
   ```
   SET SEARCH_PATH TO myschema;
   CREATE TABLE mytable (firstcol int);
   ```
2. Specify an object name that contains the schema name. Separate multiple object names by periods (.). The following shows an example.
   ```
   CREATE TABLE myschema.mytable (firstcol int);
   ```

If no schema is specified during object creation, the object will be created in the current schema. Run the following statement to query the current schema:

```
SHOW search_path;
  search_path
----------------
 "$user",public
(1 row)
```

For details about schemas, see **7 Creating and Managing Schemas**.

Step 4 Create a table.

- Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is **integer**.
  ```
  CREATE TABLE mytable (firstcol int);
  ```
  If the **DISTRIBUTE BY** statement is not used to specify the distribution column, the system automatically specifies the first column that meets the criteria as a distribution column. If **CREATE TABLE** is displayed at the end of the returned information, the table has been created.
  ```
  NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'firstcol' as the distribution column by default.
  HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
  ```
  The system catalog **PG_TABLES** contains information about all tables in a cluster. You can run the **SELECT** statement to view the attributes of a table in the system catalog.
  ```
  SELECT * FROM PG_TABLES WHERE TABLENAME = 'mytable';
  ```
- Insert data to the table.
  ```
  INSERT INTO mytable values (100);
  ```
  The **INSERT** statement inserts rows to a database table. For details about batch loading, see "About Parallel Data Import from OBS" in the *Data Warehouse Service (DWS) Developer Guide*.
- View data in the table.
  ```
  SELECT * from mytable;
   firstcol
  ----------
        100
  (1 row)
  ```


NOTE

- For details about how to create a table, see **4.1 Creating a Table**.
- In addition to the created tables, a database contains many system catalogs. These system catalogs contain cluster installation information and information about various queries and processes of GaussDB(DWS). You can collect information about a database by querying system catalogs. For details, see **6 Querying System Catalogs**.
- GaussDB(DWS) supports hybrid row and column storage, which provides high query performance for interaction analysis in complex scenarios. For details about how to select a storage model, see **3 Planning a Storage Model**.

----End

Releasing Resources

If a cluster is deployed for the practice, delete the cluster after the practice is complete.

For details about how to delete clusters, see "Deleting Clusters" in the *Data Warehouse Service (DWS) User Guide*.

Retain the cluster but clear the **mydatabase** database.

```
DROP DATABASE mydatabase;
```

To keep the cluster and the database, run the following statement to delete the tables in the database:

```
DROP TABLE mytable;
```


2 Creating and Managing Databases

Prerequisites

To create a database, you must be a database system administrator or have the permission for creating databases.

Background

- GaussDB(DWS) has two initial template databases **template0** and **template1** and a default user database **gaussdb**.
- **CREATE DATABASE** creates a database by copying a template database (**template1** by default). Do not use a client or any other tools to connect to or to perform operations on the template databases.
- A maximum of 128 databases can be created in GaussDB(DWS).
- A database system consists of multiple databases. A client can connect to only one database at a time. You are not allowed to query data across databases. If a database cluster contains multiple databases, set the **-d** parameter to specify the database to connect to.

Procedure

Step 1 Create database **db_tpcds**.

```
CREATE DATABASE db_tpcds;
```

If the following information is displayed, the database is created.

```
CREATE DATABASE
```

As stated in **Background**, the template database **template1** is copied by default to create a database. Its encoding format is SQL_ASCII. If the name of an object created in this database contains multiple-byte characters (such as Chinese characters) and exceeds the name length limit (63 bytes), the system truncates the name from the last byte instead of the last character. As a result, characters may be incomplete.

To solve the problem, the data object name should not exceed the maximum length or contain multi-byte characters.

If an object whose name is truncated mistakenly cannot be deleted, delete the object using the name before the truncation, or manually delete it from the corresponding system catalog on each node.

You can also use **template0** to create a database by using **CREATE DATABASE** and specify new encoding and locale, for example, use UTF-8 as the default database encoding (**server_encoding**). For details, see the syntax of **CREATE DATABASE**.

You can run the **show server_encoding** command to view the current database encoding.

NOTE

- Database names must comply with the general naming convention of SQL identifiers. The current user automatically becomes the owner of this new database.
- If a database system supports independent users and projects, you are advised to store them in different databases.
- If the projects or users are associated with each other and share resources, store them in different schemas in the same database. A schema is only a logical structure. For details about user permissions for schemas, see table 1 in "Separation of Permissions" in the *Data Warehouse Service (DWS) Developer Guide*.

Step 2 View databases.

- Query the database list using the **\l** meta-command.
  ```
  \l
  ```
- Query the database list in the system catalog **pg_database**:
  ```
  SELECT datname FROM pg_database;
  ```

Step 3 Modify a database.

You can run the **ALTER DATABASE** statement to modify database attributes, such as the owner, name, and default configuration attributes.

- Run the following statement to specify the default schema search path:
  ```
  ALTER DATABASE db_tpcds SET search_path TO pa_catalog,public;
  ```
- Run the following statement to rename the database:
  ```
  ALTER DATABASE db_tpcds RENAME TO human_tpcds;
  ```

Step 4 Delete a database.

You can run the **DROP DATABASE** statement to delete a database. This command deletes the system directory in the database, as well as the database directory on the disk that stores data. Only the database owner or the system administrator can delete a database. A database accessed by users cannot be deleted. You need to connect to another database before deleting this database.

Run the following statement to delete the database:
```
DROP DATABASE human_tpcds;
```

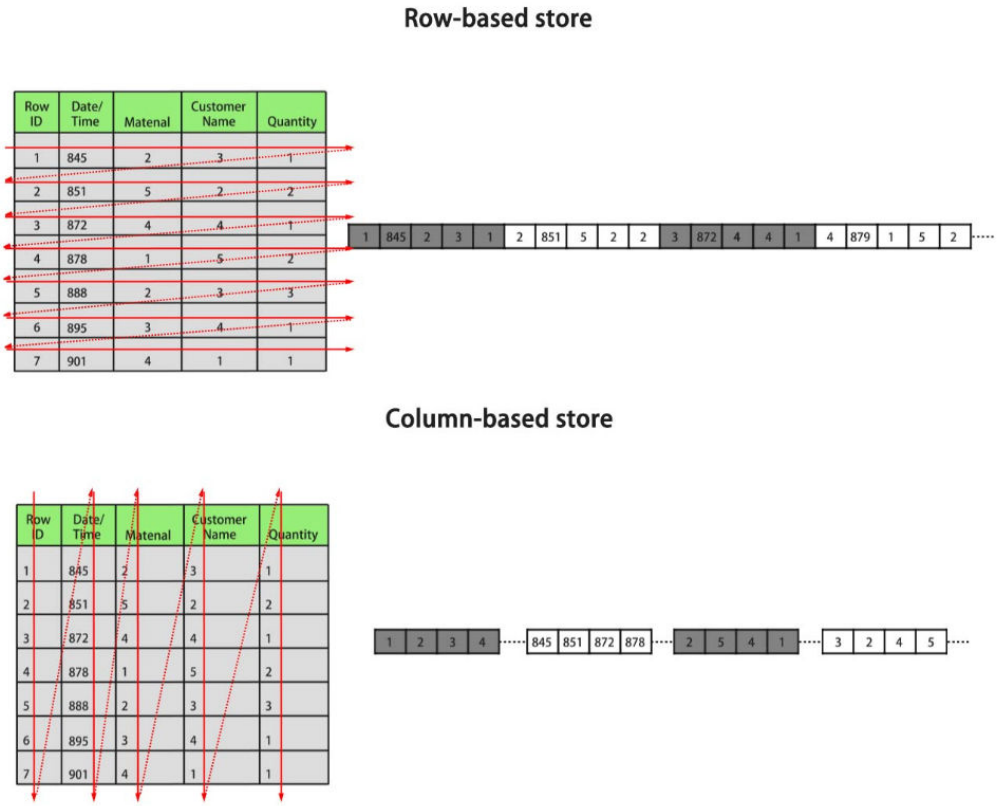
----End

3 Planning a Storage Model

GaussDB(DWS) supports hybrid row and column storage. Each storage mode applies to specific scenarios. Select an appropriate mode when creating a table.

Row storage stores tables to disk partitions by row, and column storage stores tables to disk partitions by column. By default, a table is created in row storage mode. For details about differences between row storage and column storage, see **Figure 3-1**.

Figure 3-1 Differences between row storage and column storage



In the preceding figure, the upper left part is a row-store table, and the upper right part shows how the row-store table is stored on a disk; the lower left part is

a column-store table, and the lower right part shows how the column-store table is stored on a disk.

Both storage modes have benefits and drawbacks.

Storage Mode	Benefit	Drawback
Row storage	All the columns of a record are stored in the same partition. Data can be easily inserted and updated.	All the columns of a record are read after the **SELECT** statement is executed even if only certain columns are required.
Column storage	• Only necessary columns in a query are read. • Projections are efficient. • Any column can serve as an index.	• The selected columns need to be reconstructed after the **SELECT** statement is executed. • Data cannot be easily inserted or updated.

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 includes most of the fields, row storage is recommended.

Storage Mode	Application Scenario
Row storage	• Point queries (simple index-based queries that only return a few records). • Scenarios requiring frequent addition, deletion, and modification.
Column storage	• Statistical analysis queries (requiring a large number of association and grouping operations) • Ad hoc queries (using uncertain query conditions and unable to utilize indexes to scan row-store tables)

Row-Store Table

Row-store tables are created by default. In a row-store table, data is stored by row, that is, data in each row is stored continuously. Therefore, this storage model applies to scenarios where data needs to be updated frequently.

```
CREATE TABLE customer_t1
(
  state_ID    CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID     NUMBER
);

--Delete the table.
DROP TABLE customer_t1;
```


Column-Store Table

In a column-store table, data is stored by column, that is, data in each column is stored continuously. The I/O of data query in a single column is small, and column-store tables occupy less storage space than row-store tables. This storage model applies to scenarios where data is inserted in batches, less updated, and queried for analysis. A column-store table cannot be used for point queries.

```
CREATE TABLE customer_t2
(
  state_ID    CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID     NUMBER
)
WITH (ORIENTATION = COLUMN);

--Delete the table.
DROP TABLE customer_t2;
```


4 Creating and Managing Tables

4.1 Creating a Table

Context

A table is created in a database and can be saved in different databases. Tables under different schemas in a database can have the same name. Before creating a table, perform the operations in **3 Planning a Storage Model**.

Creating a Table

Run the following statement to create a table:

```
CREATE TABLE customer_t1
(
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8)
)
with (orientation = column,compression=middle)
distribute by hash (c_last_name);
```

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

```
CREATE TABLE
```

c_customer_sk, **c_customer_id**, **c_first_name** and **c_last_name** are the column names in the table. *integer*, *char(5)*, *char(6)*, and *char(8)* are column name types.

4.2 Inserting Data to a Table

A new table contains no data. You need to insert data to the table before using it. This section describes how to insert a row or multiple rows of data from a specified table using **INSERT**. If a large amount of data needs to be imported to a table in batches, see "Import Methods" in the *Data Warehouse Service (DWS) Developer Guide*.

Background

The length of a character on the server and client may vary by the used character sets. A string entered on the client will be processed based on the server's character set, so the output may differ from the input.

Table 4-1 Comparison of character set output between the client and server

Procedure	Same Character Sets	Different Character Sets
No operations are performed on the string while it is saved and read.	Your expected result is returned.	If the character sets for input and output on the client are the same, your expected result is returned.
Operations (such as executing string functions) are performed to the string while it is saved and read.	Your expected result is returned.	The result may differ from expected, depending on the operations performed on the string.
A long string is truncated while it is saved.	Your expected result is returned.	If the character sets used on the client and server are different in character length, an unexpected result may occur.

More than one of the preceding operations can be performed on a string. For example, if the character sets of the client and server are different, a string may be processed and then truncated. In this case, the result will also be unexpected. For details, see **Table 4-2**.

NOTE

Long strings are truncated only if **DBCOMPATIBILITY** is set to **TD** (compatible with Teradata) and **td_compatible_truncation** is set to **on**.

Run the following statements to create tables **table1** and **table2** to be used in the examples:

```
CREATE TABLE table1(id int, a char(6), b varchar(6),c varchar(6)) ;
CREATE TABLE table2(id int, a char(20), b varchar(20),c varchar(20)) ;
```


Table 4-2 Examples

ID	Server Character Set	Client Character Set	Automatic Truncation Enabled	Examples	Result	Description
1	SQL_ASCII	UTF8	Yes	**INSERT INTO** *table1* **VALUES**(*1,reverse('123AA78'),reverse('123AA78'),reverse('123AA78')*);	id |a|b|c ----+------+------+------ 1 | 87| 87| 87	A string is reversed on the server and then truncated. Because character sets used by the server and client are different, character A is displayed in multiple bytes on the server and the result is incorrect.
2	SQL_ASCII	UTF8	Yes	**INSERT INTO** *table1* **VALUES**(*2,reverse('123A78'),reverse('123A78'),reverse('123A78')*);	id |a|b|c ----+------+------+------ 2 | 873| 873| 873	A string is reversed and then automatically truncated. Therefore, the result is unexpected.
3	SQL_ASCII	UTF8	Yes	**INSERT INTO** *table1* **VALUES**(*3,'87A123','87A123','87A123'*	id | a | b | c ----+-------+-------+------- 3 | 87A1 | 87A1 | 87A1	The column length in the string type is an integer multiple of the length in client character encoding. Therefore, the result is correct after truncation.

ID	Server Character Set	Client Character Set	Automatic Truncation Enabled	Examples	Result	Description
4	SQL_ASCII	UTF8	No	**INSERT INTO** *table2* **VALUES**(*1,reverse('123AA78'),reverse('123AA78'),reverse('123AA78')*); **INSERT INTO** *table2* **VALUES**(*2,reverse('123A78'),reverse('123A78'),reverse('123A78')*); 	id |a|b|c ---- +---------- --+-------- +-------- 1 | 87 321| 87 321 | 87 321 2 | 87321| 87321| 87321	Similar to the first example, multi-byte characters no longer indicate the original characters after being reversed.

Common Operations

You need to create a table before inserting data to it. For details about how to create a table, see **4 Creating and Managing Tables**.

- Insert a row to table **customer_t1**:

  Data values are arranged in the same order as the columns in the table and are separated by commas (,). Generally, they are text values (constants). Scalar expressions are also allowed.

  ```
  INSERT INTO customer_t1(c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', 'Grace');
  ```

  If you know the sequence of the columns in the table, you can obtain the same result without listing these columns. For example, the following statement generates the same result as the preceding statement:

  ```
  INSERT INTO customer_t1 VALUES (3769, 'hello', 'Grace');
  ```

  If you do not know some of the values, you can omit them. If no value is specified for a column, the column is set to the default value. The following shows an example.

  ```
  INSERT INTO customer_t1 (c_customer_sk, c_first_name) VALUES (3769, 'Grace');

  INSERT INTO customer_t1 VALUES (3769, 'hello');
  ```

  You can also specify the default value of a column or row:

  ```
  INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', DEFAULT);

  INSERT INTO customer_t1 DEFAULT VALUES;
  ```

- To insert multiple rows, run the following statement:

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

  You can also insert multiple rows by running the statement for inserting one row for multiple times. However, you are advised to run this command to improve efficiency.

- Insert data to a table from a specified table. For example, run the following statement to insert the data of table *customer_t1* to the backup table *customer_t2*.

```
CREATE TABLE customer_t2
(
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8)
) ;

INSERT INTO customer_t2 SELECT * FROM customer_t1;
```

  NOTE

  If there is no implicit conversion between the data types of the specified table and those of the current table, the two tables must have the same data types when data is inserted from the specified table to the current table.

- Delete a backup table.

```
DROP TABLE customer_t2 CASCADE;
```

  NOTE

  If the table to be deleted is associated with other tables, delete the associated tables first.

4.3 Updating Data in a Table

Data updating is performed to modify data in a database. You can update one row, all rows, or specified rows of data, or update data in a single column without affecting the data in the other columns.

The following types of information are required when the **UPDATE** statement is used to update a row:

- Table name and column names of the data to be updated
- New column data
- Rows of the data to be updated

NOTE

- You can use a schema as a modifier of the table name. If no such modifier is specified, the table is located based on the default schema.
- In the statement, **SET** is followed by the target column and the new value of the column. The new value can be a constant or an expression.
- The table can contain the **WHERE** clause to filter the data that is equal to the specified condition.
  - If the statement does not include the **WHERE** clause, all rows are updated.
  - If the statement includes the **WHERE** clause, only the rows matching the clause condition are updated.

In the **SET** clause, the equal sign (=) indicates value setting. In the **WHERE** clause, the equal sign indicates comparison. The **WHERE** clause can specify a condition using the equal and other operators.

Generally, the SQL language does not provide a unique ID for a row of data. Therefore, it is impossible to directly specify the rows of the data to be updated.

However, you can specify the rows by setting a condition that only the rows meet. If a table contains primary keys, you can specify a row by primary key.

For details about how to create a table and insert data to it, see **4.1 Creating a Table** and **4.2 Inserting Data to a Table**.

The following is an example of updating data in the table:

- **c_customer_sk** in table **customer_t1** must be changed from **9527** to **9876**:
  ```
  UPDATE customer_t1 SET c_customer_sk = 9876 WHERE c_customer_sk = 9527;
  ```
- **c_customer_sk** in table **customer_t1** must be changed from **9527** to **c_customer_sk + 100**:
  ```
  UPDATE customer_t1 SET c_customer_sk= c_customer_sk + 100 WHERE c_customer_sk= 9527;
  ```
- **c_customer_sk** in table **customer_t1** under the public mode must be changed from **9527** to **9876**:
  ```
  UPDATE public.customer_t1 SET c_customer_sk= 9876 WHERE c_customer_sk= 9527;
  ```
- If the **WHERE** clause is not included, increase all the **c_customer_sk** values by **100**.
  ```
  UPDATE customer_t1 SET c_customer_sk = c_customer_sk + 100;
  ```
- **c_customer_sk** values greater than **9527** in table **customer_t1** must be changed to **9876**:
  ```
  UPDATE customer_t1 SET c_customer_sk = 9876 WHERE c_customer_sk > 9527;
  ```
- You can run an **UPDATE** statement to update multiple columns by specifying multiple values in the **SET** clause. For example:
  ```
  UPDATE customer_t1 SET  c_customer_id = 'Admin', c_first_name = 'Local' WHERE c_customer_sk = 4421;
  ```

After data has been updated or deleted in batches, a large number of deletion markers are generated in the data file. During query, data that is marked out by these deletion markers needs to be scanned as well. In this case, the query performance deteriorates after batch updates or deletions. If data needs to be updated or deleted in batches frequently, you are advised to periodically do **VACUUM FULL** to maintain the query performance.

4.4 Viewing Data

- Query information about all tables in a database through the system catalog **pg_tables**:
  ```
  SELECT * FROM pg_tables;
  ```
- Run the **\d+** command of the **gsql** tool to query table attributes:
  ```
  \d+ customer_t1;
  ```
- Query the data volume of the table **customer_t1**:
  ```
  SELECT count(*) FROM customer_t1;
  ```
- Query all data in table **customer_t1**:
  ```
  SELECT * FROM customer_t1;
  ```
- Query data in column **c_customer_sk**:
  ```
  SELECT c_customer_sk FROM customer_t1;
  ```
- Filter repeated data in column **c_customer_sk**:
  ```
  SELECT DISTINCT( c_customer_sk ) FROM customer_t1;
  ```
- Query all data whose column **c_customer_sk** is **3869**:
  ```
  SELECT * FROM customer_t1 WHERE c_customer_sk = 3869;
  ```
- Sort data based on column **c_customer_sk**.
  ```
  SELECT * FROM customer_t1 ORDER BY c_customer_sk;
  ```


To cancel a query that has been running for a long time, see **Viewing and Stopping the Running Query Statements** in **6 Querying System Catalogs**.

4.5 Deleting Data from a Table

You can delete outdated data from a table by row.

SQL statements can only access and delete an independent row by declaring conditions that match the row. If a table has a primary key column, you can use it to specify a row. You can delete several rows that match the specified condition or delete all the rows from a table.

For example, to delete all the rows whose **c_customer_sk** column is **3869** from table **customer_t1**, run the following statement:

```
DELETE FROM customer_t1 WHERE c_customer_sk = 3869;
```

Delete the records whose **c_customer_sk** is **6885** and **4321** from the **customer_t1** table.

```
DELETE FROM customer_t1 WHERE c_customer_sk in (6885, 4321);
```

Delete the records whose **c_customer_sk** is greater than 4000 and less than 5000 from the **customer_t1** table.

```
DELETE FROM customer_t1 WHERE c_customer_sk > 4000 and c_customer_sk < 5000;
```

To delete all rows from the table, run either of the following statements:

```
DELETE FROM customer_t1;
TRUNCATE TABLE customer_t1;
```

NOTE

If you need to delete an entire table, you are advised to use the **TRUNCATE** statement rather than **DELETE**.

To delete a table, execute the following statement:

```
DROP TABLE customer_t1;
```


5 Loading Sample Data

This section describes how to load data to the default database **gaussdb**. You can obtain sample data from OBS.

NOTE

- Before performing the following operations, ensure that your SQL client has been connected to the cluster.

1. Create a table.

   Copy and run the following statements to create a table in the **gaussdb** database.

```
DROP SCHEMA if exists tpcds cascade;
CREATE SCHEMA tpcds;
SET current_schema TO tpcds;
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);

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);

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);

CREATE TABLE warehouse
(
    w_warehouse_sk            integer               not null,
    w_warehouse_id            char(16)              not null,
    w_warehouse_name          varchar(20)           ,
    w_warehouse_sq_ft         integer               ,
    w_street_number           char(10)              ,
    w_street_name             varchar(60)           ,
    w_street_type             char(15)              ,
    w_suite_number            char(10)              ,
    w_city                    varchar(60)           ,
    w_county                  varchar(30)           ,
    w_state                   char(2)               ,
    w_zip                     char(10)              ,
    w_country                 varchar(20)           ,
    w_gmt_offset              decimal(5,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE ship_mode
(
    sm_ship_mode_sk           integer               not null,
    sm_ship_mode_id           char(16)              not null,
    sm_type                   char(30)              ,
    sm_code                   char(10)              ,
    sm_carrier                char(20)              ,
    sm_contract               char(20)
)WITH (orientation = column, COMPRESSION = MIDDLE);

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);

CREATE TABLE reason
(
    r_reason_sk               integer               not null,
    r_reason_id               char(16)              not null,
    r_reason_desc             char(100)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE income_band
(
    ib_income_band_sk         integer               not null,
    ib_lower_bound            integer                       ,
    ib_upper_bound            integer
)WITH (orientation = column, COMPRESSION = MIDDLE);

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);

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_percentage        decimal(5,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE call_center
(
    cc_call_center_sk       integer             not null,
    cc_call_center_id       char(16)            not null,
    cc_rec_start_date       date                ,
    cc_rec_end_date         date                ,
    cc_closed_date_sk       integer             ,
    cc_open_date_sk         integer             ,
    cc_name                 varchar(50)         ,
    cc_class                varchar(50)         ,
    cc_employees            integer             ,
    cc_sq_ft                integer             ,
    cc_hours                char(20)            ,
    cc_manager              varchar(40)         ,
    cc_mkt_id               integer             ,
    cc_mkt_class            char(50)            ,
    cc_mkt_desc             varchar(100)        ,
    cc_market_manager       varchar(40)         ,
    cc_division             integer             ,
    cc_division_name        varchar(50)         ,
    cc_company              integer             ,
    cc_company_name         char(50)            ,
    cc_street_number        char(10)            ,
    cc_street_name          varchar(60)         ,
    cc_street_type          char(15)            ,
    cc_suite_number         char(10)            ,
    cc_city                 varchar(60)         ,
    cc_county               varchar(30)         ,
    cc_state                char(2)             ,
    cc_zip                  char(10)            ,
    cc_country              varchar(20)         ,
    cc_gmt_offset           decimal(5,2)        ,
    cc_tax_percentage       decimal(5,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

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);

CREATE TABLE web_site
(
    web_site_sk             integer             not null,
    web_site_id             char(16)            not null,
    web_rec_start_date      date                ,
```



```
    web_rec_end_date          date                      ,
    web_name                  varchar(50)               ,
    web_open_date_sk          integer                   ,
    web_close_date_sk         integer                   ,
    web_class                 varchar(50)               ,
    web_manager               varchar(40)               ,
    web_mkt_id                integer                   ,
    web_mkt_class             varchar(50)               ,
    web_mkt_desc              varchar(100)              ,
    web_market_manager        varchar(40)               ,
    web_company_id            integer                   ,
    web_company_name          char(50)                  ,
    web_street_number         char(10)                  ,
    web_street_name           varchar(60)               ,
    web_street_type           char(15)                  ,
    web_suite_number          char(10)                  ,
    web_city                  varchar(60)               ,
    web_county                varchar(30)               ,
    web_state                 char(2)                   ,
    web_zip                   char(10)                  ,
    web_country               varchar(20)               ,
    web_gmt_offset            decimal(5,2)              ,
    web_tax_percentage        decimal(5,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE store_returns
(
    sr_returned_date_sk       integer                   ,
    sr_return_time_sk         integer                   ,
    sr_item_sk                integer             not null,
    sr_customer_sk            integer                   ,
    sr_cdemo_sk               integer                   ,
    sr_hdemo_sk               integer                   ,
    sr_addr_sk                integer                   ,
    sr_store_sk               integer                   ,
    sr_reason_sk              integer                   ,
    sr_ticket_number          integer             not null,
    sr_return_quantity        integer                   ,
    sr_return_amt             decimal(7,2)              ,
    sr_return_tax             decimal(7,2)              ,
    sr_return_amt_inc_tax     decimal(7,2)              ,
    sr_fee                    decimal(7,2)              ,
    sr_return_ship_cost       decimal(7,2)              ,
    sr_refunded_cash          decimal(7,2)              ,
    sr_reversed_charge        decimal(7,2)              ,
    sr_store_credit           decimal(7,2)              ,
    sr_net_loss               decimal(7,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

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);

CREATE TABLE web_page
(
    wp_web_page_sk            integer             not null,
    wp_web_page_id            char(16)            not null,
    wp_rec_start_date         date                      ,
    wp_rec_end_date           date                      ,
    wp_creation_date_sk       integer                   ,
    wp_access_date_sk         integer                   ,
    wp_autogen_flag           char(1)                   ,
    wp_customer_sk            integer                   ,
    wp_url                    varchar(100)              ,
```



```
    wp_type                   char(50)                    ,
    wp_char_count             integer                     ,
    wp_link_count             integer                     ,
    wp_image_count            integer                     ,
    wp_max_ad_count           integer
)WITH (orientation = column, COMPRESSION = MIDDLE);

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);

CREATE TABLE catalog_page
(
    cp_catalog_page_sk        integer               not null,
    cp_catalog_page_id        char(16)              not null,
    cp_start_date_sk          integer                     ,
    cp_end_date_sk            integer                     ,
    cp_department             varchar(50)                 ,
    cp_catalog_number         integer                     ,
    cp_catalog_page_number    integer                     ,
    cp_description            varchar(100)                ,
    cp_type                   varchar(100)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE inventory
(
    inv_date_sk               integer               not null,
    inv_item_sk               integer               not null,
    inv_warehouse_sk          integer               not null,
    inv_quantity_on_hand      integer
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE catalog_returns
(
    cr_returned_date_sk       integer                     ,
    cr_returned_time_sk       integer                     ,
    cr_item_sk                integer               not null,
    cr_refunded_customer_sk   integer                     ,
    cr_refunded_cdemo_sk      integer                     ,
    cr_refunded_hdemo_sk      integer                     ,
    cr_refunded_addr_sk       integer                     ,
    cr_returning_customer_sk  integer                     ,
    cr_returning_cdemo_sk     integer                     ,
    cr_returning_hdemo_sk     integer                     ,
    cr_returning_addr_sk      integer                     ,
    cr_call_center_sk         integer                     ,
    cr_catalog_page_sk        integer                     ,
    cr_ship_mode_sk           integer                     ,
    cr_warehouse_sk           integer                     ,
    cr_reason_sk              integer                     ,
```



```
    cr_order_number           integer               not null,
    cr_return_quantity        integer                       ,
    cr_return_amount          decimal(7,2)                  ,
    cr_return_tax             decimal(7,2)                  ,
    cr_return_amt_inc_tax     decimal(7,2)                  ,
    cr_fee                    decimal(7,2)                  ,
    cr_return_ship_cost       decimal(7,2)                  ,
    cr_refunded_cash          decimal(7,2)                  ,
    cr_reversed_charge        decimal(7,2)                  ,
    cr_store_credit           decimal(7,2)                  ,
    cr_net_loss               decimal(7,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE web_returns
(
    wr_returned_date_sk       integer                       ,
    wr_returned_time_sk       integer                       ,
    wr_item_sk                integer               not null,
    wr_refunded_customer_sk   integer                       ,
    wr_refunded_cdemo_sk      integer                       ,
    wr_refunded_hdemo_sk      integer                       ,
    wr_refunded_addr_sk       integer                       ,
    wr_returning_customer_sk  integer                       ,
    wr_returning_cdemo_sk     integer                       ,
    wr_returning_hdemo_sk     integer                       ,
    wr_returning_addr_sk      integer                       ,
    wr_web_page_sk            integer                       ,
    wr_reason_sk              integer                       ,
    wr_order_number           integer               not null,
    wr_return_quantity        integer                       ,
    wr_return_amt             decimal(7,2)                  ,
    wr_return_tax             decimal(7,2)                  ,
    wr_return_amt_inc_tax     decimal(7,2)                  ,
    wr_fee                    decimal(7,2)                  ,
    wr_return_ship_cost       decimal(7,2)                  ,
    wr_refunded_cash          decimal(7,2)                  ,
    wr_reversed_charge        decimal(7,2)                  ,
    wr_account_credit         decimal(7,2)                  ,
    wr_net_loss               decimal(7,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE web_sales
(
    ws_sold_date_sk           integer                       ,
    ws_sold_time_sk           integer                       ,
    ws_ship_date_sk           integer                       ,
    ws_item_sk                integer               not null,
    ws_bill_customer_sk       integer                       ,
    ws_bill_cdemo_sk          integer                       ,
    ws_bill_hdemo_sk          integer                       ,
    ws_bill_addr_sk           integer                       ,
    ws_ship_customer_sk       integer                       ,
    ws_ship_cdemo_sk          integer                       ,
    ws_ship_hdemo_sk          integer                       ,
    ws_ship_addr_sk           integer                       ,
    ws_web_page_sk            integer                       ,
    ws_web_site_sk            integer                       ,
    ws_ship_mode_sk           integer                       ,
    ws_warehouse_sk           integer                       ,
    ws_promo_sk               integer                       ,
    ws_order_number           integer               not null,
    ws_quantity               integer                       ,
    ws_wholesale_cost         decimal(7,2)                  ,
    ws_list_price             decimal(7,2)                  ,
    ws_sales_price            decimal(7,2)                  ,
    ws_ext_discount_amt       decimal(7,2)                  ,
    ws_ext_sales_price        decimal(7,2)                  ,
    ws_ext_wholesale_cost     decimal(7,2)                  ,
    ws_ext_list_price         decimal(7,2)                  ,
```



```
    ws_ext_tax               decimal(7,2)               ,
    ws_coupon_amt            decimal(7,2)               ,
    ws_ext_ship_cost         decimal(7,2)               ,
    ws_net_paid              decimal(7,2)               ,
    ws_net_paid_inc_tax      decimal(7,2)               ,
    ws_net_paid_inc_ship     decimal(7,2)               ,
    ws_net_paid_inc_ship_tax decimal(7,2)               ,
    ws_net_profit            decimal(7,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

CREATE TABLE catalog_sales
(
    cs_sold_date_sk          integer                    ,
    cs_sold_time_sk          integer                    ,
    cs_ship_date_sk          integer                    ,
    cs_bill_customer_sk      integer                    ,
    cs_bill_cdemo_sk         integer                    ,
    cs_bill_hdemo_sk         integer                    ,
    cs_bill_addr_sk          integer                    ,
    cs_ship_customer_sk      integer                    ,
    cs_ship_cdemo_sk         integer                    ,
    cs_ship_hdemo_sk         integer                    ,
    cs_ship_addr_sk          integer                    ,
    cs_call_center_sk        integer                    ,
    cs_catalog_page_sk       integer                    ,
    cs_ship_mode_sk          integer                    ,
    cs_warehouse_sk          integer                    ,
    cs_item_sk               integer             not null,
    cs_promo_sk              integer                    ,
    cs_order_number          integer             not null,
    cs_quantity              integer                    ,
    cs_wholesale_cost        decimal(7,2)               ,
    cs_list_price            decimal(7,2)               ,
    cs_sales_price           decimal(7,2)               ,
    cs_ext_discount_amt      decimal(7,2)               ,
    cs_ext_sales_price       decimal(7,2)               ,
    cs_ext_wholesale_cost    decimal(7,2)               ,
    cs_ext_list_price        decimal(7,2)               ,
    cs_ext_tax               decimal(7,2)               ,
    cs_coupon_amt            decimal(7,2)               ,
    cs_ext_ship_cost         decimal(7,2)               ,
    cs_net_paid              decimal(7,2)               ,
    cs_net_paid_inc_tax      decimal(7,2)               ,
    cs_net_paid_inc_ship     decimal(7,2)               ,
    cs_net_paid_inc_ship_tax decimal(7,2)               ,
    cs_net_profit            decimal(7,2)
)WITH (orientation = column, COMPRESSION = MIDDLE);

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         integer             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);
```

2. Create an OBS foreign table.

   Copy and run the following statements to create an OBS foreign table in the **gaussdb** database, identifying data format and setting error tolerance.

   NOTE

   Configure the **ACCESS_KEY** and **SECRET_ACCESS_KEY** parameters as required, and run the statements to create a foreign table using the client tool.

   For details about how to obtain the values of **ACCESS_KEY** and **SECRET_ACCESS_KEY**, see "Data Import > Importing Data from OBS in Parallel > Creating Access Keys (AK and SK)".

```
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://dws/download/dws_sample_database_data_files/customer_address/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_customer_demographics_001
(
    cd_demo_sk                integer               not null,
    cd_gender                 char(1)                     ,
    cd_marital_status         char(1)                     ,
    cd_education_status       char(20)                    ,
    cd_purchase_estimate      integer                     ,
    cd_credit_rating          char(10)                    ,
    cd_dep_count              integer                     ,
    cd_dep_employed_count     integer                     ,
    cd_dep_college_count      integer
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/customer_demographics/
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_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://dws/download/dws_sample_database_data_files/ date_dim/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_warehouse_001
(
    w_warehouse_sk          integer             not null,
    w_warehouse_id          char(16)            not null,
    w_warehouse_name        varchar(20)                 ,
    w_warehouse_sq_ft       integer                     ,
    w_street_number         char(10)                    ,
    w_street_name           varchar(60)                 ,
    w_street_type           char(15)                    ,
    w_suite_number          char(10)                    ,
    w_city                  varchar(60)                 ,
    w_county                varchar(30)                 ,
    w_state                 char(2)                     ,
    w_zip                   char(10)                    ,
    w_country               varchar(20)                 ,
    w_gmt_offset            decimal(5,2)
)
SERVER gsmpp_server
```



```
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/ warehouse/warehouse',
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_warehouse_001;

CREATE FOREIGN TABLE obs_from_ship_mode_001
(
    sm_ship_mode_sk           integer               not null,
    sm_ship_mode_id           char(16)              not null,
    sm_type                   char(30)                  ,
    sm_code                   char(10)                  ,
    sm_carrier                char(20)                  ,
    sm_contract               char(20)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/ ship_mode/ship_mode' ,
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_ship_mode_001;

CREATE FOREIGN TABLE obs_from_time_dim_001
(
    t_time_sk                 integer               not null,
    t_time_id                 char(16)              not null,
    t_time                    integer                   ,
    t_hour                    integer                   ,
    t_minute                  integer                   ,
    t_second                  integer                   ,
    t_am_pm                   char(2)                   ,
    t_shift                   char(20)                  ,
    t_sub_shift               char(20)                  ,
    t_meal_time               char(20)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/time_dim/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_reason_001
(
    r_reason_sk               integer               not null,
    r_reason_id               char(16)              not null,
    r_reason_desc             char(100)
)
```



```
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/reason/reason' ,
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_reason_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://dws/download/dws_sample_database_data_files/income_band/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;

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://dws/download/dws_sample_database_data_files/item/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_store_001
(
    s_store_sk              integer              not null,
    s_store_id              char(16)             not null,
    s_rec_start_date          date                       ,
    s_rec_end_date            date                       ,
    s_closed_date_sk          integer                    ,
    s_store_name              varchar(50)                ,
    s_number_employees        integer                    ,
    s_floor_space             integer                    ,
    s_hours                   char(20)                   ,
    s_manager                 varchar(40)                ,
    s_market_id               integer                    ,
    s_geography_class         varchar(100)               ,
    s_market_desc             varchar(100)               ,
    s_market_manager          varchar(40)                ,
    s_division_id             integer                    ,
    s_division_name           varchar(50)                ,
    s_company_id              integer                    ,
    s_company_name            varchar(50)                ,
    s_street_number           varchar(10)                ,
    s_street_name             varchar(60)                ,
    s_street_type             char(15)                   ,
    s_suite_number            char(10)                   ,
    s_city                    varchar(60)                ,
    s_county                  varchar(30)                ,
    s_state                   char(2)                    ,
    s_zip                     char(10)                   ,
    s_country                 varchar(20)                ,
    s_gmt_offset              decimal(5,2)               ,
    s_tax_percentage          decimal(5,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/store/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_call_center_001
(
    cc_call_center_sk         integer              not null,
    cc_call_center_id         char(16)             not null,
    cc_rec_start_date         date                       ,
    cc_rec_end_date           date                       ,
    cc_closed_date_sk         integer                    ,
    cc_open_date_sk           integer                    ,
    cc_name                   varchar(50)                ,
    cc_class                  varchar(50)                ,
    cc_employees              integer                    ,
    cc_sq_ft                  integer                    ,
    cc_hours                  char(20)                   ,
    cc_manager                varchar(40)                ,
    cc_mkt_id                 integer                    ,
    cc_mkt_class              char(50)                   ,
    cc_mkt_desc               varchar(100)               ,
    cc_market_manager         varchar(40)                ,
    cc_division               integer                    ,
```



```
    cc_division_name          varchar(50)                ,
    cc_company                integer                    ,
    cc_company_name           char(50)                   ,
    cc_street_number          char(10)                   ,
    cc_street_name            varchar(60)                ,
    cc_street_type            char(15)                   ,
    cc_suite_number           char(10)                   ,
    cc_city                   varchar(60)                ,
    cc_county                 varchar(30)                ,
    cc_state                  char(2)                    ,
    cc_zip                    char(10)                   ,
    cc_country                varchar(20)                ,
    cc_gmt_offset             decimal(5,2)               ,
    cc_tax_percentage         decimal(5,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/call_center/call_center',
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_call_center_001;

CREATE FOREIGN TABLE obs_from_customer_001
(
    c_customer_sk             integer               not null,
    c_customer_id             char(16)              not null,
    c_current_cdemo_sk        integer                    ,
    c_current_hdemo_sk        integer                    ,
    c_current_addr_sk         integer                    ,
    c_first_shipto_date_sk    integer                    ,
    c_first_sales_date_sk     integer                    ,
    c_salutation              char(10)                   ,
    c_first_name              char(20)                   ,
    c_last_name               char(30)                   ,
    c_preferred_cust_flag     char(1)                    ,
    c_birth_day               integer                    ,
    c_birth_month             integer                    ,
    c_birth_year              integer                    ,
    c_birth_country           varchar(20)                ,
    c_login                   char(13)                   ,
    c_email_address           char(50)                   ,
    c_last_review_date        char(10)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/customer/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_web_site_001
(
    web_site_sk               integer               not null,
    web_site_id               char(16)              not null,
    web_rec_start_date        date                       ,
```



```
    web_rec_end_date          date                      ,
    web_name                  varchar(50)               ,
    web_open_date_sk          integer                   ,
    web_close_date_sk         integer                   ,
    web_class                 varchar(50)               ,
    web_manager               varchar(40)               ,
    web_mkt_id                integer                   ,
    web_mkt_class             varchar(50)               ,
    web_mkt_desc              varchar(100)              ,
    web_market_manager        varchar(40)               ,
    web_company_id            integer                   ,
    web_company_name          char(50)                  ,
    web_street_number         char(10)                  ,
    web_street_name           varchar(60)               ,
    web_street_type           char(15)                  ,
    web_suite_number          char(10)                  ,
    web_city                  varchar(60)               ,
    web_county                varchar(30)               ,
    web_state                 char(2)                   ,
    web_zip                   char(10)                  ,
    web_country               varchar(20)               ,
    web_gmt_offset            decimal(5,2)              ,
    web_tax_percentage        decimal(5,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/web_site/web_site' ,
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_web_site_001;

CREATE FOREIGN TABLE obs_from_store_returns_001
(
    sr_returned_date_sk       integer                   ,
    sr_return_time_sk         integer                   ,
    sr_item_sk                integer             not null,
    sr_customer_sk            integer                   ,
    sr_cdemo_sk               integer                   ,
    sr_hdemo_sk               integer                   ,
    sr_addr_sk                integer                   ,
    sr_store_sk               integer                   ,
    sr_reason_sk              integer                   ,
    sr_ticket_number          bigint              not null,
    sr_return_quantity        integer                   ,
    sr_return_amt             decimal(7,2)              ,
    sr_return_tax             decimal(7,2)              ,
    sr_return_amt_inc_tax     decimal(7,2)              ,
    sr_fee                    decimal(7,2)              ,
    sr_return_ship_cost       decimal(7,2)              ,
    sr_refunded_cash          decimal(7,2)              ,
    sr_reversed_charge        decimal(7,2)              ,
    sr_store_credit           decimal(7,2)              ,
    sr_net_loss               decimal(7,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/store_returns/store_returns' ,
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_returns_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://dws/download/dws_sample_database_data_files/household_demographics/
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_web_page_001
(
    wp_web_page_sk          integer              not null,
    wp_web_page_id          char(16)             not null,
    wp_rec_start_date       date                         ,
    wp_rec_end_date         date                         ,
    wp_creation_date_sk     integer                      ,
    wp_access_date_sk       integer                      ,
    wp_autogen_flag         char(1)                      ,
    wp_customer_sk          integer                      ,
    wp_url                  varchar(100)                 ,
    wp_type                 char(50)                     ,
    wp_char_count           integer                      ,
    wp_link_count           integer                      ,
    wp_image_count          integer                      ,
    wp_max_ad_count         integer
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/web_page/web_page' ,
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_web_page_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://dws/download/dws_sample_database_data_files/promotion/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_catalog_page_001
(
    cp_catalog_page_sk        integer               not null,
    cp_catalog_page_id        char(16)              not null,
    cp_start_date_sk          integer                   ,
    cp_end_date_sk            integer                   ,
    cp_department             varchar(50)               ,
    cp_catalog_number         integer                   ,
    cp_catalog_page_number    integer                   ,
    cp_description            varchar(100)              ,
    cp_type                   varchar(100)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/catalog_page/catalog_page' ,
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_catalog_page_001;

CREATE FOREIGN TABLE obs_from_inventory_001
(
    inv_date_sk               integer               not null,
    inv_item_sk               integer               not null,
    inv_warehouse_sk          integer               not null,
    inv_quantity_on_hand      integer
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/inventory/inventory' ,
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_inventory_001;

CREATE FOREIGN TABLE obs_from_catalog_returns_001
(
    cr_returned_date_sk       integer                       ,
    cr_returned_time_sk       integer                       ,
    cr_item_sk                integer               not null,
    cr_refunded_customer_sk   integer                       ,
    cr_refunded_cdemo_sk      integer                       ,
    cr_refunded_hdemo_sk      integer                       ,
    cr_refunded_addr_sk       integer                       ,
    cr_returning_customer_sk  integer                       ,
    cr_returning_cdemo_sk     integer                       ,
    cr_returning_hdemo_sk     integer                       ,
    cr_returning_addr_sk      integer                       ,
    cr_call_center_sk         integer                       ,
    cr_catalog_page_sk        integer                       ,
    cr_ship_mode_sk           integer                       ,
    cr_warehouse_sk           integer                       ,
    cr_reason_sk              integer                       ,
    cr_order_number           bigint                not null,
    cr_return_quantity        integer                       ,
    cr_return_amount          decimal(7,2)                  ,
    cr_return_tax             decimal(7,2)                  ,
    cr_return_amt_inc_tax     decimal(7,2)                  ,
    cr_fee                    decimal(7,2)                  ,
    cr_return_ship_cost       decimal(7,2)                  ,
    cr_refunded_cash          decimal(7,2)                  ,
    cr_reversed_charge        decimal(7,2)                  ,
    cr_store_credit           decimal(7,2)                  ,
    cr_net_loss               decimal(7,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/catalog_returns/catalog_returns' ,
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_catalog_returns_001;

CREATE FOREIGN TABLE obs_from_web_returns_001
(
    wr_returned_date_sk       integer                       ,
    wr_returned_time_sk       integer                       ,
    wr_item_sk                integer               not null,
    wr_refunded_customer_sk   integer                       ,
    wr_refunded_cdemo_sk      integer                       ,
    wr_refunded_hdemo_sk      integer                       ,
    wr_refunded_addr_sk       integer                       ,
    wr_returning_customer_sk  integer                       ,
    wr_returning_cdemo_sk     integer                       ,
    wr_returning_hdemo_sk     integer                       ,
    wr_returning_addr_sk      integer                       ,
    wr_web_page_sk            integer                       ,
    wr_reason_sk              integer                       ,
    wr_order_number           bigint                not null,
    wr_return_quantity        integer                       ,
    wr_return_amt             decimal(7,2)                  ,
    wr_return_tax             decimal(7,2)                  ,
    wr_return_amt_inc_tax     decimal(7,2)                  ,
```



```
    wr_fee                    decimal(7,2)                ,
    wr_return_ship_cost       decimal(7,2)                ,
    wr_refunded_cash          decimal(7,2)                ,
    wr_reversed_charge        decimal(7,2)                ,
    wr_account_credit         decimal(7,2)                ,
    wr_net_loss               decimal(7,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/web_returns/web_returns' ,
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_web_returns_001;

CREATE FOREIGN TABLE obs_from_web_sales_001
(
    ws_sold_date_sk           integer                     ,
    ws_sold_time_sk           integer                     ,
    ws_ship_date_sk           integer                     ,
    ws_item_sk                integer               not null,
    ws_bill_customer_sk       integer                     ,
    ws_bill_cdemo_sk          integer                     ,
    ws_bill_hdemo_sk          integer                     ,
    ws_bill_addr_sk           integer                     ,
    ws_ship_customer_sk       integer                     ,
    ws_ship_cdemo_sk          integer                     ,
    ws_ship_hdemo_sk          integer                     ,
    ws_ship_addr_sk           integer                     ,
    ws_web_page_sk            integer                     ,
    ws_web_site_sk            integer                     ,
    ws_ship_mode_sk           integer                     ,
    ws_warehouse_sk           integer                     ,
    ws_promo_sk               integer                     ,
    ws_order_number           bigint                not null,
    ws_quantity               integer                     ,
    ws_wholesale_cost         decimal(7,2)                ,
    ws_list_price             decimal(7,2)                ,
    ws_sales_price            decimal(7,2)                ,
    ws_ext_discount_amt       decimal(7,2)                ,
    ws_ext_sales_price        decimal(7,2)                ,
    ws_ext_wholesale_cost     decimal(7,2)                ,
    ws_ext_list_price         decimal(7,2)                ,
    ws_ext_tax                decimal(7,2)                ,
    ws_coupon_amt             decimal(7,2)                ,
    ws_ext_ship_cost          decimal(7,2)                ,
    ws_net_paid               decimal(7,2)                ,
    ws_net_paid_inc_tax       decimal(7,2)                ,
    ws_net_paid_inc_ship      decimal(7,2)                ,
    ws_net_paid_inc_ship_tax  decimal(7,2)                ,
    ws_net_profit             decimal(7,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/web_sales/web_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_web_sales_001;

CREATE FOREIGN TABLE obs_from_catalog_sales_001
(
    cs_sold_date_sk           integer                       ,
    cs_sold_time_sk           integer                       ,
    cs_ship_date_sk           integer                       ,
    cs_bill_customer_sk       integer                       ,
    cs_bill_cdemo_sk          integer                       ,
    cs_bill_hdemo_sk          integer                       ,
    cs_bill_addr_sk           integer                       ,
    cs_ship_customer_sk       integer                       ,
    cs_ship_cdemo_sk          integer                       ,
    cs_ship_hdemo_sk          integer                       ,
    cs_ship_addr_sk           integer                       ,
    cs_call_center_sk         integer                       ,
    cs_catalog_page_sk        integer                       ,
    cs_ship_mode_sk           integer                       ,
    cs_warehouse_sk           integer                       ,
    cs_item_sk                integer               not null,
    cs_promo_sk               integer                       ,
    cs_order_number           bigint                not null,
    cs_quantity               integer                       ,
    cs_wholesale_cost         decimal(7,2)                  ,
    cs_list_price             decimal(7,2)                  ,
    cs_sales_price            decimal(7,2)                  ,
    cs_ext_discount_amt       decimal(7,2)                  ,
    cs_ext_sales_price        decimal(7,2)                  ,
    cs_ext_wholesale_cost     decimal(7,2)                  ,
    cs_ext_list_price         decimal(7,2)                  ,
    cs_ext_tax                decimal(7,2)                  ,
    cs_coupon_amt             decimal(7,2)                  ,
    cs_ext_ship_cost          decimal(7,2)                  ,
    cs_net_paid               decimal(7,2)                  ,
    cs_net_paid_inc_tax       decimal(7,2)                  ,
    cs_net_paid_inc_ship      decimal(7,2)                  ,
    cs_net_paid_inc_ship_tax  decimal(7,2)                  ,
    cs_net_profit             decimal(7,2)
)
SERVER gsmpp_server
OPTIONS (
location 'obs://dws/download/dws_sample_database_data_files/catalog_sales/catalog_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_catalog_sales_001;

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://dws/download/dws_sample_database_data_files/store_sales/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;
```

3. Run **INSERT** to insert data.

```
INSERT INTO customer_address SELECT * FROM obs_from_customer_address_001 ;
INSERT INTO customer_demographics SELECT * FROM obs_from_customer_demographics_001 ;
INSERT INTO date_dim SELECT * FROM obs_from_date_dim_001;
INSERT INTO warehouse SELECT * FROM obs_from_warehouse_001;
INSERT INTO ship_mode SELECT * FROM obs_from_ship_mode_001;
INSERT INTO time_dim SELECT * FROM obs_from_time_dim_001;
INSERT INTO reason SELECT * FROM obs_from_reason_001;
INSERT INTO income_band SELECT * FROM obs_from_income_band_001;
INSERT INTO item SELECT * FROM obs_from_item_001;
INSERT INTO store SELECT * FROM obs_from_store_001;
INSERT INTO call_center SELECT * FROM obs_from_call_center_001;
INSERT INTO customer SELECT * FROM obs_from_customer_001;
INSERT INTO web_site SELECT * FROM obs_from_web_site_001;
INSERT INTO store_returns SELECT * FROM obs_from_store_returns_001;
INSERT INTO household_demographics SELECT * FROM obs_from_household_demographics_001;
INSERT INTO web_page SELECT * FROM obs_from_web_page_001;
INSERT INTO promotion SELECT * FROM obs_from_promotion_001;
INSERT INTO catalog_page SELECT * FROM obs_from_catalog_page_001;
INSERT INTO inventory SELECT * FROM obs_from_inventory_001;
INSERT INTO catalog_returns SELECT * FROM obs_from_catalog_returns_001;
INSERT INTO web_returns SELECT * FROM obs_from_web_returns_001;
INSERT INTO web_sales SELECT * FROM obs_from_web_sales_001;
INSERT INTO catalog_sales SELECT * FROM obs_from_catalog_sales_001;
INSERT INTO store_sales SELECT * FROM obs_from_store_sales_001;
```

4. Optimize performance.

```
ANALYZE customer_address;
ANALYZE customer_demographics;
ANALYZE date_dim;
ANALYZE warehouse;
ANALYZE ship_mode;
ANALYZE time_dim;
ANALYZE reason;
ANALYZE income_band;
ANALYZE item;
ANALYZE store;
ANALYZE call_center;
ANALYZE customer;
ANALYZE web_site;
ANALYZE store_returns;
ANALYZE household_demographics;
ANALYZE web_page;
ANALYZE promotion;
ANALYZE catalog_page;
```



```
ANALYZE inventory;
ANALYZE catalog_returns;
ANALYZE web_returns;
ANALYZE web_sales;
ANALYZE catalog_sales;
ANALYZE store_sales;
```

5. Run **SELECT** to query the database. For details, see "SELECT."

```
-- Query all records and sort them in alphabetic order.
SELECT r_reason_desc FROM tpcds.reason ORDER BY r_reason_desc;

-- Filter based on query conditions, and group the results:
SELECT r_reason_id, AVG(r_reason_sk) FROM tpcds.reason GROUP BY r_reason_id HAVING
AVG(r_reason_sk) > 25;
```


6 Querying System Catalogs

In addition to the created tables, a database contains many system catalogs. These system catalogs contain cluster installation information and information about various queries and processes of GaussDB(DWS). You can collect information about a database by querying system catalogs.

The description of each table in "System Catalogs and System Views" in the *Data Warehouse Service (DWS) Developer Guide* specifies whether the table is visible to all users or only to the initial user. To query tables that are visible only to the initial user, log in as the initial user.

Querying Database Tables

For example, you can run the following statement to query the **PG_TABLES** system catalog for all tables in the **public** schema:

```
SELECT distinct(tablename) FROM pg_tables WHERE SCHEMANAME = 'public';
```

Information similar to the following is displayed:

```
    tablename
-------------------
 err_hr_staffs
 test
 err_hr_staffs_ft3
 web_returns_p1
 mig_seq_table
 films4
(6 rows)
```

Querying Database Users

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

```
SELECT * FROM pg_user;
             usename              | usesysid | usecreatedb | usesuper | usecatupd | userepl |  passwd  | valbegin |
valuntil |   respool
   | parent | spacelimit | useconfig
-----------------------------------+----------+-------------+----------+-----------+---------+----------+----------
+----------+----------
----+--------+------------+-----------
dfc22b86afbd9a745668c3ecd0f15ec18 |    17107 | f           | f        | f         | f       | ******** |          |          |
default_p
ool |       0 |            |
```



```
guest                       |    17103 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |
Ruby                        |       10 | t          | t       | t        | t      | ******** |         |          | default_p
ool |      0 |           |
dbadmin                     |    16404 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |
lily                        |    16482 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |
jack                        |    16478 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |


(6 rows)
```

GaussDB(DWS) uses **Ruby** to perform routine management and O&M. You can add **WHERE usesysid > 10** to the **SELECT** statement so that only specified usernames are displayed.

```
SELECT * FROM pg_user WHERE usesysid > 10;
             usename              | usesysid | usecreatedb | usesuper | usecatupd | userepl |   passwd   | valbegin |
valuntil |   respool
   | parent | spacelimit | useconfig
-----------------------------------+----------+-------------+----------+-----------+---------+----------+----------
+----------+----------
----+--------+------------+-----------
dfc22b86afbd9a745668c3ecd0f15ec18 |    17107 | f          | f       | f        | f      | ******** |         |          |
default_p
ool |      0 |           |
guest                       |    17103 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |
dbadmin                     |    16404 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |
lily                        |    16482 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |
jack                        |    16478 | f          | f       | f        | f      | ******** |         |          | default_p
ool |      0 |           |


(5 rows)
```

Querying User Attributes

PG_AUTHID can be used to view the attribute list of all users in the database.

```
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 |
rolmonitoradmin | roloperatoradmin | rolpolicyadmin
---------+----------+------------+---------------+-------------+--------------+-------------+----------------
+---------------+----------------+--------------+-------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------+-------
--------+---------------+--------------+----------+-------------+-------------+---------+--------------+--------------
+---------------+-------------+-----------------+------------------+----------------
 dbadmin | f        | t          | f             | f           | f           | t          | f             | f             | t             |
-1 | sha256ce0ea617e7b7c0f2d38b00a12261b5e98ce18c218
89a30ebf410631c52f882d376141f31b47b67b0ceec9d10c931358140d276009bd8d19ac6a5647558c8b70d009
986e774c2ba9563e42f4331629379d40720e4a3e0997c2b592833db778908md5eb0c1ffc5c76ef6272debb03f58
5b0b9ecdfecefade |
       |             | default_pool | f        |           0 |            | n       |             |             |             |             |
f             | f             | f
 Ruby    | t        | t          | t             | t           | t           | t          | t             | t             | t             |
-1 | sha256d2058470eb2cead16d1d85a2b69207bc33e020bd4
530b67102c9c237dd2cb5d1ebae9d98c88ebf8f51950a333a4bb436488df40e645eb3d346af6c401c8fe5f83b208
7349dcccf38fd1eb8ec828b27f28af4e5066549ba0bb6d249f82c664151md5417898ceaa6a205cb03d1b8df8fb9
2f7ecdfecefade |
       |             | default_pool | t        |           0 |            | n       |           0 |             |             |             |
```



```
t             | t             | t
(2 rows)
```

Query the permissions of user **joe**.

```
SELECT * FROM pg_authid where rolname = 'joe';
```

Querying and Stopping the Running Query Statements

You can view the running query statements in the **PG_STAT_ACTIVITY** view.

Step 1 Set **track_activities** to **on**.

```
SET track_activities = on;
```

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

Step 2 Query the information about running query statements, such as the user who runs the statements and the connected database, query status, and PID of the statements.

```
SELECT datname, usename, state,pid FROM pg_stat_activity;
 datname  | usename | state  |       pid
----------+---------+--------+-----------------
 postgres | Ruby    | active | 140298793514752
 postgres | Ruby    | active | 140298718004992
 postgres | Ruby    | idle   | 140298650908416
 postgres | Ruby    | idle   | 140298625742592
 postgres | dbadmin | active | 140298575406848
(5 rows)
```

If **state** is **idle**, the connection is idle and requires a user to enter a command.

To identify queries that are not idle, run the following command:

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

Step 3 To cancel queries that have been running for a long time, use the **PG_TERMINATE_BACKEND** function to end sessions based on the thread ID.

```
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, a user terminates the current session.

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
```


NOTE

If the **PG_TERMINATE_BACKEND** function is used to terminate the backend threads of the current session, the gsql client will be reconnected automatically rather than be logged out. Information **The connection to the server was lost. Attempting reset: Succeeded.** is returned.

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

----End

7 Creating and Managing Schemas

Background

Based on schema management, multiple users can use the same database without conflicts. Database objects can be organized as manageable logical groups. In addition, third-party applications can be added to the same schema without causing conflicts. Schema management involves creating a schema, using a schema, deleting a schema, setting a search path for a schema, and setting schema permissions.

Important Notes

- The database cluster has one or more named databases. Users and user groups are shared within a cluster, but their data is exclusive. Any user who has connected to a server can only access the database that is specified in the connection request.
- A database can have one or more schemas, and a schema can contain tables and other data objects, such as data types, functions, and operators. One object name can be used in different schemas. For example, both schema1 and schema2 can have a table named mytable.
- Different from databases, schemas are not isolated. You can access the objects in a schema of the connected database based on your schema permissions. To manage schema permissions, you need to have a good understanding of the database permissions.
- A schema named with the **PG_** prefix cannot be created because this type of schema is reserved for the database system.
- If a user is created, a schema named after the user will also be created in the current database.
- To reference a table that is not modified with a schema name, the system uses **search_path** to find the schema that the table belongs to. **pg_temp** and **pg_catalog** are always the first two schemas to be searched no matter whether or how they are specified in **search_path**. **search_path** is a schema name list, and the first table detected in it is the target table. If no target table is found, an error will be reported. (If a table exists but the schema it belongs to is not listed in **search_path**, the search fails as well.) The first schema in **search_path** is called **current schema**. This schema is the first one to be searched. If no schema name is declared, newly created database objects are saved in this schema by default.

- Each database has a **pg_catalog** schema, which contains system catalogs and all built-in data types, functions, and operators. **pg_catalog** is a part of the search path and has the second highest search priority. It is searched after the schema of temporary tables and before other schemas specified in **search_path**. This search order ensures that database built-in objects can be found. To use a custom object that has the same name as a built-in object, you can specify the schema of the custom object.

Procedure

- Create a schema.
  - Run the following command to create a schema:
    ```
    CREATE SCHEMA myschema;
    ```
    If the following information is displayed, the schema named **myschema** has been successfully created:
    ```
    CREATE SCHEMA
    ```
    To create or access an object in the schema, the object name in the command should be composed of the schema name and the object name, which are separated by a dot (.), for example, **myschema.table**.
  - Run the following command to create a schema and specify the owner:
    ```
    CREATE SCHEMA myschema AUTHORIZATION dbadmin;
    ```
    If the following information is displayed, the **myschema** schema that belongs to **dbadmin** has been created successfully:
    ```
    CREATE SCHEMA
    ```
- Use a schema.

  If you want to create or access an object in a specified schema, the object name must contain the schema name. To be specific, the name consists of a schema name and an object name, which are separated by a dot (.).
  - Run the following command to create table **mytable** in **myschema**:
    ```
    CREATE TABLE myschema.mytable(id int, name varchar(20));
    ```
    To specify the location of an object, the object name must contain the schema name.
  - Run the following command to query all data of table **mytable** in **myschema**:
    ```
    SELECT * FROM myschema.mytable;
    id | name
    ----+------
    (0 rows)
    ```
- View **search_path** of a schema.

  You can set **search_path** to specify the sequence of schemas in which objects are searched. The first schema listed in **search_path** will become the default schema. If no schema is specified during object creation, the object will be created in the default schema.
  - Run the following command to view **search_path**:
    ```
    SHOW SEARCH_PATH;
     search_path
    ----------------
     "$user",public
    (1 row)
    ```
  - Run the following command to set **search_path** to **myschema** and **public** (**myschema** is searched first):
    ```
    SET SEARCH_PATH TO myschema, public;
    SET
    ```


- Set permissions for a schema.

  By default, a user can only access database objects in its own schema. Only after a user is granted with the usage permission on a schema by the schema owner, the user can access the objects in the schema.

  By granting the **CREATE** permission for a schema to a user, the user can create objects in this schema.

  - Run the following command to view the current schema:

    ```
    SELECT current_schema();
     current_schema
    ----------------
     myschema
    (1 row)
    ```

  - Run the following commands to create user **jack** and grant the usage permission on **myschema** to the user:

    ```
    CREATE USER jack IDENTIFIED BY 'password';
    GRANT USAGE ON schema myschema TO jack;
    ```

  - Run the following command to revoke the **USAGE** permission for **myschema** from **jack**:

    ```
    REVOKE USAGE ON schema myschema FROM jack;
    ```

- Delete a schema.

  - If a schema is empty, that is, it contains no database object, you can execute the **DROP SCHEMA** statement to delete it. For example, run the following command to delete an empty schema named **nullschema**:

    ```
    DROP SCHEMA IF EXISTS nullschema;
    ```

  - To delete a schema that is not null, use the keyword **CASCADE** to delete it and all its objects. For example, run the following command to delete **myschema** and all objects in it:

    ```
    DROP SCHEMA myschema CASCADE;
    ```

  - Delete user **jack**.

    ```
    DROP USER jack;
    ```


8 Creating and Managing Partitioned Tables

Background

GaussDB(DWS) supports range partitioned tables.

Range partitioned table: Data within a specific range is mapped onto each partition. The range is determined by the partition key specified when the partitioned table is created. This partitioning mode is most commonly used. The partition key is usually a date. For example, sales data is partitioned by month.

A partitioned table has the following advantages over an ordinary table:

- High query performance: The system queries only the concerned partitions rather than the whole table, improving the query efficiency.
- High availability: If a partition is faulty, data in the other partitions is still available.
- Easy maintenance: You only need to fix the faulty partition.
- Balanced I/O: Partitions can be mapped to different disks to balance I/O and improve the overall system performance.

To convert an ordinary table to a partitioned table, you need to create a partitioned table and import data to it from the ordinary table. When you design tables, plan whether to use partitioned tables based on service requirements.

Procedure

- Perform the following operations on a range partitioned table.
  - Create a range partitioned table:

```
CREATE TABLE tpcds.customer_address
(
    ca_address_sk       integer                 NOT NULL   ,
    ca_address_id       character(16)           NOT NULL   ,
    ca_street_number    character(10)                      ,
    ca_street_name      character varying(60)              ,
    ca_street_type      character(15)                      ,
    ca_suite_number     character(10)                      ,
    ca_city             character varying(60)              ,
    ca_county           character varying(30)              ,
    ca_state            character(2)                       ,
    ca_zip              character(10)                      ,
```

```
    ca_country          character varying(20)          ,
    ca_gmt_offset       numeric(5,2)                   ,
    ca_location_type    character(20)
)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE (ca_address_sk)
(
        PARTITION P1 VALUES LESS THAN(5000),
        PARTITION P2 VALUES LESS THAN(10000),
        PARTITION P3 VALUES LESS THAN(15000),
        PARTITION P4 VALUES LESS THAN(20000),
        PARTITION P5 VALUES LESS THAN(25000),
        PARTITION P6 VALUES LESS THAN(30000),
        PARTITION P7 VALUES LESS THAN(40000),
        PARTITION P8 VALUES LESS THAN(MAXVALUE)
)
ENABLE ROW MOVEMENT;
```

If the following information is displayed, the table is created.

```
CREATE TABLE
```

NOTE

Create a maximum of 1000 column-store partitioned tables.

– Insert data.

Insert data from the **tpcds.customer_address** table to the **tpcds.web_returns_p2** table.

For example, you can run the following command to insert the data of the **tpcds.customer_address** table into its backup table **tpcds.web_returns_p2**:

```
CREATE TABLE tpcds.web_returns_p2
(
    ca_address_sk       integer                 NOT NULL   ,
    ca_address_id       character(16)           NOT NULL   ,
    ca_street_number    character(10)                      ,
    ca_street_name      character varying(60)              ,
    ca_street_type      character(15)                      ,
    ca_suite_number     character(10)                      ,
    ca_city             character varying(60)              ,
    ca_county           character varying(30)              ,
    ca_state            character(2)                       ,
    ca_zip              character(10)                      ,
    ca_country          character varying(20)              ,
    ca_gmt_offset       numeric(5,2)                       ,
    ca_location_type    character(20)
)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE (ca_address_sk)
(
        PARTITION P1 VALUES LESS THAN(5000),
        PARTITION P2 VALUES LESS THAN(10000),
        PARTITION P3 VALUES LESS THAN(15000),
        PARTITION P4 VALUES LESS THAN(20000),
        PARTITION P5 VALUES LESS THAN(25000),
        PARTITION P6 VALUES LESS THAN(30000),
        PARTITION P7 VALUES LESS THAN(40000),
        PARTITION P8 VALUES LESS THAN(MAXVALUE)
)
ENABLE ROW MOVEMENT;
CREATE TABLE
INSERT INTO tpcds.web_returns_p2 SELECT * FROM tpcds.customer_address;
INSERT 0 0
```


> **NOTE**
>
> **ROW MOVEMENT** is disabled by default. In this case, cross-partition update is not allowed. To enable cross-partition update, specify **ENABLE ROW MOVEMENT**. However, if **SELECT FOR UPDATE** is executed concurrently to query the partitioned table, the query results may be inconsistent. Therefore, exercise caution when performing this operation.

- Modify the row movement attributes of a partitioned table.
  ```
  ALTER TABLE tpcds.web_returns_p2 DISABLE ROW MOVEMENT;
  ```
- Delete a partition.

  Run the following command to delete partition **P8**:
  ```
  ALTER TABLE tpcds.web_returns_p2 DROP PARTITION P8;
  ```
- Add a partition.

  Run the following command to add partition **P8** and set its range to [40000, MAXVALUE]:
  ```
  ALTER TABLE tpcds.web_returns_p2 ADD PARTITION P8 VALUES LESS THAN (MAXVALUE);
  ```
- Rename a partition.
  - Run the following command to rename partition **P8** to **P_9**:
    ```
    ALTER TABLE tpcds.web_returns_p2 RENAME PARTITION P8 TO P_9;
    ```
  - Run the following command to rename partition **P_9** to **P8**:
    ```
    ALTER TABLE tpcds.web_returns_p2 RENAME PARTITION FOR (40000) TO P8;
    ```
- Query a partition.

  Run the following command to query partition **P7**:
  ```
  SELECT * FROM tpcds.web_returns_p2 PARTITION (P7);
  SELECT * FROM tpcds.web_returns_p2 PARTITION FOR (35888);
  ```
- View partitioned tables using the system catalog **dba_tab_partitions**.
  ```
  SELECT * FROM dba_tab_partitions WHERE table_name='customer_address';
  ```
- Delete a partitioned table.
  ```
  DROP TABLE tpcds.web_returns_p2;
  ```


9 Creating and Managing Indexes

Background

Indexes accelerate the data access speed but also add the processing time of the insert, update, and delete operations. Therefore, before creating an index, consider whether it is necessary and determine the columns where indexes will be created. You can determine whether to add an index for a table by analyzing the service processing and data use of applications, as well as columns that are frequently used as search criteria or need to be sorted.

Indexes are created based on columns in database tables. When creating indexes, you need to determine the columns, which can be:

- Columns that are frequently searched: The search efficiency can be improved.
- The uniqueness of the columns and the data sequence structures is ensured.
- Columns that usually function as foreign keys and are used for connections. Then the connection efficiency is improved.
- Columns that are usually searched for by a specified scope. These indexes have already been arranged in a sequence, and the specified scope is contiguous.
- Columns that need to be arranged in a sequence. These indexes have already been arranged in a sequence, so the sequence query time is accelerated.
- Columns that usually use the WHERE clause. Then the condition decision efficiency is increased.
- Fields that are frequently used after keywords, such as **ORDER BY**, **GROUP BY**, and **DISTINCT**.

  NOTE

  - After an index is created, the system automatically determines when to reference it. If the system determines that indexing is faster than sequenced scanning, the index will be used.
  - After an index is successfully created, it must be synchronized with the associated table to ensure new data can be accurately located. Therefore, data operations increase. Therefore, delete unnecessary indexes periodically.
  - After an index is created, it takes effect on the existing data in the table.

Procedure

For details about the procedure for creating a partitioned table, see **8 Creating and Managing Partitioned Tables**.

- Creating an Index
  - Create the partitioned table index **tpcds_web_returns_p2_index1** without specifying the partition name.
    ```
    CREATE INDEX tpcds_web_returns_p2_index1 ON tpcds.web_returns_p2 (ca_address_id) LOCAL;
    ```
    If the following information is displayed, the index has been created.
    ```
    CREATE INDEX
    ```
  - Create the partitioned table index **tpcds_web_returns_p2_index2** and specify index names for all partitions. Currently, specifying index names for partial partitions is not allowed.
    ```
    CREATE INDEX tpcds_web_returns_p2_index2 ON tpcds.web_returns_p2 (ca_address_sk) LOCAL
    (
        PARTITION web_returns_p2_P1_index,
        PARTITION web_returns_p2_P2_index TABLESPACE example3,
        PARTITION web_returns_p2_P3_index TABLESPACE example4,
        PARTITION web_returns_p2_P4_index,
        PARTITION web_returns_p2_P5_index,
        PARTITION web_returns_p2_P6_index,
        PARTITION web_returns_p2_P7_index,
        PARTITION web_returns_p2_P8_index
    ) TABLESPACE example2;
    ```
    If the following information is displayed, the index has been created.
    ```
    CREATE INDEX
    ```
- Renaming an index partition

  Rename the name of index partition **web_returns_p2_P8_index** to *web_returns_p2_P8_index*_**new**.
  ```
  ALTER INDEX tpcds.tpcds_web_returns_p2_index2 RENAME PARTITION web_returns_p2_P8_index TO web_returns_p2_P8_index_new;
  ```
  If the following information is displayed, the index has been renamed.
  ```
  ALTER INDEX
  ```
- Querying indexes
  - Run the following command to query all indexes defined by the system and users:
    ```
    SELECT RELNAME FROM PG_CLASS WHERE RELKIND='i';
    ```
  - Run the following command to query information about a specified index:
    ```
    \di+ tpcds.tpcds_web_returns_p2_index2
    ```
- Deleting an index
  ```
  DROP INDEX tpcds.tpcds_web_returns_p2_index1;
  DROP INDEX tpcds.tpcds_web_returns_p2_index2;
  ```
  If the following output is displayed, the index has been deleted.
  ```
  DROP INDEX
  ```

GaussDB(DWS) supports four methods for creating indexes. For details, see **Table 9-1**.

NOTE

- After an index is created, the system automatically determines when to reference it. If the system determines that indexing is faster than sequenced scanning, the index will be used.
- After an index is successfully created, it must be synchronized with the associated table to ensure new data can be accurately located. Therefore, data operations increase. Therefore, delete unnecessary indexes periodically.

Table 9-1 Indexing Method

Indexing Method	Description
Unique index	Refers to an index that constrains the uniqueness of an index attribute or an attribute group. If a table declares unique constraints or primary keys, GaussDB(DWS) automatically creates unique indexes (or composite indexes) for columns that form the primary keys or unique constraints. Currently, only B-tree can create a unique index in GaussDB(DWS).
Composite index	Refers to an index that can be defined for multiple attributes of a table. Currently, composite indexes can be created only for B-tree in GaussDB(DWS) and a maximum of 32 columns can share a composite index.
Partial index	Refers to an index that can be created for subsets of a table. This indexing method contains only tuples that meet condition expressions.
Expression index	Refers to an index that is built on a function or an expression calculated based on one or more attributes of a table. An expression index works only when the queried expression is the same as the created expression.

- Run the following command to create an ordinary table:
  ```
  CREATE TABLE tpcds.customer_address_bak AS TABLE tpcds.customer_address;
  INSERT 0 0
  ```
- Create a common index.

  You need to query the following information in the **tpcds.customer_address_bak** table:
  ```
  SELECT ca_address_sk FROM tpcds.customer_address_bak WHERE ca_address_sk=14888;
  ```

  Generally, the database system needs to scan the **tpcds.customer_address_bak** table row by row to find all matched tuples. If the size of the **tpcds.customer_address_bak** table is large but only a few (possibly zero or one) of the WHERE conditions are met, the performance of this sequential scan is low. If the database system uses an index to maintain the ca_address_sk attribute, the database system only needs to search a few tree layers for the matched tuples. This greatly improves data query performance. Furthermore, indexes can improve the update and delete operation performance in the database.

  Run the following command to create an index:


```
CREATE INDEX index_wr_returned_date_sk ON tpcds.customer_address_bak (ca_address_sk);
```

- Create a multi-column index.

  Assume you need to frequently query records with **ca_address_sk** being **5050** and **ca_street_number** smaller than **1000** in the **tpcds.customer_address_bak** table. Run the following command:

  ```
  SELECT ca_address_sk,ca_address_id FROM tpcds.customer_address_bak WHERE ca_address_sk = 5050 AND ca_street_number < 1000;
  ```

  Run the following command to define a multiple-column index on **ca_address_sk** and **ca_street_number** columns:

  ```
  CREATE INDEX more_column_index ON
  tpcds.customer_address_bak(ca_address_sk ,ca_street_number );
  ```

- Create a partition index.

  If you only want to find records whose **ca_address_sk** is **5050**, you can create a partial index to facilitate your query.

  ```
  CREATE INDEX part_index ON tpcds.customer_address_bak(ca_address_sk) WHERE ca_address_sk = 5050;
  ```

- Create an expression index.

  Assume you need to frequently query records with **ca_street_number** smaller than **1000**, run the following command:

  ```
  SELECT * FROM tpcds.customer_address_bak WHERE trunc(ca_street_number) < 1000;
  ```

  The following expression index can be created for this query task:

  ```
  CREATE INDEX para_index ON tpcds.customer_address_bak (trunc(ca_street_number));
  ```

- Delete the **tpcds.customer_address_bak** table.

  ```
  DROP TABLE tpcds.customer_address_bak;
  ```


10 Creating and Managing Views

Background

If some columns in one or more tables in a database are frequently searched for, an administrator can define a view for these columns, and then users can directly access these columns in the view without entering search criteria.

A view is different from a basic table. It is only a virtual object rather than a physical one. A database only stores the definition of a view and does not store its data. The data is still stored in the original base table. If data in the base table changes, the data in the view changes accordingly. In this sense, a view is like a window through which users can know their interested data and data changes in the database. A view is triggered every time it is referenced.

Managing a View

- Creating a view

  Run the following command to create MyView:

  ```
  CREATE OR REPLACE VIEW MyView AS SELECT * FROM tpcds.web_returns WHERE
  trunc(wr_refunded_cash) > 10000;
  ```

  NOTE

  OR REPLACE in **CREATE VIEW** is optional. The parameter **OR REPLACE** is specified to redefine an existing view.

- Query a view.

  Query the *MyView* view. Real-time data will be returned.

  ```
  SELECT * FROM MyView;
  ```

- Run the following command to query the views in the current user:

  ```
  SELECT * FROM user_views;
  ```

- Run the following command to query all views:

  ```
  SELECT * FROM dba_views;
  ```

- View details about a specified view.

  Run the following command to view details about the dba_users view:

  ```
  \d+ dba_users
                        View "PG_CATALOG.DBA_USERS"
    Column  |         Type          | Modifiers | Storage  | Description
  ----------+-----------------------+-----------+----------+-------------
   USERNAME | CHARACTER VARYING(64) |           | extended |
  View definition:
  ```



```
SELECT PG_AUTHID.ROLNAME::CHARACTER VARYING(64) AS USERNAME
  FROM PG_AUTHID;
```

- Rebuild a view.

  Run the following command to rebuild a view without entering a query statement:

  ```
  ALTER VIEW MyView REBUILD;
  ```

- Delete a view

  Run the following command to delete MyView:

  ```
  DROP VIEW MyView;
  ```


11 Creating and Managing Sequences

Context

A sequence is a database object that generates unique integers. The values of a sequence are integers that automatically increase according to a certain rule. Sequences generate unique values because they increase automatically. This is why sequence numbers are often used as the primary keys.

You can create a sequence for a column in either of the following methods:

- Set the data type of a column to sequence integer. A sequence will be automatically created by the database for this column.
- Run the CREATE SEQUENCE statement to create a sequence. Set the initial value of the **nextval**('*sequence_name*') function to the default value of a column.

Procedure

Method 1: Set the data type of a column to a sequence integer. For example:

```
CREATE TABLE T1
(
   id    serial,
   name  text
);
```

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

```
CREATE TABLE
```

Method 2: Create a sequence and set the initial value of the **nextval**('*sequence_name*') function to the default value of a column. You can cache a specific number of sequence values to reduce the requests to the GTM, improving the performance.

1. Create a sequence.

   ```
   CREATE SEQUENCE seq1 cache 100;
   ```

   If the following information is displayed, the sequence has been created:

   ```
   CREATE SEQUENCE
   ```

2. Set the initial value of the **nextval**('*sequence_name*') function to the default value of a column.

   ```
   CREATE TABLE T2
   (
   ```



```
    id   int not null default nextval('seq1'),
    name text
);
```

If the following information is displayed, the initial value of the function has been set:

```
CREATE TABLE
```

3. Associate the sequence with a column.

   Associate the sequence with a specified column in a table. The sequence will be deleted when you delete its associated field or the table where the field belongs.

   ```
   ALTER SEQUENCE seq1 OWNED BY T2.id;
   ```

   If the following information is displayed, the owner has been set:

   ```
   ALTER SEQUENCE
   ```

4. Query the sequence.

   ```
   SELECT * FROM ALL_SEQUENCES WHERE sequence_name = 'seq1';
    sequence_owner | sequence_name | min_value |       max_value       | increment_by | cycle_flag
   ----------------+---------------+-----------+-----------------------+--------------+------------
    dbadmin        | seq1          |         1 | 9223372036854775807 |            1 | n
   (1 row)
   ```

NOTE

Methods 1 and 2 are similar except that method 2 specifies cache for the sequence. A sequence using cache has holes (non-consecutive values, for example, 1, 4, 5) and cannot keep the order of the values. After a sequence is deleted, its sub-sequences will be deleted automatically. A sequence shared by multiple columns is not forbidden in a database, but you are not advised to do that.

Currently, the preceding two methods cannot be used for existing tables.

Precautions

Sequence values are generated by the GTM. By default, each request for a sequence value is sent to the GTM. The GTM calculates the result of the current value plus the step and then returns the result. The GTM is the only node that can generate sequence values and probably becomes the performance bottleneck. Therefore, you are not advised to use sequences when sequence values need to be generated frequently (for example, using BulkLoad to import data). For example, the **INSERT FROM SELECT** statement has poor performance in the following scenario:

```
CREATE SEQUENCE newSeq1;
CREATE TABLE newT1
        (
          id   int not null default nextval('newSeq1'),
          name text
        );
INSERT INTO newT1(name) SELECT name from T1;
```

To improve the performance, run the following statements (assume that data of 10,000 rows will be imported from *T1* to *newT1*):

```
INSERT INTO newT1(id, name) SELECT id,name from T1;
SELECT SETVAL('newSeq1',10000);
```


> **NOTE**
>
> Rollback is not supported by sequence functions, including nextval() and setval(). The value of the setval function immediately takes effect on nextval in the current session in any cases and take effect in other sessions only when no cache is specified for them. If cache is specified for a session, it takes effect only after all the cached values have been used. To avoid duplicate values, use setval only when necessary. Do not set it to an existing sequence value or a cached sequence value.

If BulkLoad is used, set sufficient cache for *newSeq1* and do not set **Maxvalue** or **Minvalue**. To improve the performance, database may push down the invocation of **nextval**('*sequence_name*') to DNs. Currently, the concurrent connection requests that can be processed by the GTM are limited. If there are too many DNs, a large number of concurrent connection requests will be sent to the GTM. In this case, you need to limit the concurrent connection of BulkLoad to save the GTM connection resources. If the target table is a replication table (**DISTRIBUTE BY REPLICATION**), pushdown cannot be performed. When the data volume is large, this will be a disaster for the database. In addition, the database space may be exhausted. After the import is complete, do **VACUUM FULL**. Therefore, you are not advised to use sequences when BulkLoad is used.

After a sequence is created, a single-row table is maintained on each node to store the sequence definition and value, which is obtained from the last interaction with the GTM rather than updated in real time. The single-row table on a node does not update when other nodes request a new value from the GTM or when the sequence is modified using **setval**.

12 Creating and Managing Scheduled Tasks

Context

When a customer executes some time-consuming tasks during the day time, (for example, statistics summary task or other database synchronization tasks), the service performance will be influenced. So customers execute tasks on database during night time, increasing the workload. The scheduled task function of the database is compatible with the Oracle database scheduled task function that customers can create scheduled tasks. When the scheduled task time arrives, the task will be triggered. Therefore, the workload of OM has been reduced.

Database complies with the Oracle scheduled task function using the DBMS.JOB interface, which can be used to create scheduled tasks, execute tasks automatically, delete a task, and modify task attributes(including task ID, enable/disable a task, the task triggering time/interval and task contents).

NOTE

The hybrid data warehouse (standalone) does not support scheduled tasks.

Periodic Task Management

Step 1 Creates a test table.

```
CREATE TABLE test(id int, time date);
```

If the following information is displayed, the table has been created.

```
CREATE TABLE
```

Step 2 Create the customized storage procedure.

```
CREATE OR REPLACE PROCEDURE PRC_JOB_1()
AS
N_NUM integer :=1;
BEGIN
FOR I IN 1..1000 LOOP
INSERT INTO test VALUES(I,SYSDATE);
END LOOP;
END;
/
```

If the following information is displayed, the procedure has been created.


```
CREATE PROCEDURE
```

Step 3 Create a task.

- Create a task with unspecified **job_id** and execute the **PRC_JOB_1** storage procedure every two minutes.

```
call dbms_job.submit('call public.prc_job_1(); ', sysdate, 'interval ''1 minute''', :a);
job
-----
1
(1 row)
```

- Create task with specified job_id.

```
call dbms_job.isubmit(2,'call public.prc_job_1(); ', sysdate, 'interval ''1 minute''');
isubmit
---------


(1 row)
```

Step 4 View the created task information about the current user in the **user_jobs** view.

USER_JOBS displays all jobs owned by the user. It is accessible only to users with system administrator rights.

```
select job,dbname,start_date,last_date,this_date,next_date,broken,status,interval,failures,what from
user_jobs;
job | dbname |     start_date      |         last_date          |         this_date          |      next_date      | broken |
status |      interval       | failures |          what
-----+--------+---------------------+----------------------------+----------------------------+---------------------
+--------+--------+--------------------+----------+--------------------------
1 | gaussdb   | 2017-07-18 11:38:03 | 2017-07-18 13:53:03.607838 | 2017-07-18 13:53:03.607838 |
2017-07-18 13:54:03 | n      | s      | interval '1 minute' |         0 | call public.prc_job_1();
(1 row)
```

Table 12-1 USER_JOBS columns

Name	Type	Description
job	int4	Job ID
log_user	name not null	User name of the job creator
priv_user	name not null	User name of the job executor
dbname	name not null	Database in which the job is created
start_date	timestamp without time zone	Job start time
start_suc	text	Start time of the successful job execution
last_date	timestamp without time zone	Start time of the last job execution
last_suc	text	Start time of the last successful job execution

Name	Type	Description
this_date	timestamp without time zone	Start time of the ongoing job execution
this suc	text	Same as **THIS_DATE**
next_date	timestamp without time zone	Schedule time of the next job execution
next suc	text	Same as **next_date**
broken	text	Task status **Y**: the system does not try to execute the task. **N**: the system attempts to execute the task.
status	char	Status of the current job. The value range is **'r', 's', 'f', 'd'**. The default value is **'s'**. The indications are as follows: ● **r**: running ● **s**: finished ● **f**: failed ● **d**: aborted
interval	text	Time expression used to calculate the next execution time. If this parameter is set to **null**, the job will be executed once only.
failures	smallint	Number of consecutive failures.
what	text	Body of the PL/SQL blocks or anonymous clock that the job executes

Step 5 Stop a task.

```
call dbms_job.broken(1,true);
broken
--------

(1 row)
```

Step 6 Start a task.

```
call dbms_job.broken(1,false);
broken
--------

(1 row)
```

Step 7 Modify attributes of a task.

- Modify the **Next_date** parameter information about a task.

  -- Specify the task of modifying **Next_date** of **Job1** will be executed in one hour.

  ```
  call dbms_job.next_date(1, sysdate+1.0/24);
  next_date
  ```



```
-----------

(1 row)
```

- Modify the **Interval** parameter information of a task.

  -- Set **Interval** of **Job1** to **1**.

  ```
  call dbms_job.interval(1,'sysdate + 1.0/24');
  interval
  ----------

  (1 row)
  ```

- Modify the **What** parameter information of a **JOB**.

  -- Change **What** to the SQL statement **insert into public.test values(333, sysdate+5);** for **Job1**.

  ```
  call dbms_job.what(1,'insert into public.test values(333, sysdate+5);');
  what
  ------

  (1 row)
  ```

- Modify **Next_date**, **Interval**, and **What** parameter information of **JOB**.

  ```
  call dbms_job.change(1, 'call public.prc_job_1();', sysdate, 'interval ''1 minute''');
  change
  --------

  (1 row)
  ```

Step 8 Delete a **JOB**.

```
call dbms_job.remove(1);
remove
--------

(1 row)
```

Step 9 Set JOB permissions.

- During the creation of a job, the job is bound to the user and database that created the job. Accordingly, the user and database are added to **dbname** and **log_user** columns in the **pg_job** system view, respectively.
- If the current user is a DBA user, system administrator, or the user who created the job (**log_user** in **pg_job**), the user has the permissions to delete or modify parameter settings of the job using the remove, change, next_data, what, or interval interface. Otherwise, the system displays a message indicating that the current user has no permission to perform operations on the JOB.
- If the current database is the one that created a job, (that is, **dbname** in **pg_job**), you can delete or modify parameter settings of the job using the remove, change, next_data, what, or interval interface.
- When deleting the database that created a job, (that is, **dbname** in **pg_job**), the system associatively deletes the job records of the database.
- When deleting the user who created a job, (that is, **log_user** in **pg_job**), the system associatively deletes the job records of the user.

----End

