ApsaraDB HybridDB for PostgreSQL

Quick Start

Quick Start

ApsaraDB HybridDB for PostgreSQL is a distributed cloud database that is composed of multiple groups to provide MPP (Massively Parallel Processing) data warehousing service. HybridDB for PostgreSQL is developed based on the Greenplum Open Source Database program and is enhanced with some in-depth extensions by Alibaba Cloud.

HybridDB for PostgreSQL is compatible with the Greenplum environment and supports features including OSS storage, JSON data type, and HyperLogLog approximating analysis. For details about HybridDB features and limitations, see Features and limitations.

To use HybridDB for PostgreSQL, you need to complete the following tasks:

1. Create an instance.
2. Set up an instance, including setting up a whitelist, setting up an account, and setting the network type.
3. Connect to a database.
4. Import data. You can select to import and export data in parallel by using OSS external tables, or to import data from MySQL, from PostgreSQL, or by using the COPY command.

You can create or purchase HybridDB for PostgreSQL instances by using one of the following methods:

Create an instance on the HybridDB for PostgreSQL console.

Purchase an instance on the HybridDB for PostgreSQL Purchase Page.

This document describes the detailed steps for creating a HybridDB for PostgreSQL instance on the HybridDB for PostgreSQL console to facilitate your operations for adding or removing instances using the console.

Billing method

Now, HybridDB for PostgreSQL only supports the **Pay-As-You-Go** method.

Prerequisites

You have registered an account. If you haven’ t signed up, go to the Official Website to sign up.

Procedure

Log on to the HybridDB for PostgreSQL console.

Click **Create Instance** at the top right corner of the page.

Select the instance configuration. Details of various configuration items are as follows:

Region and zone: for guidance on how to select, see Regions and zones.

Engine: the database type. Only supports **Storage Included**.

Instance Class: the instance type. It is the unit of computing resources. Different classes have different storage spaces and computing capabilities. For details, see Instance types.

Instance Groups: the number of purchased instances. The minimum is 2. Increasing the number of groups can result in a linear performance increase.

Confirm your order information on the right-side calculation pane, and then click **Buy Now**.

Click **Activate** to confirm to activate the instance.

Go to the **Instance List** page of HybridDB for PostgreSQL console to view the newly created instance.

Set up instances

You must set up the whitelist before starting an instance. Add IP addresses or IP segments that are allowed to access a database to the whitelist to ensure security and stability.

Background

There are three scenarios for accessing HybridDB for PostgreSQL databases:

Access from the Internet.

Access from the intranet. The network types of HybridDB for PostgreSQL and ECS instances must be identical.

Access from the intranet and Internet at the same time. The network types of HybridDB for PostgreSQL and ECS instances must be identical, and the access mode must be **Safe Connection Mode**.

Note:

- To set the network type, see Set the Network Type.

Procedure

Log on to the HybridDB for PostgreSQL console.

Select the region where the target instance is located.

Click the ID of the instance to go to the **Basic Information** page of the instance.

In the left-side navigation pane, click **Security Controls**.

In the **White List Settings** page, click **Modify** under the default whitelist group to go to the **Modify Group** page.

> **Note:** You can also click **Clear** under the default whitelist group to clear the IP addresses included, and then click **Add White List Group** to create a custom group.

Delete the default address 127.0.0.1 from the whitelist and then enter a custom whitelist. Parameters are described as follows:

Group Name: The group name contains 2 to 32 characters which consist of lowercase letters, numbers or underscores (_). The group name must start with a lowercase letter and end with a letter or number. The default group name cannot be modified or deleted.

White List: Enter the IP addresses or IP segments that are allowed to access the database. IP addresses or IP segments are separated by commas (,).

The whitelist can contain IP addresses (for example, 10.10.10.1) or IP segments (for example, 10.10.10.0/24, which indicates that any IP address in the format of 10.10.10.X can access the database).

% or 0.0.0.0/0 indicates that any IP address is allowed to access the database.

> **Note:** This configuration will greatly reduce database security, and thus is not recommended unless necessary.

After an instance is created, the local loopback IP address 127.0.0.1 is added to the default whitelist, which prevents all external IP addresses from accessing the instance.

Choose an existing ECS IP Address: Click it to display all the ECS instances belonging to the same account. You can select ECS IP addresses to add the ECS instances to the whitelist.

Click **OK** to add the whitelist.

Next

Correct use of the whitelist can provide more advanced access security protection for HybridDB for PostgreSQL. For this reason, we recommend that you maintain the whitelist on a regular basis.

During the subsequent operations, you can click **Modify** under the group name to modify an existing group, or click **Delete** to delete an existing custom group.

This document describes how to create an account and reset the password for a HybridDB for PostgreSQL instance.

Create an account

Prior to using a HybridDB for PostgreSQL instance, you need to create an account for the database.

Procedure

Log on to the HybridDB for PostgreSQL console.

Select the region where the target instance is located.

Click the ID of the instance to go to the **Basic Information** page of the instance.

Click **Account Management** in the left-side navigation pane.

Click **Create Account**.

Enter the database account and password and then click **OK**.

Database Account: 2 to 16 characters, and can contain lowercase letters, numbers or underscores (_). It must begin with a letter and end with a letter or a number, for example, user4example*.

Password: 8 to 32 characters. It must contain at least three types of the following characters: uppercase letters, lowercase letters, numbers, or special characters.

Confirm Password: Enter the password again.

Reset account password

When using HybridDB for PostgreSQL, if you forget the password of the database account, you can reset the password in the HybridDB for PostgreSQL console.

Note: We recommend that you change the password on a regular basis for data security considerations.

Procedure

Log on to the HybridDB for PostgreSQL console.

Select the region where the target instance is located.

Click **Manage** under the **Action** column of the target instance to go to the **Basic Information** page of the instance.

Click **Account Management** in the left-side navigation pane.

Click **Reset password** under the account to be managed.

Enter and confirm the new password and then click **OK**.

> **Note:** The password is composed of 8 to 32 characters. It must contain at least three types of the following characters: uppercase letters, lowercase letters, numbers, or special characters. A password that has been previously used is not allowed.

Alibaba Cloud ApsaraDB supports two network types: classic network and Virtual Private Cloud (VPC). By default, HybridDB for PostgreSQL uses the classic network. If you want to use VPC, ensure that the HybridDB for PostgreSQL instance and the VPC are in the same region.

This document mainly describes the differences between the two network types and how to configure the settings.

Background

On the Alibaba Cloud platform, the classic network and VPC have the following differences:

Classic network: The cloud service on a classic network is not isolated, and unauthorized access can only be blocked by the whitelist policy of the cloud service.

Virtual Private Cloud (VPC): VPC helps you build an isolated network environment on the Alibaba Cloud. You can customize the routing table, IP address range and gateway in the VPC. You can also combine your IDC and cloud resources on the Alibaba Cloud VPC into a virtual IDC using a leased line or VPN to seamlessly migrate applications to the cloud.

Procedure

Create a VPC in the same region with the target HybridDB for PostgreSQL instance. For detailed steps, see Create a VPC.

Log on to the HybridDB for PostgreSQL console.

Select the region where the target instance is located.

Click the ID of the instance to go to the **Basic Information** page of the instance.

Click **Database Connection** in the instance menu bar.

Click **Switch to VPC**.

Select a VPC and virtual switch, and then click **OK**.

> **Note:** After the network is switched to VPC, the original Intranet address will be changed from a classic network address to a VPC address. ECS on the classic network won't be able to access the HybridDB for PostgreSQL instance. The original Internet address remains unchanged.

Greenplum Database is developed based on PostgreSQL 8.2 branch and is fully compatible with the message protocols of PostgreSQL 8.2. HybridDB for PostgreSQL is also based on PostgreSQL 8.2. Therefore, when working with HybridDB for PostgreSQL, you can directly use tools that support the message protocols of PostgreSQL 8.2, such as libpq, JDBC, ODBC, psycopg2, and pgAdmin III.

HybridDB for PostgreSQL provides a binary psql program for Redhat platform, which you can download from Client tools. The official website of Greenplum also provides an installer containing JDBC, ODBC, and libpq to facilitate the installation and use.

psql

Psql is a common tool of Greenplum. It provides a variety of commands and its binary files are located in the BIN directory after Greenplum installation. The steps for using the tool are as follows:

Use one of the following methods to connect:

Concatenate strings

```
 psql "host=yourgpdbaddress.gpdb.rds.aliyuncs.com port=3568 dbname=postgres
user=gpdbaccount password=gpdbpassword"

postgres=> select version();
version
----------------------------------------------------------------------------------------------
PostgreSQL 8.3devel (Greenplum Database 4.3.99.00 build dev) compiled on Jun 26 2016
23:44:59
(1 row)
```

Specify parameters

```
psql  -h yourgpdbaddress.gpdb.rds.aliyuncs.com -p 3568 -d postgres -U gpdbaccount
```

Enter the password to go to the psql shell interface.

```
 postgres=> select version();
version
------------------------------------------------------------------------------------------------
PostgreSQL 8.3devel (Greenplum Database 4.3.99.00 build dev) compiled on Jun 26 2016 23:44:59
(1 row)
```

Parameter descriptions:

- -h: specifies the host address.
- -p: specifies the port number.
- -d: specifies the database engine (the default engine is postgres).
- -U: specifies the connected user.

You can view more options from psql-- help. By performing \?, you can view more supported psql commands.

Reference

For more usage descriptions of Greenplum psql, see psql.

You can also use the PostgreSQL psql command, but do note the difference in usage details. For details, see PostgreSQL 8.3.23 Documentation — psql.

pgAdmin III

pgAdmin III is a GUI client of PostgreSQL, which can be used to directly connect to HybridDB for PostgreSQL. For details, see the paAdmin official page.

You can download pgAdmin III 1.6.3 from the PostgreSQL official website . pgAdmin III 1.6.3 supports a variety of platforms, such as Windows, MacOS, and Linux. For other GUI clients, see Client tools.

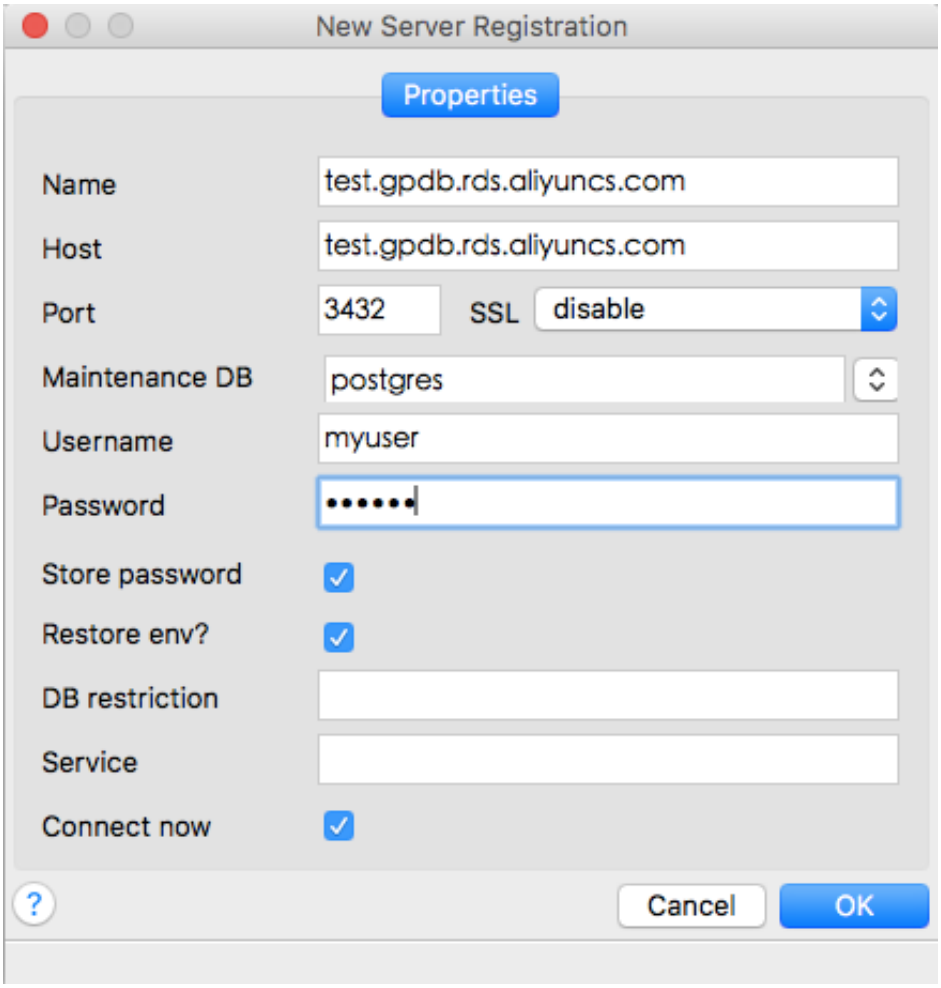
> **Note:** HybridDB for PostgreSQL is compatible with PostgreSQL 8.2. Therefore, you need to use pgAdmin III 1.6.3 or earlier versions to connect to HybridDB for PostgreSQL. (pgAdmin 4 is not supported.)

Procedure

Download and install pgAdmin III 1.6.3 or an earlier version.

Select **File** > **Add Server** to go to the **New Server Registration** page.

Enter the **Properties** as shown in the following picture:



Click **OK** to connect to the HybridDB for PostgreSQL database.

JDBC

The HybridDB for PostgreSQL JDB interface is the official PostgreSQL JDBC driver. To use the JDBC Driver, select one of the following methods:

Download it from PostgreSQL JDBC Driver and add the downloaded JDBC to the environment variables.

Use the tool package provided by the Greenplum official website. For details, see

Greenplum Database 4.3 Connectivity Tools for UNIX.

Code case:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class gp_conn {

public static void main(String[] args) {
try {
Class.forName("org.postgresql.Driver");
Connection db =
DriverManager.getConnection("jdbc:postgresql://mygpdbpub.gpdb.rds.aliyuncs.com:3568/postgres","mygpdb","my
gpdb");

Statement st = db.createStatement();
ResultSet rs = st.executeQuery("select * from gp_segment_configuration;");
while (rs.next()) {
System.out.print(rs.getString(1));
System.out.print(" | ");
System.out.print(rs.getString(2));
System.out.print(" | ");
System.out.print(rs.getString(3));
System.out.print(" | ");
System.out.print(rs.getString(4));
System.out.print(" | ");
System.out.print(rs.getString(5));
System.out.print(" | ");
System.out.print(rs.getString(6));
System.out.print(" | ");
System.out.print(rs.getString(7));
System.out.print(" | ");
System.out.print(rs.getString(8));
System.out.print(" | ");
System.out.print(rs.getString(9));
System.out.print(" | ");
System.out.print(rs.getString(10));
System.out.print(" | ");
System.out.println(rs.getString(11));
}
rs.close();
st.close();
} catch (ClassNotFoundException e) {
e.printStackTrace();
} catch (SQLException e) {
e.printStackTrace();
}
}
}
```

For more detailed documentation, see The PostgreSQL JDBC Interface.

Python

Python uses the psycopg2 library to connect to Greenplum and PostgreSQL. The procedure for using the tool is shown as follows:

Install psycopg2. In CentOS, the available methods include:

Perform yum -y install python-psycopg2

Perform pip install psycopg2

Install from the following source code.

```
 yum install -y postgresql-devel*
wget http://initd.org/psycopg/tarballs/PSYCOPG-2-6/psycopg2-2.6.tar.gz
tar xf psycopg2-2.6.tar.gz
cd psycopg2-2.6
python setup.py build
sudo python setup.py install
```

After the installation, set the PYTHONPATH and then you can use it, such as:

```
 import psycopg2
sql = 'select * from gp_segment_configuration;'
conn = psycopg2.connect(database='gpdb', user='mygpdb', password='mygpdb',
host='mygpdbpub.gpdb.rds.aliyuncs.com', port=3568)
conn.autocommit = True
cursor = conn.cursor()
cursor.execute(sql)
rows = cursor.fetchall()
for row in rows:
print row
conn.commit()
conn.close()
```

You will get a result similar to the following:

```
(1, -1, 'p', 'p', 's', 'u', 3022, '192.168.2.158', '192.168.2.158', None, None)
(6, -1, 'm', 'm', 's', 'u', 3019, '192.168.2.47', '192.168.2.47', None, None)
(2, 0, 'p', 'p', 's', 'u', 3025, '192.168.2.148', '192.168.2.148', 3525, None)
(4, 0, 'm', 'm', 's', 'u', 3024, '192.168.2.158', '192.168.2.158', 3524, None)
(3, 1, 'p', 'p', 's', 'u', 3023, '192.168.2.158', '192.168.2.158', 3523, None)
(5, 1, 'm', 'm', 's', 'u', 3026, '192.168.2.148', '192.168.2.148', 3526, None)
```

libpq

Libpq is the C language interface of PostgreSQL database. You can access a PostgreSQL database in a C program through libpq for database operations. After Greenplum or PostgreSQL has been installed, you can find its static and dynamic libraries under the lib directory.

- For libpq details, see PostgreSQL 9.4 Documentation - Chapter 31. libpq - C Library.
- For related cases, see libpq Example Programs.

ODBC

PostgreSQL ODBC is an open-source version based on the LGPL (GNU Lesser General Public License) protocol. You can download it from psqlODBC - PostgreSQL ODBC driver.

Procedure

1. Install the driver.

```
yum install -y unixODBC.x86_64
yum install -y postgresql-odbc.x86_64
```

2. Check the driver’ s configuration.

```
cat /etc/odbcinst.ini
# Example driver definitions

# Driver from the postgresql-odbc package
# Setup from the unixODBC package
[PostgreSQL]
Description = ODBC for PostgreSQL
Driver = /usr/lib/psqlodbcw.so
Setup = /usr/lib/libodbcpsqlS.so
Driver64 = /usr/lib64/psqlodbcw.so
Setup64 = /usr/lib64/libodbcpsqlS.so
FileUsage = 1

# Driver from the mysql-connector-odbc package
# Setup from the unixODBC package
[MySQL]
Description = ODBC for MySQL
Driver = /usr/lib/libmyodbc5.so
Setup = /usr/lib/libodbcmyS.so
Driver64 = /usr/lib64/libmyodbc5.so
Setup64 = /usr/lib64/libodbcmyS.so
FileUsage = 1
```

Configure DSN as the following codes. Change **** in the codes to the actual connection information.

```
[mygpdb]
Description = Test to gp
Driver = PostgreSQL
Database = ****
Servername = ****.gpdb.rds.aliyuncs.com
UserName = ****
Password = ****
Port = ****
ReadOnly = 0
```

Test connectivity.

```
echo "select count(*) from pg_class" | isql mygpdb
+---------------------------------------+
| Connected! |
| |
| sql-statement |
| help [tablename] |
| quit |
| |
+---------------------------------------+
SQL> select count(*) from pg_class
+----------------------+
| count |
+----------------------+
| 388 |
+----------------------+
SQLRowCount returns 1
1 rows fetched
```

After ODBC has connected to the instance, connect applications to ODBC. For details, see PostgreSQL ODBC driver and psqlODBC HOWTO - C#.

Reference

- Pivotal Greenplum Official Documentation
- PostgreSQL psql ODBC
- PostgreSQL ODBC Compilation
- Greenplum ODBC Download
- Greenplum JDBC Download

Import data

HybridDB for PostgreSQL supports importing data to or exporting data from OSS in parallel through OSS external tables (that is, the gpossext feature), and supports compressing OSS external table files through gzip to reduce storage space and cost.

Create an extension for OSS external tables (oss_ext)

Before using OSS external tables, you need to create an oss_ext extension first (Each database needs to create an oss_ext separately).

- To create an oss_ext, run the command: CREATE EXTENSION IF NOT EXISTS oss_ext;
- To delete an oss_ext, run the command: DROP EXTENSION IF EXISTS oss_ext;

Import data in parallel

Perform the following operations to import data:

Distribute and store the data evenly among multiple OSS files. The number of files is preferably an integral multiple of the number of HybridDB for PostgreSQL data nodes (number of Segments).

Create a READABLE external table in the HybridDB for PostgreSQL database.

Run the following command to import data in parallel.

INSERT INTO <target table> SELECT * FROM <external table>

Export data in parallel

Perform the following operations to export data:

Create a WRITABLE external table in the HybridDB for PostgreSQL database.

Run the following command to export data to OSS in parallel.

INSERT INTO <external table> SELECT * FROM <source table>

Create syntax for OSS external tables

Run the following command to create syntax for OSS external tables:

```
CREATE [READABLE] EXTERNAL TABLE tablename
( columnname datatype [, ...] | LIKE othertable )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
[( [HEADER]
[DELIMITER [AS] 'delimiter' | 'OFF']
[NULL [AS] 'null string']
[ESCAPE [AS] 'escape' | 'OFF']
[NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
[FILL MISSING FIELDS] )]
| 'CSV'
[( [HEADER]
[QUOTE [AS] 'quote']
[DELIMITER [AS] 'delimiter']
[NULL [AS] 'null string']
[FORCE NOT NULL column [, ...]]
[ESCAPE [AS] 'escape']
[NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
[FILL MISSING FIELDS] )]
[ ENCODING 'encoding' ]
[ [LOG ERRORS [INTO error_table]] SEGMENT REJECT LIMIT count
[ROWS | PERCENT] ]

CREATE WRITABLE EXTERNAL TABLE table_name
( column_name data_type [, ...] | LIKE other_table )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
[( [DELIMITER [AS] 'delimiter']
[NULL [AS] 'null string']
[ESCAPE [AS] 'escape' | 'OFF'] )]
| 'CSV'
[([QUOTE [AS] 'quote']
[DELIMITER [AS] 'delimiter']
[NULL [AS] 'null string']
[FORCE QUOTE column [, ...]] ]
[ESCAPE [AS] 'escape'] )]
[ ENCODING 'encoding' ]
[ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]

ossprotocol:
oss://oss_endpoint prefix=prefix_name
id=userossid key=userosskey bucket=ossbucket compressiontype=[none|gzip] async=[true|false]

ossprotocol:
oss://oss_endpoint dir=[folder/[folder/]...]/file_name
id=userossid key=userosskey bucket=ossbucket compressiontype=[none|gzip] async=[true|false]

ossprotocol:
oss://oss_endpoint filepath=[folder/[folder/]...]/file_name
id=userossid key=userosskey bucket=ossbucket compressiontype=[none|gzip] async=[true|false]
```

Parameters description

Common parameters

Protocol and endpoint: The format is protocol name: //oss_endpoint. The “protocol name” is oss, and “oss_endpoint” is the domain name of the corresponding OSS region.

> **Note:** If the access request is from an Alibaba Cloud host, you use the Intranet domain name (that is, with “internal” in the domain name) to avoid incurring public data usage.

id: OSS account ID.

key: OSS account key.

bucket: Specifies the bucket where the data file is located. It is required to be pre-created in OSS.

prefix: Specifies the prefix of the corresponding path name of the data file. The prefix does not support regular expressions and is only a matching prefix. In addition, it is mutually exclusive with filepath and dir. That is, you cannot specify them at the same time.

If you create a READABLE external table for data import, all the OSS files with this prefix will be imported.

If you have specified prefix=test/filename, all the following files will be imported:

- test/filename
- test/filenamexxx
- test/filename/aa
- test/filenameyyy/aa
- test/filenameyyy/bb/aa

If you have specified prefix=test/filename/, only the following files will be imported (other preceding files listed won’t be imported):

- test/filename/aa

If you create a WRITABLE external table for data export, a unique file name will be generated automatically based on the prefix to name the exported file.

> **Note:** There will be more than one exported file. Every data node will export one or more files. The exported file name format is prefix_tablename_uuid.x. To be specific, the uuid is the generated int64 integer value (time stamp in microsecond), and the x is the node ID. HybridDB for PostgreSQL supports multiple export operations with the same external table. The exported files from every export operation are differentiated by the UUID, and the exported files in the same export operation share the same UUID.

dir: The path of virtual folders in OSS. It is mutually exclusive with prefix and filepath.

> The folder path ends with "/ ", such as test/mydir/.
>
> If you use this parameter to create an external table during data importing, all the files under the specified virtual directory will be imported, excluding its subdirectories and files under the subdirectories. Unlike filepath, the dir directory has no naming requirements for files under it.
>
> If you use this parameter to create an external table during data exporting, all the data will be exported to the multiple files under this directory. The output file names follow a format of filename.x. To be specific, the x is a number but may be discontinuous.

filepath: The file name that contains a path in OSS. It is mutually exclusive with prefix and dir. You can only specify it at the creation of a READABLE external table (that is, only usable during data import).

> The file name contains the file path, but does not contain the bucket name.
>
> The file naming rule must follow filename or filename.x during data import. x is required to start from 1 and be continuous. For example, if you specify filepath = filename and the OSS contains the following files, the imported files will include filename, filename.1, and filename.2. Because filename.3 does not exist, filename.4 won't be imported.

```
filename
filename.1
filename.2
filename.4,
```

Import mode parameters

async: whether to enable asynchronous mode to load data or not.

- Enabling worker threads to load data from OSS can accelerate the import performance.
- Asynchronous mode is enabled by default, and it consumes more hardware resources than the normal mode. You can use async=false or async=f to disable it.

compressiontype: The compression format of the imported files.

- If specified to none (default value), it indicates that the imported files are not compressed.
- If specified to gzip, it indicates that the imported format is gzip. Only the gzip compression format is supported now.

Export mode parameters

oss_flush_block_size: The buffer size for a single data flushed to OSS, 32 MB by default. The value ranges from 1 MB to 128 MB.

oss_file_max_size: The maximum size of the file written to OSS. Once this limit is exceeded, the export will switch to another file to continue data writing. The value is 1,024 MB by default and ranges from 8 MB to 4,000 MB.

In addition, pay attention to the following items for the export mode:

WRITABLE is the key word of the external table in the export mode. It must be explicitly specified when you create an external table.

Now the export mode only supports prefix and dir parameter modes, and does not support filepath.

The DISTRIBUTED BY clause in the export mode enables the data node (Segment) to write data to OSS by the specified distribution key.

Other general parameters

There are also the following fault-tolerant parameters for the import and export modes:

oss_connect_timeout: Sets the connection timeout. The unit is second and the default value is 10 seconds.

oss_dns_cache_timeout: Sets the DNS timeout. The unit is second and the default value is 60 seconds.

oss_speed_limit: Controls the minimum tolerable rate. The default value is 1,024 (that is, 1 K).

oss_speed_time: Controls the maximum tolerable time. The default value is 15 seconds.

With all the preceding parameters set as default, timeout will be triggered when the transmission speed is slower than 1 K for 15 consecutive seconds. For details, see OSS SDK error handling.

All other parameters are compatible with the legacy syntax of Greenplum EXTERNAL TABLE. For specific syntax explanations, see Greenplum External Table Syntax Official Documentation. Such parameters mainly include:

- FORMAT: The supported file formats, including text and csv.
- ENCODING: The encoding format of the data in the file, such as UTF-8.
- LOG ERRORS: Specifies that the clause can ignore erroneous data during the import and write the data into error_table. You can also specify the error reporting threshold using the count parameter.

Examples

```
# Create an external table for OSS import
create readable external table ossexample
(date text, time text, open float, high float,
low float, volume int)
location('oss://oss-cn-hangzhou.aliyuncs.com
prefix=osstest/example id=XXX
key=XXX bucket=testbucket' compressiontype=gzip)
FORMAT 'csv' (QUOTE ''' DELIMITER E'\t')
ENCODING 'utf8'
LOG ERRORS INTO my_error_rows SEGMENT REJECT LIMIT 5;
create readable external table ossexample
(date text, time text, open float, high float,
low float, volume int)
location('oss://oss-cn-hangzhou.aliyuncs.com
dir=osstest/ id=XXX
key=XXX bucket=testbucket')
FORMAT 'csv'
LOG ERRORS SEGMENT REJECT LIMIT 5;
create readable external table ossexample
(date text, time text, open float, high float,
low float, volume int)
location('oss://oss-cn-hangzhou.aliyuncs.com
filepath=osstest/example.csv id=XXX
key=XXX bucket=testbucket')
FORMAT 'csv'
LOG ERRORS SEGMENT REJECT LIMIT 5;
```

```
# Create an external table for OSS export
create WRITABLE external table ossexample_exp
(date text, time text, open float, high float,
low float, volume int)
location('oss://oss-cn-hangzhou.aliyuncs.com
prefix=osstest/exp/outfromhdb id=XXX
key=XXX bucket=testbucket') FORMAT 'csv'
DISTRIBUTED BY (date);
create WRITABLE external table ossexample_exp
(date text, time text, open float, high float,
low float, volume int)
location('oss://oss-cn-hangzhou.aliyuncs.com
dir=osstest/exp/ id=XXX
key=XXX bucket=testbucket') FORMAT 'csv'
DISTRIBUTED BY (date);

# Create a heap table to load data
create table example
(date text, time text, open float,
high float, low float, volume int)
DISTRIBUTED BY (date);

# Load data from ossexample to example in parallel
insert into example select * from ossexample;

# Export data from example to OSS in parallel
insert into ossexample_exp select * from example;

# We can see from the following query plan that every segment will participate in the work.
# They pull data from the OSS in parallel and then use the Redistribute Motion node to distribute the hashed data
to corresponding segments. Data-receiving segments store the data to the database through the Insert node.
explain insert into example select * from ossexample;
QUERY PLAN
------------------------------------------------------------------------------------------------
Insert (slice0; segments: 4) (rows=250000 width=92)
-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..11000.00 rows=250000 width=92)
Hash Key: ossexample.date
-> External Scan on ossexample (cost=0.00..11000.00 rows=250000 width=92)
(4 rows)

# We can see from the following query plan that the segment exports local data directly to the OSS without data
redistribution.
explain insert into ossexample_exp select * from example;
QUERY PLAN
------------------------------------------------------------------
Insert (slice0; segments: 3) (rows=1 width=92)
-> Seq Scan on example (cost=0.00..0.00 rows=1 width=92)
(2 rows)
```

Attentions

Apart from the location-related parameters, the rest part of the syntax for creating and using

external tables is consistent with that of Greenplum.

The data importing performance is related with the HybridDB for PostgreSQL cluster resources (CPU, IO, memory, network, and so on) and OSS. We recommend that you use compressed column store when creating a table to achieve optimal importing performance. For example, you can specify the clause WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib, COMPRESSLEVEL=5, BLOCKSIZE=1048576). For details, see Greenplum Database Tabulation Syntax Official Documentation.

The ossendpoint region should match the HybridDB for PostgreSQL region to ensure the data importing performance. We recommend that you configure the OSS and HybridDB for PostgreSQL instances in the same region to achieve the optimal performance. For related information, see OSS Endpoint Information.

SDK error handling

In the case of operation errors during data import and export, the error log displays the following information:

code: The HTTP status code of the erroneous request.

error_code: The error code of OSS.

error_msg: The error message of OSS.

req_id: The UUID of the request. If you cannot solve a problem, use the req_id to ask OSS development engineers for help.

For details, see OSS API Error Response. Timeout-related errors can be handled using oss_ext related parameters.

FAQs

The import is too slow. See the import performance descriptions in the preceding Attentions section.

Reference

- OSS Endpoint Information
- OSS Help Page

- OSS SDK Error Handling
- OSS API Error Response
- Greenplum Database External Table Syntax Official Documentation
- Greenplum Database Tabulation Syntax Official Documentation

The mysql2pgsql tool supports migrating tables in MySQL to HybridDB for PostgreSQL, Greenplum Database, PostgreSQL, or PPAS without storing the data separately. The principle of the tool is connecting to the source MySQL database and the target database at the same time, querying and retrieving the data to be exported in the MySQL database, and then importing the data to the target database using the COPY command. The tool supports multithread import (every worker thread is in charge of importing a part of database tables).

Parameter configuration

Modify the “my.cfg” configuration file, and configure the source and target database connection information.

The connection information of the source MySQL database is as follows:

Note: You need to have the read permission on all user tables in the source MySQL database connection information.

```
[src.mysql]
host = "192.168.1.1"
port = "3306"
user = "test"
password = "test"
db = "test"
encodingdir = "share"
encoding = "utf8"
```

The connection information of the target PostgreSQL database (including PostgreSQL, PPAS and HybridDB for PostgreSQL) is as follows:

Note: You need to have the write permission on the target table in the target PostgreSQL database.

```
[desc.pgsql]
connect_string = "host=192.168.1.1 dbname=test port=5888 user=test password=pgsql"
```

Usage discription

The usage of mysql2pgsql is described as follows:

```
./mysql2pgsql -l <tables_list_file> -d -j <number of threads>
```

Parameter descriptions:

-l: Optional parameter, used to specify a text file that contains tables to be synchronized. If this parameter is not specified, all the tables in the database specified in the configuration file will be synchronized. <tables_list_file>is a file name. The file contains tables set to be synchronized and query conditions on the tables. An example of the content format is shown as follows:

```
table1 : select * from table_big where column1 < '2016-08-05'
table2 :
table3
table4: select column1, column2 from tableX where column1 != 10
table5: select * from table_big where column1 >= '2016-08-05'
```

-d: Optional parameter, indicating to only generate the tabulation DDL statement of the target table without performing actual data synchronization.

-j: Optional parameter, specifying the number of threads used for data synchronization. If this parameter is not specified, five threads will be used concurrently.

Typical usage

Full-database migration

The full-database migration operations are shown as follows:

Get the DDL statements of the corresponding table on the target end running the following command:

./mysql2pgsql -d

Create a table on the target based on these DDL statements with the distribution key information added.

Perform the following command to synchronize all tables:

./mysql2pgsql

This command will migrate the data from all MySQL tables in the database specified in the configuration file to the target. Five threads are used during the process (the default thread number is five) to read and import the data from all tables involved.

Partial table migration

The partial table migration operations are shown as follows:

Create a new “tab_list.txt” file and insert the following content:

```
t1
t2 : select * from t2 where c1 > 138888
```

Run the following command to synchronize the specified t1 and t2 tables:

```
./mysql2pgsql -l tab_list.txt
```

Note: For the t2 table, only the data that meets the c1 > 138888 condition is migrated.

Download and instructions

To download the binary installer of mysql2pgsql, click here.

To view the mysql2pgsql source code compilation instructions, click here.

The pgsql2pgsql tool supports migrating tables in HybridDB for PostgreSQL, Greenplum Database, PostgreSQL, or PPAS to HybridDB for PostgreSQL, Greenplum Database, PostgreSQL, or PPAS without storing the data separately.

Supported features

pgsql2pgsql supports the following features:

Full-database migration from PostgreSQL, PPAS, Greenplum Database, or HybridDB for PostgreSQL to PostgreSQL, PPAS, Greenplum Database, or HybridDB for PostgreSQL.

Full-database migration and incremental data migration from PostgreSQL or PPAS (9.4 or later version) to PostgreSQL, or PPAS.

Parameters configuration

Modify the “my.cfg” configuration file, and configure the source and target database connection information.

The connection information of the source PostgreSQL database is shown as follows:

Note: The user is preferably the corresponding database owner in the source PostgreSQL database connection information.

```
[src.pgsql]
connect_string = "host=192.168.1.1 dbname=test port=5888 user=test password=pgsql"
```

The connection information of the local temporary PostgreSQL database is shown as follows:

```
[local.pgsql]
connect_string = "host=192.168.1.1 dbname=test port=5888 user=test2 password=pgsql"
```

The connection information of the target PostgreSQL database is shown as follows:

Note: You need to have the write permission on the target table of the target PostgreSQL database.

```
[desc.pgsql]
connect_string = "host=192.168.1.1 dbname=test port=5888 user=test3 password=pgsql"
```

Note:

If you want to perform incremental data synchronization, the connected source database should have the permission to create replication slots.

PostgreSQL 9.4 and later versions support logic flow replication, so it supports the incremental migration if PostgreSQL serves as the data source. The kernel will support logic flow replication only after you enable the following kernel parameters.

- wal_level = logical
- max_wal_senders = 6
- max_replication_slots = 6

Use pgsql2pgsql

Full-database migration

Run the following command to perform a full-database migration:

```
./pgsql2pgsql
```

The migration program will migrate the table data of all the users in the corresponding PostgreSQL database by default to PostgreSQL.

Status information query

Connect to the local temporary database and you can view the status information in a single migration process. The information is stored in the db_sync_status table, including the start and end time of the full-database migration, the start time of the incremental data migration, and the data situation of incremental synchronization.

Download and instructions

To download the binary installer of mysql2pgsql, click here.

To view the mysql2pgsql source code compilation instructions, click here.

You can directly run the \COPY command to import local text file data to HybridDB for PostgreSQL. The premise is that the local text file must be formatted, such as using commas (,), colons (:) or special symbols as separators.

Note:

Parallel writing of massive data is not available because the \COPY command performs serial data writing through the master node. If you want to parallelly write massive data, use the OSS-based data importing method instead.

> The \COPY command is an action instruction of PostgreSQL. If you use the database instruction COPY rather than \COPY, note that only STDIN is supported and file is not supported. That is because the "root user" does not have the super user permission to perform operations on file format files.

Reference of the \COPY command is as follows:

```
\COPY table [(column [, ...])] FROM {'file' | STDIN}
[ [WITH]
[OIDS]
[HEADER]
[DELIMITER [ AS ] 'delimiter']
[NULL [ AS ] 'null string']
[ESCAPE [ AS ] 'escape' | 'OFF']
[NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
[CSV [QUOTE [ AS ] 'quote']
[FORCE NOT NULL column [, ...]]
[FILL MISSING FIELDS]
[[LOG ERRORS [INTO error_table] [KEEP]
SEGMENT REJECT LIMIT count [ROWS | PERCENT] ]

\COPY {table [(column [, ...])] | (query)} TO {'file' | STDOUT}
[ [WITH]
[OIDS]
[HEADER]
[DELIMITER [ AS ] 'delimiter']
[NULL [ AS ] 'null string']
[ESCAPE [ AS ] 'escape' | 'OFF']
[CSV [QUOTE [ AS ] 'quote']
[FORCE QUOTE column [, ...]] ]
[IGNORE EXTERNAL PARTITIONS ]
```

Note:

- HybridDB for PostgreSQL also supports the running of COPY statements using JDBC which encapsulates the CopyIn method. For detailed usage, see Interface CopyIn.
- For the usage of COPY command, see COPY.

The interface protocol of HybridDB for PostgreSQL is compatible with Greenplum Database community edition and PostgreSQL 8.3. You can connect to a HybridDB for PostgreSQL database using a Greenplum database or a PostgreSQL client. This document describes the available client tools for connection.

GUI client tools

For HybridDB for PostgreSQL, you can directly use the client tools supporting Greenplum, such as SQL Workbench, Navicat Premium, Navicat For PostgreSQL, and pgadmin III (1.6.3).

Command line client psql

RHEL 6/7 and CentOS 6/7 platforms

For RHEL 6/7 (Red Hat Enterprise Linux) and CentOS 6/7 platforms, you can download the tool from the following addresses and unzip the downloaded package to use it.

For RHEL 6 or CentOS 6 platforms, click hybriddb_client_package_el6 to download.

For RHEL 7 or CentOS 7 platforms, click hybriddb_client_package_el7 to download.

Other Linux platforms

For other Linux platforms, the applicable compilation methods for client tools are listed as follows:

To get the source code, the following methods are available:

Get the git directory (ensure that you have installed the git tool).

```
git clone https://github.com/greenplum-db/gpdb.git
cd gpdb
git checkout 5d870156
```

Directly download the code .

```
wget https://github.com/greenplum-db/gpdb/archive/5d87015609abd330c68a5402c1267fc86cbc9e1f.zip
unzip 5d87015609abd330c68a5402c1267fc86cbc9e1f.zip
cd gpdb-5d87015609abd330c68a5402c1267fc86cbc9e1f
```

To compile, the GCC or other compilation tools are required.

```
./configure
make -j32
make install
```

Use psql and pg_dump. The paths of the two tools are listed as follows:

For psql: /usr/local/pgsql/bin/psql

For pg_dump: /usr/local/pgsql/bin/pg_dump

Windows and other platforms

For other client tools for Windows and other platforms, go to Pivotal Greenplum Client for a download link. Note that you will be asked to become a member of Pivotal network to download.