

ApsaraDB HybridDB for PostgreSQL

Quick Start

Quick Start

Overview

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 limits, see [Features and limits](#).

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.

Create an instance

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

- Create an instance in 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 in the console.

Billing method

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

Prerequisites

You have registered an account and signed up.

Procedure

Log on to the HybridDB for PostgreSQL consoleHybridDB for PostgreSQL console.

Click **Create Instance**.

Select the instance configuration. The options include:

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 two. More groups provide higher linear performance.

Confirm your order information, and then click **Buy Now**.

Click **Activate** to activate the instance.

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

Note: The instance initialization takes some time. You can perform subsequent operations only after the instance status becomes Running.

Set up an instance

Set up a whitelist

You must set up the whitelist before starting an instance. Add IP addresses or IP segments that are allowed to access a database 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 **Whitelist 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 Whitelist 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, and consists 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.

Whitelist: 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: We recommend that you not use this configuration unless necessary, because it can greatly reduce database security.

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

The whitelist provides an advanced access protection for HybridDB for PostgreSQL. So, 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.

Set up an account

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

Create an account

Before using a HybridDB for PostgreSQL instance, you must create an account for the database.

Note:

- You cannot delete the initial account after it is created.
- You cannot create other accounts on the console, but you can create them by running SQL statements after logging in to 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: contains 2 to 16 characters, and consists of lowercase letters, numbers, or underscores (_). It must start with a letter and end with a letter or number. For example, user4example*.

Password: contains 8 to 32 characters. It must consist of 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 must consist of 8 to 32 characters and contain at least three types of the following characters: uppercase letters, lowercase letters, numbers, or special characters. A password that is previously used is not allowed.

Set the network type

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

The classic network and VPC have the following differences:

Classic network: The cloud service in 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 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 by 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**.

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 changes from a classic network address to a VPC address. ECS on the classic network can no longer access the HybridDB for PostgreSQL instance. The original Internet address remains unchanged.

Connect to a HybridDB for PostgreSQL database

Cloud Database HybridDB for PostgreSQL is fully compatible with the message protocols of PostgreSQL 8.2 and allows direct access to tools that support the PostgreSQL 8.2 message protocols such as libpq, JDBC, ODBC, psycopg2, pgadmin III, and so on.

Greenplum also provides an installation package that includes JDBC, ODBC, and libpq, which can be easily installed and used by users. For more information please see the [Greenplum official documentation](#).

GUI Tools

HybridDB for PostgreSQL users can use Greenplum-supported graphical client tools directly, such as [SQL Workbench] (<http://www.sql-workbench.net/>), [Navicat Premium] (<https://www.navicat.com/Download/navicat-premium>), [Navicat For PostgreSQL] (<https://www.navicat.com/download/navicat-for-postgresql>), [pgadmin III (1.6.3)] (<https://www.postgresql.com/Org/ftp/pgadmin/pgadmin3/v1.6.3/>) and so on.

The following content takes pgAdmin III as an example to illustrate the use of graphical client tools.

pgAdmin III

pgAdmin III is a GUI client of PostgreSQL, which can be used directly to connect to HybridDB for PostgreSQL. For more information, see [paAdmin official page](#).

You can download pgAdmin III 1.6.3 from [PostgreSQL official website](#) . pgAdmin III 1.6.3 supports a variety of platforms, such as Windows, MacOS, and Linux.

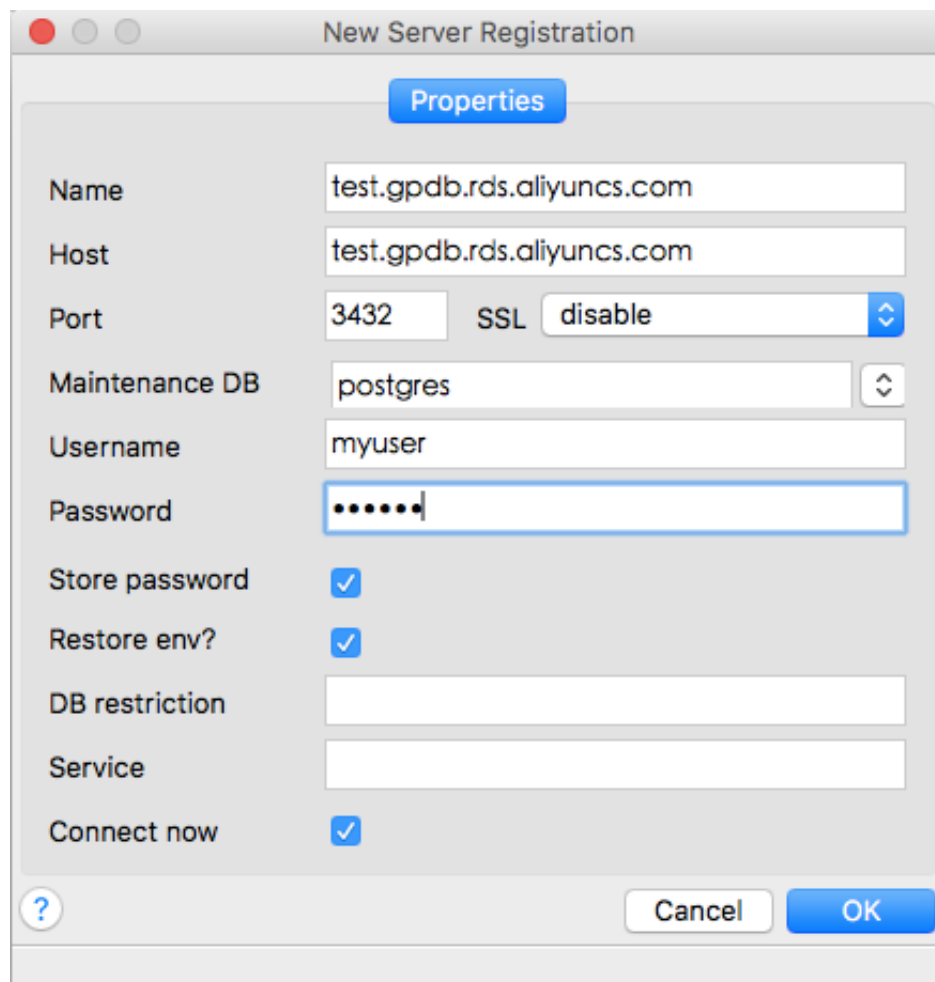
Note: HybridDB for PostgreSQL supports PostgreSQL 8.2 version, and you must use the matching pgAdmin version. The matching version is pgAdmin III 1.6.3 or earlier versions.

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 figure:

A screenshot of a 'New Server Registration' dialog box. It has a title bar with standard window controls. Below the title bar is a blue button labeled 'Properties'. The main area contains several fields: 'Name' and 'Host' both containing 'test.gpdb.rds.aliyuncs.com'; 'Port' set to '3432'; 'SSL' set to 'disable' with a dropdown arrow; 'Maintenance DB' set to 'postgres' with a dropdown arrow; 'Username' set to 'myuser'; 'Password' with a masked field '.....'; 'Store password' checked; 'Restore env?' checked; 'DB restriction' and 'Service' as empty text boxes; and 'Connect now' checked. At the bottom are a help icon (question mark in a circle), a 'Cancel' button, and an 'OK' button.

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

Command Line tools

Users can also use following several command line tools to connect to HybridDB for PostgreSQL instance's database.

psql

psql is a common command line client tool for HybridDB for PostgreSQL. For RHEL (Red Hat Enterprise Linux) 6 or RHEL 7 and CentOS 6 or CentOS 7, Alibaba Cloud provides compressed software packages that can be used directly after decompression. .

For RHEL 6 or CentOS 6 platforms, click **download**.

For RHEL 7 or CentOS 7 platforms, click **download**.

For other Linux platforms, users need to download the source code and use it after compiling and installation. The compiling methods is as follows:

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

Get the git directory directly(make sure that you have installed the git tool).

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

Directly download codes.

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

2. You need the GCC or other compilers to compile the code and install the software.

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

After the installation the path of psql is as follows:

```
psql: `/usr/local/pgsql/bin/psql`
```

Enter the preceding directory, and use psql to connect to HybridDB for PostgreSQL instance' s database following to the procedure:

Use one of the following methods to connect to the database:

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
```

Parameter descriptions:

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

You can view more parameters by performing `psql-- help`. And in the `psql` prompt, you can view more supported `psql` commands by performing `'\?'`.

Enter the password to go to the `psql` shell interface. The `psql` shell is as follows:

```
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)
```

Reference

- For more usage descriptions of Greenplum `psql`, see `psql`.
- You also use the PostgreSQL `psql` command, but do note the difference in usage details. For more information, see [PostgreSQL 8.3.23 Documentation — psql](#).

JDBC

Users can use JDBC to connect to HybridDB for PostgreSQL instance's database. Here are two ways to get this tool:

Download JDBC provided by PostgreSQL official website, click [PostgreSQL JDBC Driver](#) and add the downloaded JDBC to the **CLASSPATH** Class variable before using it.

Use the tool package provided by the Greenplum official website. For more information, see [Greenplum Database 4.3 Connectivity Tools for UNIX](#).

Providing the follow code example as a reference, and the users can modify it according to the practices.

Code example:

```
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","mygpdb");

            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 information, see [PostgreSQL JDBC Interface](#).

Python

Users can also use Python to connect to HybridDB for PostgreSQL instance's database. Python uses the psycopg2 library to connect to Greenplum and PostgreSQL. The procedure for using the tool is described as follows:

Install psycopg2. In CentOS, three methods are available:

Perform `yum -y install python-psycopg2`

Perform `pip install psycopg2`

Install from the 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 environment variable before using the tool. For example,

```
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()
```

A result similar to the following is returned.

```
(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 to manipulate database. After Greenplum or PostgreSQL is installed, you can find its static libraries and dynamic libraries under the *lib* directory.

For related cases, see [libpq Example Programs](#).

- For the details of libpq, see [PostgreSQL 9.4 Documentation - Chapter 31. libpq - C Library](#).

ODBC

PostgreSQL ODBC is an open-source version based on the LGPL (GNU Lesser General Public License) protocol. You can download it from the [official website of PostgreSQL](#).

Procedure

Install the driver.

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

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 the 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 is connected to the instance, connect applications to ODBC. For more information, see [PostgreSQL ODBC driver](#) and [psqlODBC HOWTO - C#](#).

Windows and other platforms

Go to [Pivotal Greenplum Client](#) for download links of other client tools for Windows and other platforms.

Reference

- [Pivotal Greenplum Official Documentation](#)
- [PostgreSQL psql ODBC](#)
- [PostgreSQL ODBC Compilation](#)
- [Greenplum ODBC Download](#)

- Greenplum JDBC Download

Import data

Parallel import from OSS or export to OSS

HybridDB for PostgreSQL supports parallel import from OSS or export to OSS through external tables (which is called the gpossex function). It can also compress OSS external table files in gzip format to reduce the storage space and the costs.

The gpossex function can read or write text/csv files or text/csv files in gzip format.

Create an extension for OSS external tables (oss_ext)

Before using OSS external tables, create an oss_ext extension for each database.

- 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

Follow these steps to import data:

Distribute and store the data evenly in 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

Follow these steps 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>
```

Syntax of creating OSS external tables

The syntax of creating OSS external tables is as follows:

```
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, use the intranet domain name (that is, with “internal” in the domain name) to avoid incurring public network traffic.

id: OSS account ID.

key: OSS account key.

bucket: Specifies the bucket where the data file is located. You need to create the bucket in OSS in advance.

prefix: Specifies the prefix of the corresponding path 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. You can set only one of them.

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

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

- test/filename
- test/filenameexxx
- test/filename/aa

- test/filenameyyy/aa
- test/filenameyyy/bb/aa

If you have specified `prefix=test/filename/`, only the following files are imported (other files precedingly listed are not imported):

- test/filename/aa

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

Note: When more than one file are exported, every data node exports 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 allows you to use the same external table to export data multiple times. The exported files from each export are identified by the `UUID`, and the exported files in the same export share the same `UUID`.

`dir`: The path of virtual folders in OSS. It is mutually exclusive with `prefix` and `filepath`, you can only set one of them.

The folder path ends with `"/` `"`. For example, `test/mydir/`.

If you use this parameter to create an external table during data importing, all the files under the specified virtual directory are 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 is exported to the multiple files under this directory. The output file names follow the 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 set one of them. And you can **ONLY** specify *filepath* 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:

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

As a result, the imported files include filename, filename.1, and filename.2. Because filename.3 does not exist, filename.4 won't be imported.

Import mode parameters

`async`: whether to enable asynchronous mode to import data or not.

- Enabling worker threads to import data from OSS can improve 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.

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. When this limit is exceeded, the export switches 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.

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 according to the specified distribution key.

Other general parameters

The import and export modes also involve the following fault tolerance parameters:

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

`oss_speed_time`: Sets the maximum tolerable duration. The default value is 15 seconds.

With all the preceding parameters set as default, timeout is 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 by 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 participates 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)
```

Attention

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 must 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](#).

TEXT/CSV format description

You can specify the following parameters in the external table DDL parameters to specify the file format for read/write operations of OSS:

- The TEXT/CSV row delimiter is `'\n'`, which is a newline character.
- DELIMITER is the delimiter used to define columns:
 - If DELIMITER is included in the user data, the QUOTE parameter is required.
 - Recommended column delimiters are `'`, `'\t'`, `'|'` or some infrequent characters.
- QUOTE is used to wrap user data that contains special characters in columns.
 - Strings containing special characters are wrapped by QUOTE to distinguish user data from control characters.
 - Sometimes you do not need to wrap data with QUOTE due to considerations of performance optimization, for example, in the case of integers.
 - QUOTE cannot be the same as DELIMITER. The default QUOTE is double quotes.
 - If the user data contains a QUOTE character, it needs to be distinguished by using

the escape character ESCAPE.

- ESCAPE is the escape character for special characters
 - The escape character appears before certain special characters to indicate that they are not special characters.
 - ESCAPE is the same as QUOTE by default, which is double quotes.
 - It can also be specified as `'\'` (MySQL's default escape character) or other characters.

Typical Default TEXT/CSV Control Characters

Control Character \ Format	TEXT	CSV
DELIMITER (Column delimiter)	\t (tab)	, (comma)
QUOTE (quoted)	" (double-quote)	"(double-quote)
ESCAPE (escape)	(Not applicable)	Same as QUOTE
NULL (null)	\N (backslash-N)	(empty string without quotes)

All control characters must be single-byte characters.

SDK error handling

If the data import and export fails, 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 the 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 by using `oss_ext` related parameters.

FAQs

If the import is too slow, see the import performance descriptions in the preceding [Attention](#) 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](#)

Use Data Integration to synchronize data

Data Integration is a data synchronization platform provided by Alibaba Cloud big data service. The platform offers offline (full/incremental) data access channels for more than 20 data sources of different network environments and supports data storage across heterogeneous systems and elastic expansion, featuring high reliability, high security, and low costs. Check out the [Supported data source types](#) to learn about data sources available.

This document describes how to use Data Integration for **Data Import** and **Data Export** with HybridDB for PostgreSQL. It provides both procedures in the **Wizard Mode** (guided by a visualized interface) and sample code in the **Script Mode** (template-based parameter configuration).

Use cases

Using the synchronization jobs in Data Integration, you can:

Synchronize data in HybridDB for PostgreSQL to other data sources and perform expected processing on the data.

Synchronize processed data from other data sources to HybridDB for PostgreSQL.

Prerequisites

Complete the following operations on the Data Integration and HybridDB for PostgreSQL ends respectively.

Data Integration

Follow these steps to create a project in Data Integration.

Open a real-name-authenticated account on the official Alibaba Cloud website and create an AccessKey for accessing the account.

Activate MaxCompute and the system automatically generates a default ODPS data source. Log on to Data IDE by using the primary account.

Create a project. Users can collaborate in projects to complete a workflow and jointly maintain data and jobs. For this reason, you must create a project first before using Data IDE.

If you want to create data integration jobs by using a subaccount, you must grant related permissions to the subaccount.

HybridDB for PostgreSQL

Before importing data, you must create the target database and table in HybridDB for PostgreSQL you want to migrate data to on the PostgreSQL client.

If the source database to export data from is HybridDB for PostgreSQL, we recommend that you set the IP whitelist in the HybridDB for PostgreSQL console. You can follow these steps to set the IP whitelist.

Log on to the HybridDB for PostgreSQL console.

Select the expected instance, and click **Add Whitelist Group** on the **Whitelist Settings** page under the **Data Security** page.

Add the following IP addresses:

10.152.69.0/24,10.153.136.0/24,10.143.32.0/24,120.27.160.26,10.46.67.156,120.27.160.81,10.46.64.81,121.43.110.160,10.117.39.238,121.43.112.137,10.117.28.203,118.178.84.74,10.27.63.41,118.178.56.228,10.27.63.60,118.178.59.233,10.27.63.38,118.178.142.154,10.27.63.15,100.64.0.0/8.

Note: If you use a custom resource group to schedule a HybridDB for PostgreSQL data synchronization job, you must add the IP address of the computer hosting the custom resource group to the HybridDB for PostgreSQL whitelist.

Add data source

A new HybridDB for PostgreSQL data source must added to Data Integration before you can use Data

Integration for data synchronization to HybridDB for PostgreSQL. Follow these steps to add a data source.

Log on to Alibaba Cloud DTplus platform as a developer and click **Data IDE > Console**.

Click **Enter Work Zone** in the action bar of the corresponding project.

Click the data source in the Data Integration module in the top menu bar.

Click **New Data Source**.

In the **New Data Source** window, select **PostgreSQL** as the **Data Source Type**.

Select to configure the PostgreSQL data source in the form of a **JDBC** instance. The parameters include:

- Data Source Name: consists of letters, numbers, and underscores. It must begin with a letter or an underscore, and cannot exceed 60 characters.
- Data Source Description: a brief description of the data source. The description must not exceed 80 characters in length.
- Data Source Type: select PostgreSQL.
- Network Type: the select the network type.
- JDBC URL: the JDBC URL. Format: jdbc:PostgreSQL://IP:Port/database.
- Username/Password: the username and password used to connect to the database.

When you complete the settings, click **Test Connectivity**.

When the connectivity test is passed, click **OK**.

Import data by using Data Integration

You can use one of the following methods to configure the synchronization job.

If you use the visualized wizard, see [Configure synchronization jobs in the wizard mode](#). The wizard mode can be switched to the script mode.

If you use template-based parameter configuration, see [Configure synchronization jobs in the script mode](#). The script mode cannot be switched to the wizard mode.

Before going ahead, make sure you have added the HybridDB for PostgreSQL data source to Data

Integration by following the Add data source procedure.

Configure synchronization jobs in the wizard mode

Follow these steps to configure the synchronization job.

Select the **Wizard Mode** to create a synchronization job.

Select a data source. The parameters include:

- **Data Source:** select **odps_first(odps)**, that is, MaxCompute.
- **Table:** select **hpg**.
- **Data Preview:** the window is collapsed by default. You can click it to expand it.

After entering the preceding information, click **Next**.

Select a target. The parameters include:

- Data Source: select **I_PostGreSql(postgresql)**.
- Table: select **public.person**.
- Prepared Statement Before Import: enter the SQL statement to run before the data synchronization job starts.

Currently, you can run only one SQL statement in the wizard mode. But you can run multiple SQL statements in the script mode. For example, to clear old data.

- Prepared Statement after Import: enter the SQL statement to run after the data synchronization job starts.

Currently, you can run only one SQL statement in the wizard mode. But you can run multiple SQL statements in the script mode. For example, to add a time stamp.

- Primary Key Conflict: select **Insert Into**. If the primary key conflicts with the unique index, Data Integration processes the data as dirty data.

After entering the preceding information, click **Next**.

Map fields. You must configure the field mapping relationships. The **Source Table Fields** on the left correspond one to one with the **Target Table Fields** on the right.

Description:

- You can enter constants. The value must be enclosed in single-byte single

- quotation marks. For example, 'abc' or '123' .
- Scheduling parameters can be used together. For example, `#{bdp.system.bizdate}` and others.
- You can enter the partition columns to synchronize. For example, partition columns with PT.
- If the value you entered cannot be parsed, the type is displayed as 'Unrecognized' .
- You cannot configure ODPS functions.

After that, click **Next**.

Control channels. You can configure the maximum job rate and dirty data checking rules. The parameters include:

Maximum Job Rate: determines the highest rate possible for data synchronization jobs. The actual rate of the job may vary with the network environment, database configuration, and other factors.

Number of Concurrent Jobs: the maximum job rate = Number of concurrent jobs * Transmission rate of a single concurrent job. When the maximum job rate is specified, use the following method to select the number of concurrent jobs:

- If your data source is an online business database, we recommend that you not set a large value for the concurrent job count to avoid interfering with the online database.
- If you require a high data synchronization rate, we recommend that you select the highest job rate and a large concurrent job count.

Preview and save settings. After the preceding configuration, you can scroll up or down to view the job configuration. After that, click **Save**.

Get results. After saving a synchronization job,

- Click **Run Job** to run the job immediately.
- Click **Submit** on the right to submit the synchronization job to the scheduling system.

The scheduling system automatically and periodically runs the job from the next day according to the configuration attributes. For related scheduling configuration, see [Scheduling configuration](#).

Configure synchronization jobs in the script mode

The sample code is as follows:

```

{
  "configuration": {
    "reader": {
      "plugin": "odps",
      "parameter": {
        "partition": "pt=${bdp.system.bizdate}"//Partition information
        "datasource": "odps_first"//Data source name. We recommend that you add the data source before configuring
        synchronization jobs. The value of this configuration item must be the same as the name of the data source you
        added.
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "table": "hpg"//Source table name
      }
    },
    "writer": {
      "plugin": "postgresql",
      "parameter": {
        "postSql": []//Prepare the statement after the import
        "datasource": "I_PostGreSql"//Data source name. We recommend that you add the data source before configuring
        synchronization jobs. The value of this configuration item must be the same as the name of the data source you
        added.
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "table": "public.person"//Target table name
        "preSql": []//Prepare the statement before the import
      }
    },
    "setting": {
      "speed": {
        "concurrent": 7//Number of concurrent jobs
        "mbps": 7//The maximum job rate
      }
    }
  },
  "type": "job",
  "version": "1.0"
}

```

Export data by using Data Integration

You can use one of the following methods to configure the synchronization job.

- If you use the visualized wizard, see [Configure synchronization jobs in the wizard mode](#).
- If you use template-based parameter configuration, see [Configure synchronization jobs in the script mode](#).

Before going ahead, make sure you have added the HybridDB for PostgreSQL data source to Data Integration by following the [Add data source procedure](#).

Configure synchronization jobs in the wizard mode

Follow these steps to configure the synchronization job.

Select the **Wizard Mode** to create a synchronization job.

Select a source. The parameters include:

- Data Source: select **I_PostGreSql(postgresql)**.
- Table: select **public.person**.
- Data Preview: the window is collapsed by default. You can click it to expand it.

Data Filtering: set the filtering condition for data synchronization.

PostgreSQLReader concatenates an SQL statement based on the specified column, table, and WHERE conditions, and extracts data according to the SQL statement.

For example, you can specify the actual use case in the where condition during a test. Usually the data on the day is selected for synchronization. In this case, you can set the where condition to `id > 2` and `sex = 1`. The where condition can effectively help with incremental business data synchronization. If the where condition is not configured or is left null, full table data synchronization applies.

Split key: if you specify the splitPk when using PostgreSQLReader to extract data, it means that you want to use the fields represented by the splitPk for data sharding. In this case, the Data Integration initiates concurrent jobs to synchronize data, which greatly improves the efficiency of data synchronization.

We recommend that you use primary keys of tables, because primary keys are generally evenly distributed with less risks of data hot spots. The splitPk only supports splitting integers, and does not support strings, floating points, dates, and other types. If the non-supported data type is specified as the splitPk, the splitPk feature is ignored and data is synchronized in a single channel. If the splitPk value is not provided, including a null value is provided, data in the table is synchronized in a single channel.

Select a target. The parameters include:

- Data Source: select **odps_first(odps)**, that is, MaxCompute.
- Table: select **hpg**.

After entering the preceding information, click **Next**.

Map fields. You must configure the field mapping relationships. The **Source Table Fields** on the left correspond one to one with the **Target Table Fields** on the right. After that, click **Next**.

Control channels. You can configure the maximum job rate and dirty data checking rules. After that, click **Next**.

Preview and save settings. After the preceding configuration, you can scroll up or down to view the job configurations. After that, click **Save**.

So far, you have created a data synchronization job in the wizard mode to export data from HybridDB for PostgreSQL.

Configure synchronization jobs in the script mode

The sample code is as follows:

```
{
  "configuration": {
    "reader": {
      "plugin": "postgresql",
      "parameter": {
        "datasource": "I_PostGreSql", //Data source name. We recommend that you add the data source before configuring
        synchronization jobs. The value of this configuration item must be the same as the name of the data source you
        added.
        "table": "public.person", //Source table name
        "where": "", //Filtering condition
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "splitPk": "" //Split key
      }
    },
    "writer": {
      "plugin": "odps",
      "parameter": {
        "datasource": "odps_first", //Data source name. We recommend that you add the data source before configuring
```

synchronization jobs. The value of this configuration item must be the same as the name of the data source you added.

```
"column": [
  "id",
  "name",
  "year",
  "birthdate",
  "ismarried",
  "interest",
  "salary"
],
"table": "hpg", //Target table name
"truncate": true,
"partition": "pt=${bdp.system.bizdate}" //Partition information
}
},
"setting": {
  "speed": {
    "mbps": 5, //The maximum job rate
    "concurrent": 5 //Number of concurrent jobs
  }
}
},
"type": "job",
"version": "1.0"
}
```

Import data from MySQL

The mysql2pgsql tool supports migrating tables in MySQL to HybridDB for PostgreSQL, Greenplum Database, PostgreSQL, or PPAS without storing the data separately. This tool connects to the source MySQL database and the target database at the same time, queries and retrieves the data to be exported in the MySQL database, and then imports the data to the target database by using the COPY command. It 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 are 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 are used concurrently.

Typical usage

Full-database migration

The procedure is as follows:

Run the following command to get the DDL statements of the corresponding table on the target end:

```
./mysql2pgsql -d
```

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

Run the following command to synchronize all tables:

```
./mysql2pgsql
```

This command migrates 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 procedure is as follows:

Create a new file (tab_list.txt) 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

Download the binary installer of `mysql2pgsql`

View the `mysql2pgsql` source code compilation instructions

Import data from PostgreSQL

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.

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 versions) 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 must 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 only supports logic flow replication 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
```

By default, the migration program migrates the table data of all the users in the corresponding PostgreSQL database 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

Download the binary installer of rds_dbsync

View the rds_dbsync source code compilation instructions

Import data by using the COPY command

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 unavailable 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 the 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 using JDBC that encapsulates the CopyIn method to run the COPY statements. For detailed method, see [Interface CopyIn](#).
- For the usage of COPY command, see [COPY](#).