

# Alibaba Cloud AnalyticDB for PostgreSQL

## Quick Start

Issue: 20190531

## Legal disclaimer

---

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.



# Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 <b>Danger:</b> Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 <b>Warning:</b> Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 <b>Notice:</b> Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 <b>Note:</b> You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
<b>Bold</b>	It is used for buttons, menus, page names, and other UI elements.	Click <b>OK</b> .
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid <i>Instance_ID</i></code>
[ ] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>swich {stand   slave}</code>



# Contents

---

Legal disclaimer.....	I
Generic conventions.....	I
1 Overview.....	1
2 Create an instance.....	2
3 Set up an instance.....	4
3.1 Set up a whitelist.....	4
3.2 Set up an account.....	6
3.3 Set the network type.....	7
4 Connect to a AnalyticDB for PostgreSQL database.....	9
5 Import data.....	17
5.1 Migrate data using different solutions.....	17
5.2 Migrate data by using an OSS external table.....	18
5.3 Use Data Integration to synchronize data.....	28
5.4 Migrate data from MySQL to AnalyticDB for PostgreSQL.....	36
5.5 Migrate data from PostgreSQL to AnalyticDB for PostgreSQL.....	39
5.6 Migrate data to AnalyticDB for PostgreSQL by using the Copy command....	41
5.7 Migrate data from Amazon Redshift to ApsaraDB AnalyticDB for PostgreSQL.....	42



# 1 Overview

---

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

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

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

1. *Create an instance*.
2. Set up an instance, including *Set up a whitelist*, *Set up an account*, and *Set 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*.

## 2 Create an instance

---

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

- Create an instance in the AnalyticDB for PostgreSQL console.
- Purchase an instance on the AnalyticDB for PostgreSQL Purchase Page.

This document describes the detailed steps for creating a AnalyticDB for PostgreSQL instance in the console.

### Billing method

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

### Prerequisites

You have registered an account and signed up.

### Procedure

1. Log on to the [AnalyticDB for PostgreSQL console](#).
2. Click Create Instance.
3. 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.
4. Confirm your order information, and then click Buy Now.
5. Click Activate to activate the instance.
6. Go to the Instance List page of [AnalyticDB 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.

**Related API**

API	Description
<a href="#">CreateDBInstance</a>	Creates a database instance.

## 3 Set up an instance

---

### 3.1 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 AnalyticDB for PostgreSQL databases:

- Access from the Internet.
- Access from the intranet. The network types of AnalyticDB for PostgreSQL and ECS instances must be identical.
- Access from the intranet and Internet at the same time. The network types of AnalyticDB for PostgreSQL and ECS instances must be identical.



#### Note:

To set the network type, see [Set the Network Type](#).

#### Procedure

1. Log on to the [AnalyticDB for PostgreSQL console](#).
2. Select the region where the target instance is located.
3. Click the ID of the instance to go to the Basic Information page of the instance.
4. In the left-side navigation pane, click Security Controls.
5. 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.

6. 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.

7. Click OK to add the whitelist.

Next

The whitelist provides an advanced access protection for AnalyticDB 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.

Related API

API	Description
<a href="#">DescribeDBInstanceIPArrayList</a>	Returns a list of IP addresses that are allowed to access the database instance.
<a href="#">ModifySecurityIps</a>	Modifies the whitelist of IP addresses.

## 3.2 Set up an account

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

### Create an account

#### Prerequisites

Before using a AnalyticDB 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

1. Log on to the [AnalyticDB for PostgreSQL console](#).
2. Select the region where the target instance is located.
3. Click the ID of the instance to go to the Basic Information page of the instance.
4. Click Account Management in the left-side navigation pane.
5. Click Create Account.
6. 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 AnalyticDB for PostgreSQL, if you forget the password of the database account, you can reset the password in the [AnalyticDB for PostgreSQL console](#).



#### Note:

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

#### Procedure

1. Log on to the [AnalyticDB for PostgreSQL console](#).
2. Select the region where the target instance is located.
3. Click Manage under the Action column of the target instance to go to the Basic Information page of the instance.
4. Click Account Management in the left-side navigation pane.
5. Click Reset password under the account to be managed.
6. 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.

#### Related API

API	Description
<a href="#">CreateAccount</a>	Creates an account.
<a href="#">DescribeAccounts</a>	Returns the account information for a database.
<a href="#">ModifyAccountDescription</a>	Modifies the description of the database.
<a href="#">ResetAccountPassword</a>	Resets the password for an account.

## 3.3 Set the network type

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

1. Create a VPC in the same region with the target AnalyticDB for PostgreSQL instance. For detailed steps, see [Create a VPC](#).
2. Log on to the [AnalyticDB for PostgreSQL console](#).
3. Select the region where the target instance is located.
4. Click the ID of the instance to go to the Basic Information page of the instance.
5. Click Database Connection.
6. Click Switch to VPC.
7. 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 AnalyticDB for PostgreSQL instance. The original Internet address remains unchanged.

## Related API

API	Description
<a href="#">ModifyDBInstanceNetworkType</a>	Switches the network connection type for an instance.



## 4 Connect to a AnalyticDB for PostgreSQL database

---

Cloud Database AnalyticDB 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

AnalyticDB for PostgreSQL users can use Greenplum-supported graphical client tools directly, such as [SQL Workbench](#), [Navicat PremiumNavicat For PostgreSQL](#), [pgAdmin III \(1.6.3\)](#) 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 AnalyticDB for PostgreSQL. For more information, see [pgAdmin 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.



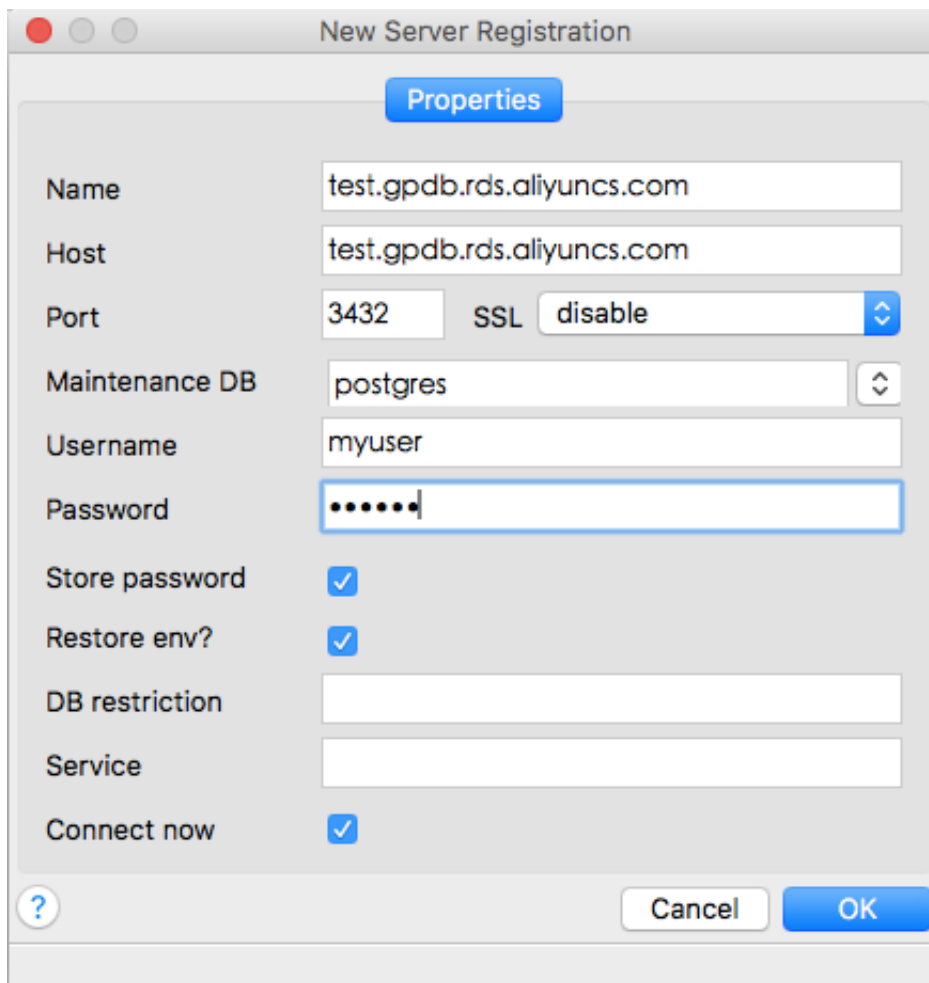
#### Note:

AnalyticDB 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

1. Download and install pgAdmin III 1.6.3 or an earlier version.
2. Select File > Add Server to go to the New Server Registration page.

### 3. Enter the Properties as shown in the following figure:



The screenshot shows a 'New Server Registration' dialog box with a 'Properties' tab. The fields are filled with the following information:

- Name: test.gpdb.rds.aliyuncs.com
- Host: test.gpdb.rds.aliyuncs.com
- Port: 3432
- SSL: disable
- Maintenance DB: postgres
- Username: myuser
- Password: masked with dots
- Store password:
- Restore env?:
- DB restriction: (empty)
- Service: (empty)
- Connect now:

Buttons: Cancel, OK

### 4. Click OK to connect to the AnalyticDB for PostgreSQL database.

#### Command Line tools

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

#### psql

psql is a common command line client tool for AnalyticDB 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 AnalyticDB for PostgreSQL instance' s database following to the procedure:

### 1. Use one of the following methods to connect to the database:

- Connection strings

```
psql "host = yourgpdbad.dress.gpdb.rds.aliyuncs.com
port = 3432 dbname = postgres user = gpdbaccount password
= gpdbpassword"
```

- Specify parameters

```
psql -h yourgpdbad.dress.gpdb.rds.aliyuncs.com -p
3432 -d postgres -U gpdbaccount
```

Parameter descriptions are as follows:

- **-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 `\?`.

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

```
postgres =>
```

## 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 AnalyticDB 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 : 3432
/ 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 . printStack Trace ();
} catch ( SQLException e ) {
    e . printStack Trace ();
}
}
}

```

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

## Python

Users can also use Python to connect to AnalyticDB 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:

### 1. 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

```

### 2. 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 .
aliyun . com ', port = 3432 )
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

### 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 definition s
# Driver from the postgresql - odbc package
# Setup from the unixODBC package
[ PostgreSQL ]
Description = ODBC for PostgreSQL
Driver = / usr / lib / psqlodbcw . so
Setup = / usr / lib / libodbcpsq ls . so
Driver64 = / usr / lib64 / psqlodbcw . so
Setup64 = / usr / lib64 / libodbcpsq ls . 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
```

3. 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
```

4. 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
```

5. 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](#)



## 5 Import data

### 5.1 Migrate data using different solutions

AnalyticDB for PostgreSQL provides various migration solutions, which meet different business needs such as migrating data from on-premises PostgreSQL databases to AnalyticDB for PostgreSQL databases and migrating data between cloud services. It enables you to smoothly migrate data between AnalyticDB for PostgreSQL databases and other databases without affecting your business. Implement data migration from databases such as AnalyticDB for PostgreSQL, Greenplum Database, PostgreSQL, PPAS, and Amazon Redshift to AnalyticDB for PostgreSQL.

Data migration scenarios of AnalyticDB for PostgreSQL and related operations are as follows:

Operation	Migration Type	Scenario
<a href="#">Migrate data by using an OSS external table</a>	Migrate data to AnalyticDB for PostgreSQL/Migrate data between Alibaba Cloud products/Migrate data to on-premises database	Use OSS external tables to import and export data between AnalyticDB for PostgreSQL and OSS.
<a href="#">Use Data Integration to synchronize data</a>	Migrate data between Alibaba Cloud products	Use Data Integration to import data into or export data from AnalyticDB for PostgreSQL.
<a href="#">Migrate data from MySQL to AnalyticDB for PostgreSQL</a>	Migrate data to AnalyticDB for PostgreSQL	Use the mysql2pgsql tool to import tables from local MySQL into AnalyticDB for PostgreSQL.
<a href="#">Migrate data from PostgreSQL to AnalyticDB for PostgreSQL</a>	Migrate data to AnalyticDB for PostgreSQL/Migrate data between Alibaba Cloud products	Use the mysql2pgsql tool to import tables from AnalyticDB for PostgreSQL, Greenplum, PostgreSQL, or PPAS into AnalyticDB for PostgreSQL.

Operation	Migration Type	Scenario
<a href="#">Migrate data to AnalyticDB for PostgreSQL by using the Copy command</a>	Migrate data to AnalyticDB for PostgreSQL	Use the <code>\ COPY</code> command to import data from local text files into AnalyticDB for PostgreSQL.

## 5.2 Migrate data by using an OSS external table

AnalyticDB for PostgreSQL can import or export data from or to OSS tables in parallel by using the OSS external table function, `gpossect`. AnalyticDB for PostgreSQL also supports gzip file compression for the OSS external tables to reduce file size and storage costs.

Currently, `gpossect` can read from and write to TEXT and CSV files as well as gzip compressed TEXT and CSV files.

This topic includes the following contents:

- [Operation instructions](#)
- [Parameter description](#)
- [Examples](#)
- [Precautions](#)
- [TEXT and CSV format description](#)
- [SDK error handling](#)
- [FAQ](#)
- [References](#)

### Operation instructions

Accessing and editing OSS external tables through AnalyticDB for PostgreSQL consists of the following operations:

- [Create an OSS external table plug-in \(`oss\_ext`\)](#)
- [Import data in parallel](#)
- [Export data in parallel](#)
- [Create OSS external table syntax](#)

Create an OSS external table plug-in (`oss_ext`)

To use an OSS external table, you must first create an OSS external table plug-in in AnalyticDB for PostgreSQL. You must create a plug-in for each database that you want to access.

- **Creation statement:** `CREATE EXTENSION IF NOT EXISTS oss_ext ;`
- **Deletion statement:** `DROP EXTENSION IF EXISTS oss_ext ;`

### Import data in parallel

Perform the following procedure to import data:

1. Distribute data evenly among multiple OSS files for storage. We recommend that the number of OSS files be an integer multiple of the number of segments in AnalyticDB for PostgreSQL.
2. Create a READABLE external table in AnalyticDB for PostgreSQL.
3. Execute the following statement to import data in parallel:

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

### Export data in parallel

Perform the following procedure to export data:

1. Create a WRITABLE external table in AnalyticDB for PostgreSQL.
2. Execute the following statement to export data to OSS in parallel:

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

### Create OSS external table syntax

Execute the following statements to create OSS external table syntax:

```
CREATE [ READABLE ] EXTERNAL TABLE tablename
( columnname datatype [, ...] | LIKE othertable )
LOCATION (' ossprotoco l ')
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 ' ]
[ DISTRIBUTE D BY ( column , [ ... ] ) | DISTRIBUTE D
RANDOMLY ]
ossprotocol :
    oss:// oss_endpoint prefix = prefix_name
    id = userossid key = useroskey bucket = ossbucket
    compressio ntype =[ none | gzip ] async =[ true | false ]
ossprotocol :
    oss:// oss_endpoint dir =[ folder /[ folder /]...]/
file_name
    id = userossid key = useroskey bucket = ossbucket
    compressio ntype =[ none | gzip ] async =[ true | false ]
ossprotocol :
    oss:// oss_endpoint filepath =[ folder /[ folder /]...]/
file_name
    id = userossid key = useroskey bucket = ossbucket
    compressio ntype =[ none | gzip ] async =[ true | false ]

```

## Parameter description

This section provides definitions of parameters used in various operations, including:

- [Common parameters](#)
- [Import mode parameters](#)
- [Export mode parameters](#)
- [Other common parameters](#)

## Common parameters

- **Protocol and endpoint:** indicates the communication protocol and endpoint address in the "protocol name://oss\_endpoint" format. The protocol name indicates oss and oss\_endpoint indicates the domain name of the OSS region.



**Notice:**

You can access the database from an Alibaba Cloud host by using an internal endpoint containing "internal" in the name in order not to generate public traffic.

- **id**: the AccessKey ID of the OSS account.
  - **key**: the AccessKey Secret of the OSS account.
  - **bucket**: the bucket that contains the data file to be operated. It must be an existing bucket in OSS.
  - **prefix**: the prefix of the path name corresponding to the data file. Prefixes are directly matched and cannot be controlled by regular expressions. Only one parameter among prefix, filepath, and dir can be specified at a time as they are mutually exclusive.
- All OSS files containing the specified prefix will be imported if you create a READABLE external table for data import.
    - The following files will be imported if you set prefix to test/filename:
      - test/filename
      - test/filenamexxx
      - test/filename/aa
      - test/filenameyyy/aa
      - test/filenameyyy/bb/aa
    - Only the following file out of the preceding files will be imported if you set prefix to test/filename/:
      - test/filename/aa
  - The exported files are uniquely named based on this parameter if you create a READABLE external table for data import.



**Notice:**

One or more files can be exported for each data node. The names of exported files are in the `prefix_tab lename_uuid . x` format. **uuid** indicates a timestamp in microseconds as an int64 value. **x** indicates the node ID. You can use an external table for multiple export operations. The files exported at a time have a unique uuid value.

- **dir**: the virtual folder path in OSS. Only one parameter among prefix, filepath, and dir can be specified at a time as they are mutually exclusive.
  - A folder path must end with "/" such as `test / mydir /`.
  - If you use this parameter to create an external table for data import, all files under the specified virtual directory will be imported, excluding its subdirectories and files in those subdirectories. Unlike filepath, dir does not need to specify the names of files in the directory.
  - All data will be exported to multiple files in the specified directory if this parameter is used to create an external table for data export. The names of exported files are in the `filename . x` format, where x is a number. The values of x may be inconsecutive.
- **filepath**: the name of the OSS file, which contains the path. Only one parameter among prefix, filepath, and dir can be specified at a time as they are mutually exclusive. This parameter can be specified only when you are creating a READABLE external table for data import.
  - The file specified by filepath must include the file name and path, but cannot include the bucket.
  - The file name specified for data import must be in the `filename` or `filename . x` format. The values of x must be consecutive numbers starting from 1. For example, filepath is set to filename and OSS contains the following files:

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

The files named filename, filename.1, and filename.2 will be imported because they are numbered consecutively. filename.4 will not be imported because it is not consecutive in the file order.

### Import mode parameters

- **async**: indicates whether to enable asynchronous data import.
  - You can enable an auxiliary thread to import data from OSS for better performance.
  - Asynchronous data import is enabled by default. You can set `async` to `false` or `f` to disable asynchronous data import.
  - Asynchronous data import consumes more hardware resources than normal data import.
- **compressiontype**: indicates the format to be used to compress imported files.
  - If this parameter is set to `none` (default), imported files are not compressed.
  - If this parameter is set to `gzip`, files are imported in the GZIP format. Currently, only the GZIP format is supported.
- **compressionlevel**: indicates the degree to which files written to OSS are compressed. Valid values: 1 to 9. Default value: 6.

#### Export mode parameters

- **oss\_flush\_block\_size**: indicates the buffer size for a single data flush to OSS. Default value: 32 MB. Valid values: 1 MB to 128 MB.
- **oss\_file\_max\_size**: indicates the maximum size of files written to OSS. If a file reaches the maximum size, the remaining data is written to another file. Default value: 1024 MB. Valid values: 8 MB to 4000 MB.
- **num\_parallel\_worker**: indicates the number of active parallel compression threads when data is written to OSS. Valid values: 1 to 8. Default value: 3.
- **compressiontype**: indicates the compression format of exported files.
  - If this parameter is set to `none` (default), exported files are not compressed.
  - If this parameter is set to `gzip`, exported files are in the GZIP format. Currently, only the GZIP format is supported.

#### Take note of the following characteristics of export mode:

- `WRITABLE` is the keyword of the external table for data export. You must specify this keyword when creating an external table.
- Only the `prefix` and `dir` parameters are supported for data export. The `filepath` parameter is not supported.
- You can use the `DISTRIBUTED BY` clause to write data from segments to OSS based on specified distribution keys.

## Other common parameters

The following error-tolerance parameters can be used for data import and export:

- **oss\_connect\_timeout**: indicates the connection timeout period in seconds. Default value: 10 seconds.
- **oss\_dns\_cache\_timeout**: indicates the DNS timeout period in seconds. Default value : 60 seconds.
- **oss\_speed\_limit**: indicates the minimum data transmission rate. Default value: 1 Kbit/s.
- **oss\_speed\_time**: indicates the timeout period when the data transmission rate is lower than the minimum value. Default value: 15 seconds.

When the default values are used for the preceding parameters, a timeout is triggered if the transmission rate is lower than 1 Kbit/s for 15 consecutive seconds. For more information, see [OSS SDK error handling](#).

The other parameters are compatible with the original external table syntax of Greenplum. For more information about the syntax, see [Greenplum official documentation on external table syntax](#) . These parameters include:

- **FORMAT**: indicates the supported file format, such as TEXT and CSV.
- **ENCODING**: indicates the data encoding format of a file, such as UTF8.
- **LOG ERRORS**: indicates that the clause can ignore imported erroneous data and write the data to `error_table`. You can also use the count parameter to specify the error reporting threshold.

## Examples

```
# Create an OSS external table for data 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 compressio ntype = gzip ')
  FORMAT ' csv ' ( QUOTE ' ' ' DELIMITER E '\ t ' )
  ENCODING ' utf8 '
  LOG ERRORS INTO my_error_r ows 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 OSS external table for data 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 '
DISTRIBUTE D 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 '
DISTRIBUTE D BY ( date );
# Create a heap table for data loading .
create table example
( date text , time text , open float ,
  high float , low float , volume int )
DISTRIBUTE D BY ( date );
# Load data into example from ossexample in parallel .
insert into example select * from ossexample ;
# Export data from example to OSS in parallel
insert into ossexample _exp select * from example ;
# As shown in the following execution plan , all
segments are involved in the task .
# Each segment pulls data from OSS in parallel . The
redistribution motion node calculates the hash value
of the data and distribute s the hash value to the
corresponding segment . That segment then imports 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 )
-> Redistribution 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 )
# As shown in the following query plan , each
segment exports local data directly to OSS without
redistributing the data .
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 )
```

## Precautions

- The syntax for creating and using external tables, except for the syntax of location-related parameters, is the same as that for Greenplum.
- Data import performance depends on the OSS performance as well as available resources of the AnalyticDB for PostgreSQL cluster, such as CPU, I/O, memory, and network resources. We recommend that you use column-based storage and compression when creating a table for the best import performance. For example, you can specify the following clause: WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib, COMPRESSLEVEL=5, BLOCKSIZE=1048576). For more information, see [Greenplum official documentation on database table creation syntax](#).
- We recommend that OSS and AnalyticDB for PostgreSQL instances be in the same region for the best import performance.

## TEXT/CSV format description

The following parameters specify the formats of files read from and written to OSS. They can be specified in the external DDL parameters.

- The string used as a line delimiter or line break for TEXT and CSV files is '\n.'
- **DELIMITER**: indicates the string used to delimit columns.
  - If the DELIMITER parameter is specified, the QUOTE parameter must also be specified.
  - Recommended column delimiters include ',', '\t', '|', and certain uncommon characters.

- **QUOTE:** indicates a pair of characters that enclose a string that contains special characters.
  - QUOTE characters are used to differentiate user data strings that contain special characters from code for the machine.
  - For the sake of efficient coding, it is unnecessary to enclose data such as integers in QUOTE characters.
  - QUOTE cannot be the same string specified in DELIMITER. The default value of QUOTE is a pair of double quotation marks (").
  - User data that contains QUOTE characters must also contain ESCAPE characters to differentiate the user data from code for the machine.
- **ESCAPE:** indicates the escape character.
  - Escape characters are placed before characters that otherwise have a special meaning.
  - If ESCAPE is not specified, its default value is the same as QUOTE.
  - You can also use other characters as ESCAPE characters such as the '\ ' used by MySQL by default.

#### Default control characters for TEXT and CSV files

Control character	TEXT	CSV
DELIMITER	\t (tab)	, (comma)
QUOTE	" (double quotation mark)	" (double quotation mark)
ESCAPE	N/A	Same as QUOTE
NULL	\N (backslash-N)	(Empty string without quotation marks)



Note:

All control characters must be single-byte characters.

#### SDK error handling

If an error occurs during the import or export process, the error log contains the following information:

- **code:** the HTTP status code of the error request.
- **error\_code:** the OSS error code.

- `error_msg`: the OSS error message.
- `req_id`: the UUID that identifies the request. If you require assistance in solving a problem, you can submit a ticket with the `req_id` of the failed request to OSS development engineers.

For more information, see [OSS API error responses](#). Timeout errors can be handled by using `oss_ext` related parameters.

## FAQ

If the import process is taking an abnormally long time, see the descriptions on import performance in the “Precautions” section.

## References

- [OSS endpoint information](#)
- [OSS help](#)
- [OSS SDK error handling](#)
- [OSS API error responses](#)
- [Greenplum official documentation on database external table syntax](#)
- [Greenplum official documentation on database table creation syntax](#)

## 5.3 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 AnalyticDB 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 AnalyticDB for PostgreSQL to other data sources and perform expected processing on the data.
- Synchronize processed data from other data sources to AnalyticDB for PostgreSQL.

## Prerequisites

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

### Data Integration

Follow these steps to create a project in Data Integration.

1. Open a real-name-authenticated account on the official Alibaba Cloud website and create an AccessKey for accessing the account.
2. Activate MaxCompute and the system automatically generates a default ODPS data source. Log on to Data IDE by using the primary account.
3. 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.
4. If you want to create data integration jobs by using a subaccount, you must grant related permissions to the subaccount.

## AnalyticDB for PostgreSQL

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

If the source database to export data from is AnalyticDB for PostgreSQL, we recommend that you [set the IP whitelist](#) in the AnalyticDB for PostgreSQL console. You can follow these steps to set the IP whitelist.

1. Log on to the [AnalyticDB for PostgreSQL console](#).
2. Select the expected instance, and click Add Whitelist Group on the Whitelist Settings page under the Data Security page.
3. 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 AnalyticDB for PostgreSQL data synchronization job, you must add the IP address of the computer hosting the custom resource group to the AnalyticDB for PostgreSQL whitelist.

## Add data source

A new AnalyticDB for PostgreSQL data source must added to Data Integration before you can use Data Integration for data synchronization to AnalyticDB for PostgreSQL. Follow these steps to add a data source.

1. Log on to the [DataWorks console](#) as an administrator and click Enter Workspace in the actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source source to pop up the supported data source.
4. In the New Data Source window, select PostgreSQL as the Data Source Type.
5. Select to configure the PostgreSQL data source in the form of a JDBC instance. The parameters include:
  - Type: data source without a public IP address.
  - Name: It is a combination of letters, numbers, and underlines It must begin with a letter or underline and cannot exceed 60 characters.
  - Description: It is a brief description of the data source with no more than 80 characters.
  - Resource Group: It is used to run synchronization tasks, and generally multiple machines can be bound when you add a resource group. For details, see [Add scheduling resources](#).
  - JDBC URL: Format: jdbc:mysql://ServerIP:Port/database.
  - Username/Password: The user name and password used to connect to the database.
6. When you complete the settings, click Test Connectivity.
7. When the connectivity test is passed, click Complete.

## 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 AnalyticDB 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.

1. Select the Wizard Mode to create a synchronization job.
2. 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.

3. 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.

4. 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.

5. 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.

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



## 7. 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
        informatio n
        " datasource ": " odps_first ", // Data source name .
        We recommend that you add the data source before
        configurin g synchroniz ation jobs . The value of this
        configurat ion 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 ": " l_PostGreS ql ", // Data source name .
        We recommend that you add the data source before
        configurin g synchroniz ation jobs . The value of this
        configurat ion 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
      }
    }
  }
}
```

```

import
  " preSql ": [] // Prepare the statement before the
}
},
" 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 AnalyticDB 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.

1. Select the Wizard Mode to create a synchronization job.
2. 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. PostgreSQL Reader 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.

3. 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.

4. 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.

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

6. 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 AnalyticDB for PostgreSQL.

Configure synchronization jobs in the script mode

The sample code is as follows:

```
{
  " configuration ": {
    " reader ": {
      " plugin ": " postgresql ",
      " parameter ": {
        " datasource ": " l_PostgreSQL ",// Data source name .
        " 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 .
      " column ": [
        " id ",
        " name ",
        " year ",
        " birthdate ",
        " ismarried ",
        " interest ",
        " salary "
      ],
      " table ": " hpg ",// Target table name
      " truncate ": true ,
      " partition ": " pt =${ bdp . system . bizdate }"// Partition
    }
  },
  " setting ": {
    " speed ": {
      " mbps ": 5 ,// The maximum job rate
      " concurrent ": 5 // Number of concurrent jobs
    }
  },
  " type ": " job ",
  " version ": " 1 . 0 "
}

```

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.

## 5.4 Migrate data from MySQL to AnalyticDB for PostgreSQL

The `mysql2pgsql` tool supports migrating tables in MySQL to AnalyticDB 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 "
encodingdi r = " share "
encoding = " utf8 "
```

- The connection information of the target PostgreSQL database (including PostgreSQL, PPAS and AnalyticDB 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_st ring = " host = 192 . 168 . 1 . 2   dbname = test
port = 3432   user = test   password = pgsql "
```

## Usage description

The usage of `mysql2pgsql` is described as follows:

```
./ mysql2pgsql l - l < tables_lis t_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_lis t_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:

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

```
./ mysql2pgsq l - d
```

2. Create a table on the target based on these DDL statements with the distribution key information added.
3. Run the following command to synchronize all tables:

```
./ mysql2pgsq l
```

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:

1. Create a new file ( `tab_list . txt` ) and insert the following content:

```
t1
```

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

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

```
./mysql2pgsql l -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](#)

## 5.5 Migrate data from PostgreSQL to AnalyticDB for PostgreSQL

The `pgsql2pgsql` tool supports migrating tables in AnalyticDB for PostgreSQL, Greenplum Database, PostgreSQL, or PPAS to AnalyticDB 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 AnalyticDB for PostgreSQL to PostgreSQL, PPAS, Greenplum Database, or AnalyticDB 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_st ring = " host = 192 . 168 . 1 . 1    dbname = test
port = 3432    user = test    password = pgsql "
```

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

```
[ local . pgsql ]
connect_st ring = " host = 192 . 168 . 1 . 2    dbname = test
port = 3432    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_st ring = " host = 192 . 168 . 1 . 3    dbname = test
port = 3432    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 stream replication, so it supports the incremental migration if PostgreSQL serves as the data source. The kernel only supports logic stream 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:

```
./ pgsq12pgsq 1
```

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](#)

## 5.6 Migrate data to AnalyticDB for PostgreSQL by using the Copy command

You can directly run the `\ COPY` command to import local text file data to AnalyticDB 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 in this case 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.

Syntax 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 ]

```

- AnalyticDB 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](#).

## 5.7 Migrate data from Amazon Redshift to ApsaraDB AnalyticDB for PostgreSQL

This topic describes how to migrate data from Amazon Redshift to ApsaraDB AnalyticDB for PostgreSQL.

Overall procedure

A typical migration process is as follows:

1. Prepare resources: Amazon Redshift, Amazon S3, ApsaraDB AnalyticDB for PostgreSQL, and Alibaba Cloud OSS.
2. Import the data in Redshift to S3.
3. Use OSSImport to import data files in .csv format from S3 to OSS.
4. In AnalyticDB for PostgreSQL, create the required objects, including schemas, tables, views, and functions.
5. Import data from the OSS external table into AnalyticDB for PostgreSQL.

The following figure shows the general workflow:



## Preparations on AWS

### Prepare information for accessing the S3 service

- Access Key ID and Secret Access Key
- The endpoint of the bucket in S3, for example, s3.ap-southeast-2.amazonaws.com
- The bucket name, for example, alibaba-hybrid-export

### Data format requirements for data export

- The data file must be in CSV format
- The size of the file to be exported cannot exceed 50 MB
- The order of the column values in the file is the same as the column order of the table creation statement
- Ideally, the number of files to be exported is the same as the number of segments in AnalyticDB for PostgreSQL or a multiple of the number of segments

### Recommended Redshift UNLOAD command option

The following UNLOAD command is the recommended format for the Redshift UNLOAD option, which is compatible with AnalyticDB for PostgreSQL:

```
unload (' select * from test ')
to ' s3 :// xxx - poc / test_expor t_ '
access_key _id '< Your access key id >'
secret_acc ess_key '< Your access key secret >'
DELIMITER AS ','
ADDQUOTES
ESCAPE
NULL AS ' NULL '
MAXFILESIZ E 50 mb ;
```

Specifically, in an UNLOAD command, the following options are recommended:

```
DELIMITER AS ','
ADDQUOTES
ESCAPE
NULL AS ' NULL '
MAXFILESIZ E 50 mb
```

### Get the DDL statement of the object in the Redshift database

Export all DDL statements from AWS Redshift including, but not limited to, schema, table, function, and view.

## Preparations on Alibaba Cloud

### Prepare information about the Alibaba Cloud RAM user account

- The RAM user account ID
- The RAM user account password
- The RAM user account AccessKeyId
- The RAM user account AccessKeySecret (paired with the preceding AccessKeyId to form an AccessKey)

### Prepare a bucket in OSS

Create a bucket in Alibaba Cloud OSS in the same region as the AWS S3 bucket (for example, the Sydney (ap-southeast-2) region).

After the OSS bucket is created, the Internet endpoint and VPC endpoint (that is, the intranet endpoint) of the bucket can be obtained from the OSS Console.

### Download and install OSSImport

- Create an ECS instance in the same area as the OSS bucket, with a network bandwidth of 100Mbps. In the following example, an instance running Windows is created.
- [Download and install the latest version of OSSImport](#).
- After you unzip the OSSImport package, the following folders and files are displayed.

```
ossimport
├── bin
│   ├── ossimport2.jar # The JAR package including master
│   │   , worker , tracker , and console modules
│   └── conf
│       ├── local_job.cfg # Standalone job configurat ion
│       └── file
│           └── sys.properties # Configurat ion file for the
│               system running
│   ├── console.bat # Windows command line , which can
│   │   run distribute d call - in tasks
│   ├── console.sh # Linux command line , which can
│   │   run distribute d call - in tasks
│   ├── import.bat # The configurat ion file for
│   │   one - click import and execution in Windows is the
│   │   data migration job configured in conf / local_job . cfg ,
│   │   including start , migration , validation , and retry
│   ├── import.sh # The configurat ion file of
│   │   one - click import and execution in Linux is the
│   │   data migration job configured in conf / local_job . cfg ,
│   │   including start , migration , validation , and retry
│   └── logs # Log directory
└── README.md # Descriptio n documentat ion . We
    recommend that you carefully read the documentat ion
    before using this feature
```

## Migrate data files from S3 to OSS using OSSImport

### Configure OSSImport

In the following example, OSSImport is used in the standalone deployment mode.

Edit `conf / local_job . cfg` file. In this example, only the parameter configuration that must be modified is provided. For detailed configuration instructions for OSSImport, see [Architecture and configuration](#).

```
srcType = s3
srcAccessKey = "your AWS Access Key ID "
srcSecretKey = "your AWS Access Key Secret "
srcDomain = s3 . ap - southeast - 2 . amazonaws . com
srcBucket = alibaba - hybrid - export
srcBucket =
destAccessKey = "your Alibaba Cloud Access Key ID "
destSecretKey = "your Alibaba Cloud Access Key Secret "
destDomain = http :// oss - ap - southeast - 2 - internal . aliyuncs . com
destBucket = alibaba - hybrid - export - 1
destPrefix =
isSkipExists = true
```

### Start the OSSImport Migration Task

In the OSSImport stand-alone deployment mode, you can start the migration task by executing `import . bat`.

### Monitor task status

During the data migration process, you can see the output in the command execution window. Additionally, you can review the usage of the network bandwidth through the resource manager.

In this example, because the ECS instance and the OSS bucket are deployed in the same region, the network speed between data uploading from the instance to the bucket is not limited. Notably, because data is downloaded from S3 to OSS through the Internet, the speed of data transfer between ECS and OSS is essentially the same as the speed of data transfer between S3 and ECS. In this case, the upload speed is limited by the download speed.

### Failed task retry (optional)

Sub-tasks may fail due to network or other reasons. Failure Retry only retries failed tasks, and will not retry the successful tasks. To retry failed tasks, execute `console . bat retry` in `cmd.exe` under the instance.

Check the files migrated to the OSS Bucket (optional)

You can check files through the OSS Console. We also recommend using the `ossbrowser` client tool to view and modify files in the bucket. [Download ossbrowser](#).

Scrub the csv files (optional)



Note:

If you want to scrub the data of your csv file, the following commands can be used.

- Replace `NULL` in the csv files with blank spaces.
- Replace `\,` with `,` in the csv files.

We recommend you perform data scrubbing operations on locally stored data. Specifically, you need to first download the data file to be scrubbed to ECS through the `ossbrowser` tool, and then scrub the data. After that, you need to upload the scrubbed data files to another newly created bucket (so as to be distinguished from the original CSV files). In later uses, when downloading the original csv files or uploading the scrubbed files, we recommend that `ossbrowser` use the intranet endpoint of OSS to reduce unnecessary charges to your account.

## DDL conversion from Redshift to AnalyticDB for PostgreSQL

This section describes the preparations required before creating a AnalyticDB for PostgreSQL database object. Specifically, it describes how to convert the DDL statements in Redshift syntax format to AnalyticDB for PostgreSQL syntax format. This section also describes the corresponding syntax conventions.

### CREATE SCHEMA

The following statement is an example that conforms to the PostgreSQL syntax format, which you can save as `create schema.sql`

```
CREATE SCHEMA schema1
  AUTHORIZATION xxxpoc ;
GRANT ALL ON SCHEMA schema1 TO xxxpoc ;
GRANT ALL ON SCHEMA schema1 TO public ;
COMMENT ON SCHEMA model IS ' for xxx migration poc
test ' ;

CREATE SCHEMA oss_extern al_table
```

```
AUTHORIZATION xxxpoc ;
```

## CREATE FUNCTION

Because Redshift provides some SQL functions of which the corresponding functions are not yet supported in HybridDB, you can choose to customize these functions or rewrite them. Specific examples are described as follows.

- Replace `CONVERT_TIMEZONE ( a , b , c )` with following code:

```
timezone ( b , timezone ( a , c ) )
```

- Replace `GETDATE()` with following code:

```
current_timestamp ( 0 ) : timestamp
```

- Replace and optimize user defined functions (UDFs).

For example, a SQL function of Redshift is as follows:

```
CREATE OR REPLACE FUNCTION public . f_jdate ( dt
timestamp without time zone )
RETURNS character varying AS
'
    from datetime import timedelta , datetime
    if dt . hour < 4 :
        d = timedelta ( days =- 1 )
        dt = dt + d
    return str ( dt . date ( ) )'
LANGUAGE plpythonu IMMUTABLE ;
COMMIT ;
```

Replace the preceding function with the following SQL statement:

```
to_char ( a - interval ' 4 hour ' , ' yyyy - mm - dd ' )
```

- Other Redshift standard SQL functions.

In your actual scenario, we recommend that you query the standard SQL function library of PostgreSQL at [Functions and Operators in PostgreSQL 8.2](#). In doing so, you can determine which functions you need to manually modify and implement so that they are compatible with AnalyticDB for PostgreSQL. The following is a list of commonly used functions:

- - [ISNULL\(\)](#)
- [DATEADD\(\)](#)
- [DATEDIFF\(\)](#)
- [REGEXP\\_COUNT\(\)](#)
- [LEFT\(\)](#)
- [RIGHT\(\)](#)

## CREATE TABLE

- Change compression encoding. AnalyticDB for PostgreSQL does not support the full list of [Redshift Compression Encoding](#). Compression encodings which are not supported are listed as follows:

- BYTEDICT
- DELTA
- DELTA32K
- LZO
- MOSTLY8
- MOSTLY16
- MOSTLY32
- RAW (no compression)
- RUNLENGTH
- TEXT255
- TEXT32K
- ZSTD

ENCODE XXX should be removed and replaced with following option in CREATE TABLE statement:

```
with ( COMPRESSTY PE = { ZLIB | QUICKLZ | RLE_TYPE | NONE } )
```

- Change distribution keys. Redshift supports three types of distribution keys. For more information, see [Distribution Styles](#). The following information indicates



the rules you need to apply to modify the distribution keys so that the keys are compatible with AnalyticDB for PostgreSQL.

- **DISTSTYLE EVEN:** Replace with `distribute d randomly`
- **DISTKEY:** Replace with `distribute d by ( colname1 ,...)`
- **ALL:** Remove (not supported)

**Change SORT key.** Replace the **COMPOUND** or **INTERLEAVED** options in Redshift sort key clause `[ COMPOUND | INTERLEAVE D ] SORTKEY ( column_name [, ...] )` with following clause:

```
with ( APPENDONLY = true , ORIENTATION = column )
sortkey ( volume );
```

### Example 1

The following statement is a **CREATE TABLE** statement that conforms to Redshift syntax:

```
CREATE TABLE schema1 . table1
(
  filed1  VARCHAR ( 100 ) ENCODE  lzo ,
  filed2  INTEGER  DISTKEY ,
  filed3  INTEGER ,
  filed4  BIGINT  ENCODE  lzo ,
  filed5  INTEGER ,
)
INTERLEAVE D  SORTKEY
(
  filed1 ,
  filed2
);
```

After conversion, the **CREATE TABLE** statement that conforms to the AnalyticDB for PostgreSQL syntax is as follows:

```
CREATE TABLE schema1 . table1
(
  filed1  VARCHAR ( 100 ) ,
  filed3  INTEGER ,
  filed5  INTEGER
)
WITH ( APPENDONLY = true , ORIENTATION = column , COMPRESSTYPE =
zlib )
DISTRIBUTE D  BY ( filed2 )
SORTKEY
(
  filed1 ,
  filed2
)
```

## Example 2

The following statement is a CREATE TABLE statement that conforms to Redshift syntax. It includes the ENCODE and SORTKEY options:

```
CREATE TABLE schema2 . table2
(
    filed1 VARCHAR ( 50 ) ENCODE lzo ,
    filed2 VARCHAR ( 50 ) ENCODE lzo ,
    filed3 VARCHAR ( 20 ) ENCODE lzo ,
)
DISTSTYLE EVEN
INTERLEAVE D SORTKEY
(
    filed1
);
```

After conversion, the CREATE TABLE statement that conforms to the AnalyticDB for PostgreSQL syntax is as follows:

```
CREATE TABLE schema2 . table2
(
    filed1 VARCHAR ( 50 ),
    filed2 VARCHAR ( 50 ),
    filed3 VARCHAR ( 20 ),
)
WITH ( APPENDONLY = true , ORIENTATIO N = column , COMPRESSTY PE
= zlib )
DISTRIBUTE D randomly
SORTKEY
(
    filed1
);
```

## CREATE VIEW

Similar to the CREATE TABLE statements in the preceding section, if you need to use a CREATE VIEW statement, you need to first convert the statement so that it conforms to the AnalyticDB for PostgreSQL syntax.

Create and Configure a AnalyticDB for PostgreSQL instance

For more information, see:

- [Create an instance](#)
- [Set up a whitelist](#)
- [Set up an account](#)

## Create Database Objects

Follow the instructions in [Connect to a AnalyticDB for PostgreSQL database](#), and use `psql` or `pgAdmin III 1.6.3` to connect to an instance.

Then, modify the DDL statements in Redshift syntax to DDL statements that conform to the AnalyticDB for PostgreSQL syntax, and then execute these DDL statements to create database objects.

### CREATE EXTERNAL TABLE

AnalyticDB for PostgreSQL supports parallel import from OSS and export to OSS through external tables (which is called the `gpossex` function). It can also compress external table files in `gzip` format to reduce the storage space and the costs. The `gpossex` function can read or write text and csv files, or text and csv files in `gzip` format. For more information, see [Migrate data by using an OSS external table](#).

### Import data by using INSERT INTO script

After external tables in OSS and database objects in AnalyticDB for PostgreSQL are created, you need to prepare an `INSERT` script to import data from the external tables to the target tables in AnalyticDB for PostgreSQL. Then, you need to save the `INSERT` script as `insert . sql`, and then execute this file.

The format of the `INSERT` statement is `INSERT INTO < TABLE NAME >`  
`SELECT * FROM < OSS EXTERNAL TABLE NAME >;`

#### Example:

```
INSERT INTO schema1 . table1 SELECT * FROM oss_external_table . table1 ;
```

After the import is completed, you can use `SELECT` statements to verify the imported data and compare them with the source data.

### Run a VACUUM script to defragment the database

After the external tables in OSS are imported into AnalyticDB for PostgreSQL, you need to defragment the database by running `VACUUM` script. Then, you need to save the `VACUUM` script as `vacuum . sql`, and then execute this file. For more information about `VACUUM`, see [VACUUM](#).