Alibaba Cloud AnalyticDB for PostgreSQL

Quick Start

Issue: 20190514

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.	 Danger: 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.	 Warning: 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.	 Notice: 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.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

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

Contents

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 commmand*.

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
CreateDBInstance	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 Deleteto delete an existing custom group.

Related API

API	Description
DescribeDBInstanceIPArrayList	Returns a list of IP addresses that are allowed to access the database instance.
ModifySecurityIps	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
CreateAccount	Creates an account.
DescribeAccounts	Returns the account information for a database.
ModifyAccountDescription	Modifies the description of the database.
ResetAccountPassword	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 environmen t 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
ModifyDBInstanceNetworkType	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*.

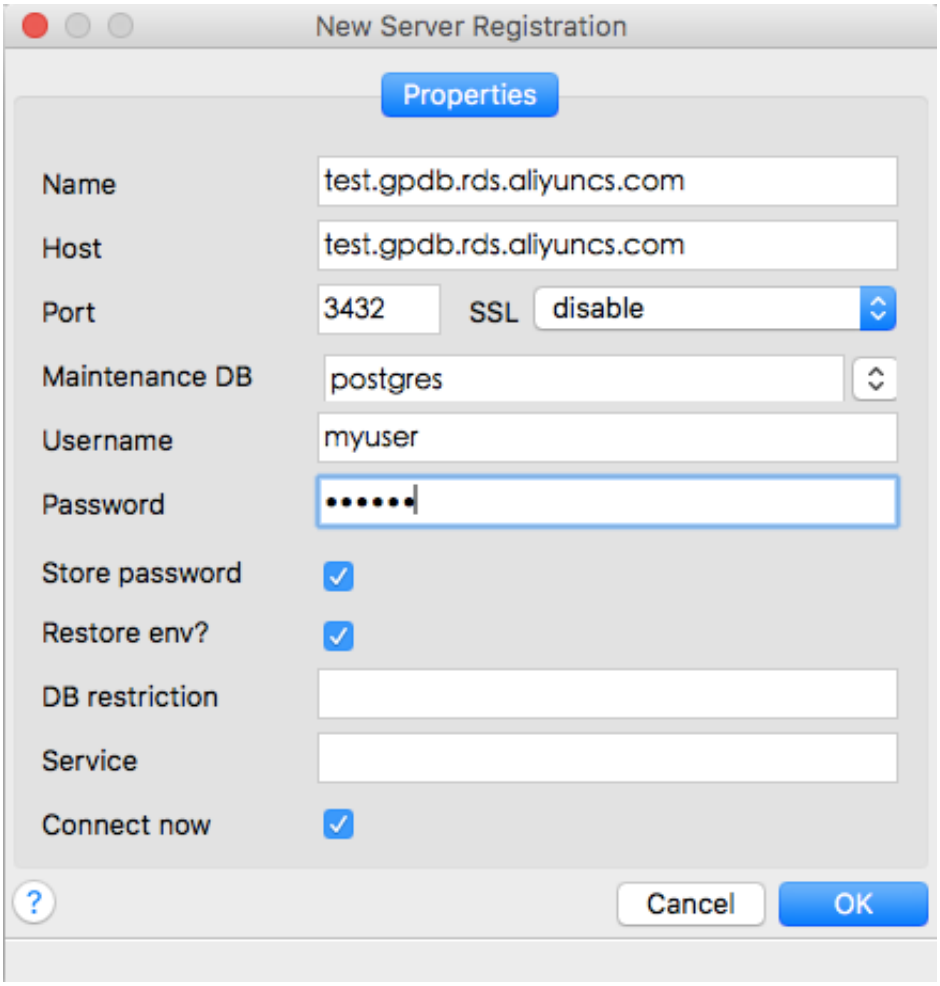
You can download pgAdmin III 1.6.3 from *PostgreSQL official website* . pgAdmin III 1.6.3 supports a variety of platforms, such as Windows, MacOS, and Linux.

> **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:



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 decompress ion. .

- 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 /
   5d87015609 abd330c68a 5402c1267f c86cbc9e1f . zip
   unzip 5d87015609 abd330c68a 5402c1267f c86cbc9e1f . zip
   cd gpdb - 5d87015609 abd330c68a 5402c1267f c86cbc9e1f
   ```

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 = gpdbaccoun t password
   = gpdbpasswo rd "
   ```

   · Specify parameters

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

   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 . DriverMana  ger ;
import   java . sql . ResultSet ;
import   java . sql . SQLExcepti  on ;
import   java . sql . Statement ;
public   class   gp_conn  {
    public   static   void   main ( String []  args ) {
        try  {
            Class . forName (" org . postgresql . Driver ");
            Connection   db  =  DriverMana  ger . getConnect  ion ("
jdbc : postgresql :// mygpdbpub . gpdb . rds . aliyuncs . com : 3432
/ postgres "," mygpdb "," mygpdb ");
            Statement   st  =  db . createStat  ement ();
            ResultSet   rs  =  st . executeQue  ry (" select  *  from
  gp_segment  _configura  tion ;");
            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  ( ClassNotFo  undExcepti  on   e ) {
             e . printStack  Trace ();
        }  catch  ( SQLExcepti  on   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 _configura tion ;'
   conn = psycopg2 . connect ( database =' gpdb ', user =' mygpdb
  ', password =' mygpdb ', host =' mygpdbpub . gpdb . rds .
  aliyuncs . 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 ]
 Descriptio  n      =  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 ]
 Descriptio  n      =  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 ]
 Descriptio  n  =  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                  |
+---------------------+
 SQLRowCoun  t   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
Parallel import from OSS or export to OSS	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.
Use Data Integration to synchronize data	Migrate data between Alibaba Cloud products	Use Data Integration to import data into or export data from AnalyticDB for PostgreSQL.
Migrate data from MySQL to AnalyticDB for PostgreSQL	Migrate data to AnalyticDB for PostgreSQL	Use the mysql2pgsql tool to import tables from local MySQL into AnalyticDB for PostgreSQL.
Migrate data from PostgreSQL to AnalyticDB for PostgreSQL	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
Migrate data to AnalyticDB for PostgreSQL by using the Copy command	Migrate data to AnalyticDB for PostgreSQL	Use the `\ COPY` command to import data from local text files into AnalyticDB for PostgreSQL.

5.2 Parallel import from OSS or export to OSS

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

The gpossext 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:

1. Distribute and store the data evenly in multiple OSS files. The number of files is preferably an integral multiple of the number of AnalyticDB for PostgreSQL data nodes (number of Segments).
2. Create a READABLE external table in the AnalyticDB for PostgreSQL database.
3. 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:

1. Create a WRITABLE external table in the AnalyticDB for PostgreSQL database.

2. 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 (' 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_tabl e ]] SEGMENT REJECT LIMIT
  count
      [ ROWS | PERCENT ] ]
CREATE WRITABLE EXTERNAL TABLE table_name
( column_nam e data_type [, ...] | LIKE other_tabl e )
LOCATION (' ossprotoco l ')
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 ]
ossprotoco l :
   oss :// oss_endpoi nt prefix = prefix_nam e
    id = userossid key = userosskey bucket = ossbucket
compressio ntype =[ none | gzip ] async =[ true | false ]
ossprotoco l :
   oss :// oss_endpoi nt dir =[ folder /[ folder /]...]/ file_name

    id = userossid key = userosskey bucket = ossbucket
compressio ntype =[ none | gzip ] async =[ true | false ]
ossprotoco l :
   oss :// oss_endpoi nt filepath =[ folder /[ folder /]...]/
file_name
```

```
    id = userossid   key = userosskey   bucket = ossbucket
compressio  ntype =[ none | gzip ]  async =[ true | false ]
```

Parameters description

Common parameters

- Protocol and endpoint: The format is `protocol  name : // oss_endpoi  nt` . 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/filenamexxx
      - 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_tab lename_uui d . x`. To be specific, the `uuid` is the generated int64 integer value (time stamp in microsecond), and the `x` is the node ID. AnalyticDB 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.
- compressiontype: The compression format of the exported files.
  - If specified to none (default value), it indicates that the exported files are not compressed.
  - If specified to gzip, it indicates that the exported format is gzip. Only the gzip compression format is supported.

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 '  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   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 '
         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   to   load   data
 create   table   example
        ( date   text ,  time   text ,  open   float ,
```

```
          high   float ,  low   float ,  volume   int )
          DISTRIBUTE  D   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   participat  es   in   the   work .
#  They   pull   data   from   the   OSS   in   parallel   and   then
   use   the   Redistribu  te   Motion   node   to   distribute   the
   hashed   data   to   correspond  ing   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 )
   ->   Redistribu  te   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   redistribu  tion .
 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 AnalyticDB 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 , ORIENTATIO N = column , COMPRESSTY PE = zlib , COMPRESSLE VEL = 5 , BLOCKSIZE = 1048576 )`. For details, see *Greenplum Database Tabulation Syntax Official Documentation*.**
- **The ossendpoint region must match the AnalyticDB for PostgreSQL region to ensure the data importing performance. We recommend that you configure the OSS and AnalyticDB 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 delimeter used to define columns:
  - If DELIMITER is included in the user data, the QUOTE parameter is required.
  - Recommended column delimeters 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 espace 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.

Table 5-1: 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*

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 Securitypage.
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:

```
{
  " configurat  ion ": {
    " 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
```

```
        " 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 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:

```
{
  " configurat  ion ": {
    " reader ": {
      " plugin ": " postgresql ",
      " parameter ": {
        " 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 .
        " 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
 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 ",// Target   table   name
        " truncate ":  true ,
        " partition ": " pt =${ bdp . system . bizdate }"// Partition
   informatio  n
      }
    },
    " setting ": {
      " speed ": {
        " mbps ":  5 ,// The   maximum   job   rate
        " concurrent ":  5 // Number   of   concurrent   jobs
      }
    }
  },
  " type ": " job ",
  " version ": " 1 . 0 "
}
```

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, querries 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:

```
./ mysql2pgsq  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:

```
./ mysql2pgsq 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:

```
./ pgsql2pgsq  l
```

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_tabl  e ] [ KEP ]
         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
srcAccessK  ey =" your   AWS   Access   Key   ID "
srcSecretK  ey =" your   AWS   Access   Key   Secret "
srcDomain = s3 . ap - southeast - 2 . amazonaws . com
srcBucket = alibaba - hybrid - export
srcBucket =
destAccess  Key =" your   Alibaba   Cloud   Access   Key   ID "
destSecret  Key =" your   Alibaba   Cloud   Access   Key   Secret "
destDomain = http :// oss - ap - southeast - 2 - internal . aliyuncs
. com
destBucket = alibaba - hybrid - export - 1
destPrefix =
isSkipExis  tFile = 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.*Dowload 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
  AUTHORIZAT  ION   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
```

```
AUTHORIZAT ION 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_TI MEZONE ( a , b , c )` with following code:

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

- Replace GETDATE() with following code:

```
current_ti mestamp ( 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 PostgreSQL8.2*. 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_nam e [, ...] ) ]` with following clause:

```
with ( APPENDONLY = true , ORIENTATIO N = 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 , ORIENTATIO N = column , COMPRESSTY PE =
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 gpossext function). It can also compress external table files in gzip format to reduce the storage space and the costs. The gpossext function can read or write text and csv files, or text and csv files in gzip format. For more information, see *Parallel import from OSS or export to OSS*.

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_extern
al_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*.