阿里云
分析型数据库PostgreSQL版

用户指南

文档版本：20190722

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 管理实例

1.1 创建实例

您可以通过如下两种方式购买或创建云数据库AnalyticDB for PostgreSQL实例：

- 在阿里云官网的 云数据库AnalyticDB for PostgreSQL购买页面 直接购买。
- 在阿里云AnalyticDB for PostgreSQL 数据库管理控制台新建实例。

为便于您在控制台上进行增减实例的操作，本文以通过阿里云AnalyticDB for PostgreSQL数据库管理控制台的方式为例，详细介绍创建AnalyticDB for PostgreSQL实例的操作步骤。

前提条件

- 已注册阿里云账号。若尚未注册，请前往阿里云官网进行注册。
- 阿里云账户余额大于等于 100 元。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 单击页面右上角的新建实例，进入实例购买页面。
3. 选择计费方式。
   - 按量付费：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。
   - 预付费：即包年包月计费方式，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。

   > 说明：
   > 按量付费实例可以转为包年包月实例。包年包月实例无法转为按量付费实例。

4. 设置以下参数。

参数	说明
地域	实例所在的地理位置。购买后无法更换地域。 - 请根据目标用户所在的地理位置就近选择地域，提升用户访问速度。 - 请确保实例与需要连接的ECS实例创建于同一个地域，否则它们无法通过内网互通，只能通过外网互通，无法发挥最佳性能。

参数	说明
可用区	可用区是地域中的一个独立物理区域，不同可用区之间没有实质性区别。 您可以选择将实例与ECS实例创建在同一可用区或不同的可用区。
引擎	目前仅有计算存储一体版的选项。
网络	· 经典网络：传统的网络类型。 · 专有网络（推荐）：也称为VPC（Virtual Private Cloud）。VPC是一种隔离的网络环境，安全性和性能均高于传统的经典网络。如果选择专有网络，你需要事先创建与实例在同一地域下的 VPC 和交换机，具体步骤请参考专有网络。
计算组规格	计算资源单位，不同的计算组规格有不同的存储空间和计算能力。关于规格详情，请参见AnalyticDB for PostgreSQL的规格总览。
计算组节点	所购买的“计算组”数量，最小单位为2个，计算组个数的增加可以线性地提升性能。

5. 完成设置后，单击立即购买。
6. 在确认订单页面，勾选《AnalyticDB for PostgreSQL服务协议》，然后单击去开通完成订单支付。
7. 用户可在实例列表页面查看新建实例。

> 说明:
> AnalyticDB for PostgreSQL数据库初始化需要一定时间，待实例列表中的实例运行状态显示为运行中，才可进行后续操作。

相关API

API	描述
CreateDBInstance	创建实例

1.2 设置白名单

在启用实例前，您必须先修改白名单。为保障数据库的安全稳定，请将需要访问数据库的 IP 地址或者 IP 段加入白名单。

背景信息

访问 AnalyticDB for PostgreSQL 数据库有如下三种场景：

· 外网访问 AnalyticDB for PostgreSQL 数据库。

- 内网访问 AnalyticDB for PostgreSQL 数据库。请确AnalyticDB for PostgreSQL 和 ECS 网络类型一致。
- 内外网同时访问 AnalyticDB for PostgreSQL 数据库。请确AnalyticDB for PostgreSQL 和 ECS 网络类型一致。

> 注意:
> 关于设置网络类型，请参见设置网络类型。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 选择目标实例所在地域。
3. 单击目标实例的 ID， 进入实例基本信息页面。
4. 在实例菜单栏中，选择数据安全性，进入数据安全性页面。
5. 在白名单设置标签页中，单击 default 白名单分组后的修改，进入修改白名单分组页面。

   > 注意:
   > 您也可以单击 default 白名单分组后的清空，删除默认分组中的白名单，然后单击添加白名单分组新建自定义分组。

6. 删除“组内白名单”中的默认白名单 127.0.0.1，然后填写自定义白名单。参数说明如下所示：
   - 分组名称：2~32 个字符，由小写字母、数字或下划线组成，开头需为小写字母，结尾需为字母或数字。默认分组不可修改，且不可删除。
   - 组内白名单：填写可以访问数据库的 IP 地址或者 IP 段，各 IP 地址或者 IP 段间用英文逗号分隔。
     - 白名单功能支持设置 IP 地址（如 10.10.10.1）或者 IP 段（如 10.10.10.0/24，表示 10.10.10.X 的 IP 地址都可以访问数据库）。
     - % 或者 0.0.0.0/0 为允许任何 IP 访问。

       > 注意:
       > 该设置将极大降低数据库安全性，如非必要请勿使用。

     - 新建实例设置了本地环回 IP 地址 127.0.0.1 为默认白名单，禁止任何外部 IP 访问本实例。
   - 加载 ECS 内网 IP：单击将显示同账号下的 ECS，可以快速添加 ECS 到白名单中。
7. 单击确定，添加白名单。

后续操作

正确使用白名单可以让 AnalyticDB for PostgreSQL 得到高级别的访问安全保护，建议您定期维护白名单。

后续操作中，您可以单击分组名称后的修改修改已有分组，或者单击删除删除已有的自定义分组。

相关API

API	描述
DescribeDBInstanceIPArrayList	查询允许访问实例的IP名单
ModifySecurityIps	修改白名单

1.3 设置账号

本文档将介绍如何在 AnalyticDB for PostgreSQL 的实例中创建账号及重置密码。

创建账号

前提条件

在使用云数据库 AnalyticDB for PostgreSQL 之前，需要AnalyticDB for PostgreSQL 实例中创建账号。

> 注意:
>
> - 初始账号创建后，无法删除该账号。
> - 用户无法在控制台创建其他账号，但是登录到数据库后可通过SQL语句创建其他账号。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 选择目标实例所在地域。
3. 单击目标实例的 ID，进入实例基本信息页面。
4. 在实例菜单栏中，选择账号管理，进入账号管理页面。
5. 单击创建初始账号，进入创建账号页面。
6. 填写数据库账号和密码，然后单击确定。
   - 数据库账号：2~16 个字符，由小写字母、数字或下划线组成，开头需为字母，结尾需为字母或数字，如 *user4example*。
   - 密码：8~32 个字符，由大写、小字、数字或特殊字符中的三类字符组成。
   - 确认密码：输入与密码一致的字段。

重置密码

在使用 AnalyticDB for PostgreSQL 过程中，如果忘记数据库账号密码，可以通过 AnalyticDB for PostgreSQL 数据库管理控制台 重新设置密码。

> 注意:
>
> 为保障数据安全，建议您定期更换密码。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 选择目标实例所在地域。
3. 单击目标实例的 ID，进入实例基本信息页面。
4. 在实例菜单栏中，选择账号管理，进入账号管理页面。
5. 单击需要管理账号后的重置密码，进入重置账户密码页面。
6. 输入新密码并确认新密码后，单击确定。

> 注意:
>
> 密码有 8~32 个字符，由大写、小字、数字或特殊字符中的三类字符组成。建议不要使用曾经用过的密码。

相关API

API	描述
CreateAccount	创建账户
DescribeAccounts	查询数据库账户信息
ModifyAccountDescription	修改数据库账户的备注名
ResetAccountPassword	重置账户密码

1.4 设置网络类型

阿里云数据库支持经典网络和专有网络两种网络类型。AnalyticDB for PostgreSQL 默认使用经典网络，如果您要使用专有网AnalyticDB for PostgreSQL 中的实例和专有网络必须在同一个地域。本章主要介绍两种网络类型的区别及设置方法。

背景信息

在阿里云平台上，经典网络和专有网络有如下区别：

- 经典网络：经典网络中的云服务在网络上不进行隔离，只能依靠云服务自身的白名单策略来阻挡非法访问。
- 专有网络（Virtual Private Cloud，简称 VPC）：专有网络帮助用户在阿里云上构建出一个隔离的网络环境。用户可以自定义专有网络里面的路由表、IP 地址范围和网关。用户可以通过专线或者 VPN 的方式将自建机房与阿里云专有网络内的云资源组合成一个虚拟机房，实现应用平滑上云。

操作步骤

1. 创建与目标 AnalyticDB for PostgreSQL 实例所在地域一致的专有网络，详细操作步骤请参见创建专有网络。
2. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
3. 选择目标实例所在地域。
4. 单击目标实例对应操作栏下的管理按钮，进入实例基本信息页面。
5. 在实例菜单栏中，选择数据库连接，进入数据连接页面。
6. 单击切换为专有网络，进入切换为专有网络选择页面。
7. 选择一个专有网络和虚拟交换机，然后单击确定。

> 注意:
> 切换为专有网络后，原内网地址将从经典网络切换到专有网络，经典网络下的 ECS 将无法访问专有网络下的 AnalyticDB for PostgreSQL 实例，原外网地址保持不变。

相关API

API	描述
ModifyDBInstanceNetworkType	切换网络类型

1.5 客户端访问实例

云数据库 AnalyticDB for PostgreSQL 完全兼容 PostgreSQL 8.2 的消息协议，可以直接使用支持 PostgreSQL 8.2 消息协议的工具，例如 DMS控制台、命令行psql、libpq、JDBC、ODBC、psycopg2、pgadmin III 等。

DMS 控制台

数据管理（Data Management Service，简称DMS）支持MySQL、SQL Server、PostgreSQL、PPAS、Petadata等关系型数据库，DRDS等OLTP数据库，AnalyticDB、DLA等OLAP数据库和MongoDB、Redis等NoSQL的数据库管理。它是一种

集数据管理、结构管理、用户授权、安全审计、数据趋势、数据追踪、BI图表、性能与优化和服务器管理于一体的数据管理服务。

本章节将为您介绍如何使用DMS登录云数据库 AnalyticDB for PostgreSQL。

1. 登录AnalyticDB for PostgreSQL 控制台。
2. 创建实例，具体操作请参见创建实例；若已完成创建操作，请单击目标实例ID进入实例详情页。
3. 创建账号，具体操作请参见设置账号。

    说明:
   该账号用于登录DMS，每个实例只能创建一个。
4. 单击实例详情页右上角登录数据库。
5. 在RDS数据库登录页面输入用户名和密码，单击登录。
6. 若有页面弹出如下提示，请根据当前情况选择相应设置。
   - 设置所有实例：当前用户下所有的实例都会被添加DMS的IP地址，以后不需要重复操作该步骤。
   - 设置本实例：只有当前登录的实例添加DMS的IP地址。
   - 不设置：不设置DMS的IP地址，该操作将无法使用DMS登陆数据库。

命令行 psql

psql 是 Greenplum 中比较常用的工具，提供了丰富的命令，其二进制文件在 Greenplum 安装后的 BIN 目录下。使用步骤如下所示：

1. 通过如下任意一种方式进行连接：

   · 连接串的方式

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

   · 指定参数的方式

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

   参数说明：

   - -h：指定主机地址。
   - -p：指定端口号。
   - -d：指定数据库（默认的数据库是 postgres），
   - -U：指定连接的用户。
   - 可以通过`psql --help`查看更多选项。在 psql 中，可以执行`\?`查看更多 psql 中支持的命令。

2. 输入密码，进入 psql 的 Shell 界面。psql的Shell界面如下：

   ```
   postgres=>
   ```

参考文档

· 关于 Greenplum 的 psql 的更多使用方法，请参见文档“psql”。
· AnalyticDB for PostgreSQL 也支持 PostgreSQL 的 psql 命令，使用时请注意细节上的差异。详情参见“PostgreSQL 8.3.23 Documentation — psql”。

下载方式

对于 RHEL（Red Hat Enterprise Linux）和 CentOS 版本 6 和 7 平台，可以通过以下地址进行下载，解压后即可使用：

· RHEL 6 或 CentOS 6 平台，请单击 HybridDB_client_package_el6 进行下载。
· RHEL 7 或 CentOS 7 平台，请单击 HybridDB_client_package_el7 进行下载。

Windows 及其它平台的客户端工具，请到 Pivotal 网站下载 AnalyticDB for PostgreSQL Client。

pgAdmin III

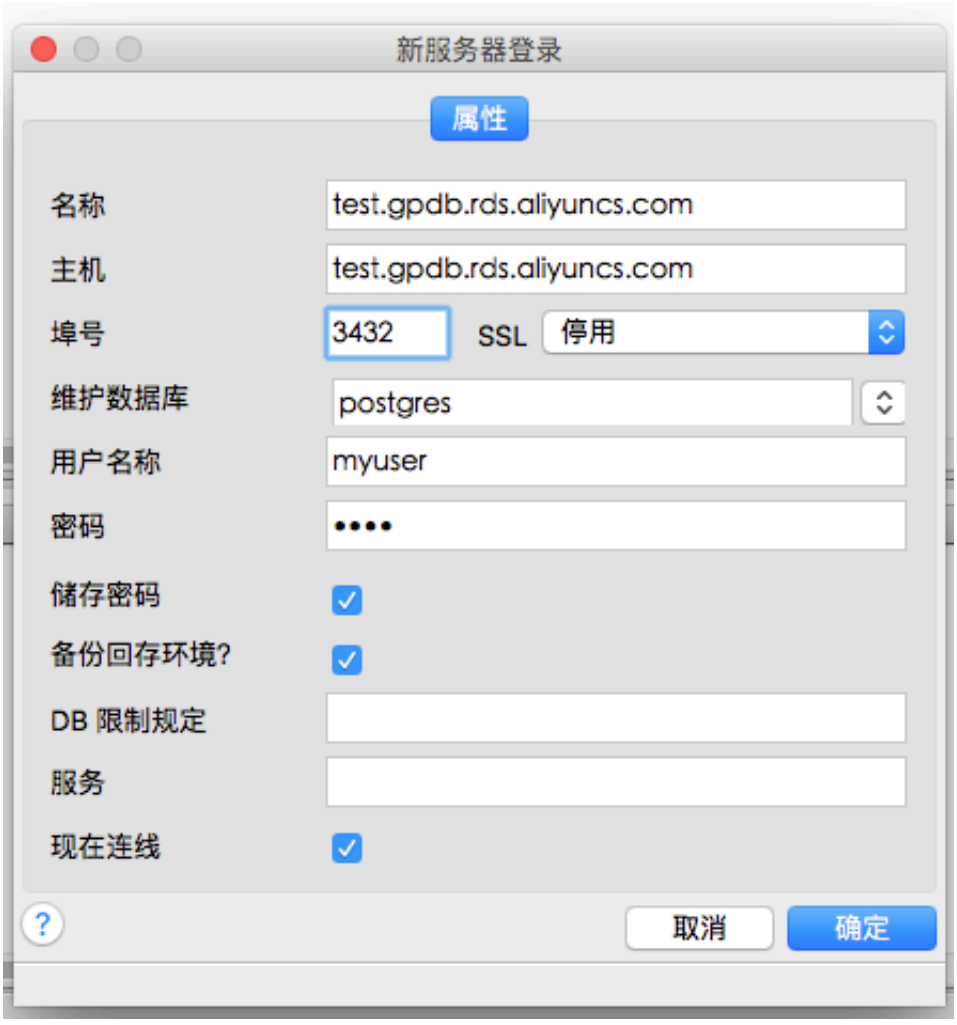
pgAdmin III 是 PostgreSQL 图形客户端，可以直接用于连接 AnalyticDB for PostgreSQL。详情参见 官网。

您可以从 PostgreSQL 官网 下载 pgAdmin III 1.6.3。pgAdmin III 1.6.3 支持各种平台，例如 Windows、MacOS 和 Linux。其它图形客户端，详情参见 图形客户端工具。

> **注意:**
>
> AnalyticDB for PostgreSQL 与 PostgreSQL 8.2 版本兼容，因此必须使用 pgAdmin III 1.6.3 或之前的版本才能连AnalyticDB for PostgreSQL（pgAdmin 4 当前不支持）。

操作步骤

1. 下载安装 pgAdmin III 1.6.3 或之前的版本。
2. 选择文件 > 新增服务器，进入配置连接窗口。
3. 填写配置信息，如下图所示：



4. 单击确定，即可连接到 AnalyticDB for PostgreSQL。

JDBC

用户需要使用 PostgreSQL 官方提供的JDBC。下载方法如下：

- 单击 这里，下载 PostgreSQL 的官方 JDBC，下载之后加入到环境变量中。

- 也可采用 Greenplum 官网提供的工具包，详情请参见“Greenplum Database 4.3 Connectivity Tools for UNIX”。

代码示例

```
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.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

详细文档，请参见“The PostgreSQL JDBC Interface”。

Python

Python 连接 Greenplum 和 PostgreSQL 采用的库是 psycopg2。使用步骤如下：

1. 安装 psycopg2。在 CentOS 下，有如下三种安装方法：

   - 执行如下命令：`yum -y install python-psycopg2`
   - 执行如下命令：`pip install psycopg2`
   - 从源码安装：

```
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. 安装后，设置 PYTHONPATH 环境变量，之后就可以引用，如下所示：

```
 import psycopg2
 sql = 'select * from gp_segment_configuration;'
 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()
```

   会得到类似以下的结果：

```
(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 是 PostgreSQL 数据库的 C 语言接口，用户可在 C 程序中通过 libpq 库访问 PostgreSQL 数据库并进行数据库操作。在安装了 Greenplum 或者 PostgreSQL 之后，在其 lib 目录下可以找到其静态库和动态库。

相关案例请参见 这里，此处不再列举。

关于 libpq 详情，请参见“PostgreSQL 9.4.10 Documentation — Chapter 31. libpq - C Library”。

ODBC

PostgreSQL 的 ODBC 是基于 LGPL（GNU Lesser General Public License）协议的开源版本，可以在 PostgreSQL 官网下载。

操作步骤

1. 安装驱动。

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

2. 查看驱动配置。

```
cat /etc/odbcinst.ini
# Example driver definitions
# Driver from the postgresql-odbc package
# Setup from the unixODBC package
[PostgreSQL]
Description     = ODBC for PostgreSQL
Driver          = /usr/lib/psqlodbcw.so
Setup           = /usr/lib/libodbcpsqlS.so
Driver64        = /usr/lib64/psqlodbcw.so
Setup64         = /usr/lib64/libodbcpsqlS.so
FileUsage       = 1
# Driver from the mysql-connector-odbc package
# Setup from the unixODBC package
[MySQL]
Description     = ODBC for MySQL
Driver          = /usr/lib/libmyodbc5.so
Setup           = /usr/lib/libodbcmyS.so
Driver64        = /usr/lib64/libmyodbc5.so
Setup64         = /usr/lib64/libodbcmyS.so
FileUsage       = 1
```

3. 配置 DSN，将如下代码中的`****`改成对应的连接信息。

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

4. 测试连通性。

```
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. ODBC 已连接上实例，将应用连接 ODBC 即可，具体操作请参见 这里 和 C# 连接到 PostgreSQL。

其他参考信息

图形客户端工具

AnalyticDB for PostgreSQL 用户可以直接使用 Greenplum 支持的其它图形化客户端工具，包括

- SQL Workbench
- Navicat Premium
- Navicat For PostgreSQL
- pgadmin III (1.6.3)

Greenplum 客户端参考

Greenplum 官网也提供了一个安装包，包含 JDBC、ODBC 和 libpq，用户可方便地安装和使用，详情参见Greenplum 官方文档。

参考文档

- Pivotal Greenplum 官方文档
- PostgreSQL psqlODBC
- PostgreSQL ODBC 编译
- Greenplum ODBC 下载
- Greenplum JDBC 下载

1.6 申请外网地址

如果您的应用部署在与您的AnalyticDB for PostgreSQL实例在同一地域且网络类型相同的ECS上，则无需申请外网地址。如果您的应用部署在与您的AnalyticDB for PostgreSQL实例在不同地域或网络类型不同的ECS或者阿里云以外的系统上，需申请外网地址，用于对接应用。

说明:

只要在同一地域内（可用区可以不同）且网络类型相同的实例，就可以通过内网互通。

应用场景

内外网地址的使用场景如下所示：

- 单独使用内网地址：
  - 适用于应用部署在与您的AnalyticDB for PostgreSQL实例在同一地域的ECS上且实例与ECS的网络类型相同时。
- 单独使用外网地址：
  - 适用于应用部署在与您的AnalyticDB for PostgreSQL实例在不同地域的ECS上时。
  - 适用于应用部署在阿里云以外的系统上时。
- 同时使用内外网地址：
  - 适用于应用中有些模块部署在与您的AnalyticDB for PostgreSQL实例同一地域且网络类型相同的ECS上，有些模块又部署在与您的AnalyticDB for PostgreSQL实例在不同地域的ECS上时。
  - 适用于应用中有些模块部署在与您的AnalyticDB for PostgreSQL实例在同一地域且网络类型相同的ECS上，有些模块又部署在阿里云以外的系统上时。

注意事项

- 在访问数据库前，您需要将访问数据库的IP地址或者IP段加入白名单，操作请参见#unique_21。
- 使用外网地址会降低实例的安全性，请谨慎选择。为了获得更快的传输速率和更高的安全级别，建议您将应用迁移到与您的AnalyticDB for PostgreSQL在同一地域的ECS上。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 在实例列表上方选择实例所在地域。
3. 找到目标实例，在操作栏中，单击管理。
4. 在基本信息页面，单击申请外网地址，跳转到数据库连接页面。也可直接单击左侧导航的数据库连接。
5. 在数据库连接页面，单击申请外网地址。
6. 在弹出的对话框中，单击确定，生成外网地址。

生成外网地址后，可在数据库连接页面，通过释放外网地址按键释放外网地址。

相关API

API	描述
AllocateInstancePublicConnection	申请外网地址

1.7 释放外网地址

申请外网地址之后，如果网络环境发生变化，用户不再需要用外网地址连接实例，可在AnalyticDB for PostgreSQL控制台释放外网地址。释放外网地址之后，请注意更改涉及该地址的应用设置。

在释放外网地址之前，请先阅读以下应用场景。

应用场景

内外网地址的使用场景如下所示：

- 单独使用内网地址：
  - 适用于应用部署在与您的AnalyticDB for PostgreSQL实例在同一地域的ECS上且实例与ECS的网络类型相同时。
- 单独使用外网地址：
  - 适用于应用部署在与您的AnalyticDB for PostgreSQL实例在不同地域的ECS上时。
  - 适用于应用部署在阿里云以外的系统上时。
- 同时使用内外网地址：
  - 适用于应用中有些模块部署在与您的AnalyticDB for PostgreSQL实例同一地域且网络类型相同的ECS上，有些模块又部署在与您的AnalyticDB for PostgreSQL实例在不同地域的ECS上时。
  - 适用于应用中有些模块部署在与您的AnalyticDB for PostgreSQL实例在同一地域且网络类型相同的ECS上，有些模块又部署在阿里云以外的系统上时。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 在实例列表上方选择实例所在地域。
3. 找到目标实例，在操作栏中，单击管理。
4. 单击导航栏左侧的数据库连接。
5. 在数据库连接页面，单击释放外网地址。

   如未申请外网地址，数据库连接页面仅有申请外网地址按键。

6. 在弹出的对话框中，单击确定，释放外网地址。

相关API

API	描述
ReleaseInstancePublicConnection	释放实例的外网连接串

1.8 升级实例规格

在使用云数据库 AnalyticDB for PostgreSQL 过程中，随着您的数据量和计算量的动态增长，一些计算资源如CPU、磁盘、内存以及数据处理节点数量将成为数据处理速度的瓶颈。为了支持实例的动态扩展，AnalyticDB for PostgreSQL 提供在线升级实例规格的功能，目前暂不支持对实例的降级操作。本文介绍了升级实例规格的相关操作。

查看当前实例规格

AnalyticDB for PostgreSQL 实例规格包括 计算组规格 和计算组数量 ，各种计算组规格详细说明见实例规格 。

您可以通过以下的步骤来查看当前的实例规格：

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 选择要操作实例所在的地域。例如，华东1。
3. 单击目标实例右侧操作栏中的管理按钮，进入基本信息页面。

   在基本信息页面的配置信息中显示了当前的实例规格，包括计算组规格、计算组详情、计算组数量、计算资源汇总。

AnalyticDB for PostgreSQL 目前提供两类规格：

- 高性能，规格名称以gpdb.group.segsdx开始，提供更好的I/O能力，带来更高的性能。
- 高容量，规格名称以gpdb.group.seghdx开始，提供更大、更实惠的空间，满足更高的存储需要。

升级实例规格

当确认需要升级实例规格后，您可以按照下面的步骤进行实例规格的升级：

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 选择要操作实例所在的地域。例如，华东1。
3. 单击目标实例右侧操作栏中的升级按钮，进入确认订单页面。

4. 选择合适的计算组规格和计算组数量，点击去开通按钮。

   目前，AnalyticDB for PostgreSQL 支持不同的计算组规格和计算组数量组合，详情参见规则组合速查表。在选择新的计算组规格和计算组数量组合时，要满足以下约束：

   - 新旧计算组规格必须为同一类计算组规格，而且新的计算组规格 >= 旧的计算组规格
   - 如果新的计算组规格 = 旧的计算组规格，新的计算组数量 > 旧的计算组数量

   在选择计算组规格和计算组数量的时候，除了满足以上约束，您还需要对自己的数据量和计算量进行评估，从而选择合适的计算组规格和计算组数量组合，详情可参考如何选择实例规格 。

   > 注意:
   > 根据您数据量的不同，实例规格升级的过程大约需要30分钟到3个小时的时间。在此过程中，为了保证数据的一致性，您实例只对外提供只读服务，并且会闪断两次，请您提前做出调整。当升级实例规格结束，对应实例恢复运行中状态，您可以正常访问数据库，而且实例的数据库内核版本自动升级为最新。

在完成了上述操作之后，您可以回到控制台，查看目标实例的运行状态。如果实例规格升级操作完成，则实例运行状态为运行中，否则为升降级中。

1.9 修改连接地址

AnalyticDB for PostgreSQL支持修改实例连接地址。例如，您改用其他AnalyticDB for PostgreSQL实例后，无需对应用程序进行修改，只需将新实例的连接地址修改为原实例的连接地址。

前提条件

- 实例正常运行。
- 实例已设置了白名单。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 在实例列表上方选择实例所在地域。
3. 找到目标实例，在操作栏中，单击管理。
4. 单击左侧导航的数据库连接。
5. 在数据库连接页面，单击修改连接地址。

6. 在弹出的对话框中，单击连接类型下拉框，选择网络类型。

   可选的网络类型有内网地址和外网地址，其中只有在申请了外网地址后，才有外网地址这个选项。

7. 在随后出现的连接地址和端口输入框中，填入相关信息后，单击确定。

   设置成功后刷新页面，页面显示新的连接地址。

相关API

API	描述
ModifyDBInstanceConnectionString	修改实例连接地址和端口

1.10 按量付费转包年包月

购买按量付费的实例后，您可以根据需求将实例改变成包年包月的计费方式。

注意事项

- 包年包月的实例无法转成按量付费的实例，在您进行计费方式的转变前请务必考虑清楚，以免造成资源浪费。
- 在计费周期内，包年包月的实例只支持升级配置，不支持降级配置或者释放。
- 变更实例计费方式成功后，实例会即刻按照包年包月的实例计费。
- 按量付费实例变更为包年包月时会产生一个新购订单，您必须完成该订单的支付流程，计费方式的变更才能生效。若未支付或未成功支付，您的订单管理页面将会出现未完成订单，之后您将无法新购实例或变更实例计费类型。

变更条件

- 实例计费类型为按量付费，且实例状态为运行中。
- 实例没有未完成的变更计费方式（即新购）的订单。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 找到目标实例，在操作栏中，单击按量转包年包月。
3. 在确认订单页面，选择购买时长，勾选《AnalyticDB for PostgreSQL服务协议》，单击去开通。
4. 在订单信息页面，确认订单信息，单击确认支付。

说明:

此时系统会生成一个转包年包月的订单。若该订单未支付或作废，将导致您无法进行新购实例或转包年包月的操作。您可以在订单管理页面支付或作废该订单。

1.11 重启实例

为了更好地满足用户的需求，AnalyticDB for PostgreSQL 会不断更新数据库内核版本。在实例创建时，默认使用的是最新版本的数据库内核。当新的版本发布之后，您可以通过重启实例来更新数据库内核版本，从而使用新版本中扩展的功能。本文介绍了重启实例的方法。

 注意:

重启实例时，实例将不可用，请在业务低谷时执行该操作。

操作步骤

您可以按照下面的步骤来重启实例：

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 选择要操作实例所在的地域。例如，华东1。
3. 单击目标实例右侧操作栏中的管理 按钮，进入基本信息页面。
4. 单击页面右上角的重启实例，并在确认框中单击确定。如果您绑定了手机，还需要进行手机验证码验证。

 注意:

重启过程一般耗时3到30分钟，在此过程中该实例不能对外提供服务，请您提前做出调整。当实例重启结束，对应实例恢复运行中状态，您可以正常访问数据库。

完成了上述操作后，您可以回到控制台查看目标实例的运行状态。如果实例重启操作完成，则实例运行状态为运行中，否则为重启中。

相关API

API	描述
RestartDBInstance	重启实例

1.12 开启SQL审计

AnalyticDB for PostgreSQL控制台提供SQL审计功能，您可通过该功能查看SQL明细、定期审计SQL。开通SQL审计功能后，实例性能不会受到影响。

背景信息

SQL审计会统计所有DML和DDL操作信息，这些信息是系统通过网络协议分析所得，在SQL查询量较大的时候可能会丢失少量记录。

注意事项

- SQL审计记录的保存时间为30天。
- 实例默认关闭SQL审计功能。开启该功能后，实例会产生额外费用，详细收费标准请参见云数据库AnalyticDB for PostgreSQL 详细价格信息。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 在实例列表上方选择实例所在地域。
3. 找到目标实例，在操作栏中，单击管理。
4. 单击左侧导航的数据安全性。
5. 选择SQL审计页签，单击开启SQL审计。
6. 在弹出的对话框中单击确定，开启SQL审计功能。
   - 在开始SQL审计功能后，您可通过时间、DB、User、关键字等条件查询SQL信息。
   - 您还可在SQL审计页签，通过关闭SQL审计按键关闭已经开通的SQL审计。

相关API

API	描述
ModifySQLCollectorPolicy	开启或关闭指定实例的SQL采集功能

1.13 释放实例

根据业务要求，您可以手动释放按量付费的实例。

前提条件

实例类型为按量付费实例。

> **说明:**
> 您可以随时释放按量付费的实例；包年包月的实例不能主动删除或释放，到期后自动释放。

操作步骤

1. 登录云数据库AnalyticDB for PostgreSQL管理控制台。
2. 在实例列表上方选择实例所在地域。

3. 找到目标实例，在操作栏中，单击管理。
4. 在基本信息页面，单击运行状态右侧的释放按键。
5. 在弹出的对话框中勾选您确定释放实例？，然后单击确定，释放实例。

> **注意:**
> 释放后的实例不可恢复，请谨慎操作。

相关API

API	描述
DeleteDBInstance	释放实例

2 定义数据库对象

2.1 操作指引

本章节介绍AnalyticDB for PostgreSQL中的数据定义语言（DDL）以及如何创建和管理数据库对象。

- 创建和删除数据库
- 创建和管理Schema
- 创建和管理表
- 表分布策略
- 表存储格式
- 分区表定义
- 使用索引
- 管理视图

2.2 创建和删除数据库

数据库(Database)是表、索引、视图、存储过程、操作符的集合。用户可以在一个AnalyticDB for PostgreSQL实例中创建多个数据库，但是客户端程序一次只能连接上并且访问一个数据库，无法跨数据库进行查询。

创建数据库

使用`CREATE DATABASE`命令创建一个新的数据库，命令如下：

```
CREATE DATABASE <dbname> [ [WITH] [OWNER [=] <dbowner>]
                        [ENCODING [=] <encoding>]
```

说明：

- <dbname>：待创建的数据库名称。
- <dbowner>：拥有新数据库的数据库用户名，默认为执行该命令的用户。
- <encoding>：在新数据库中使用的字符集编码。指定一个字符串常量（例如 'SQL_ASCII'），一个整数编码号，默认为utf-8。

示例：

```
CREATE DATABASE mygpdb;
```

删除数据库

使用DROP DATABASE命令删除一个数据库。它会移除该数据库的元数据并且删除该数据库在磁盘上的目录及其中包含的数据，命令如下：

```
DROP DATABASE <dbname>
```

说明：

<dbname>：待删除的数据库名称。

示例：

```
DROP DATABASE mygpdb;
```

更多信息

详情请参考Pivotal Greenplum 官方文档。

2.3 创建和管理Schema

Schema是数据库的命名空间，它是一个数据库内部的对象（表、索引、视图、存储过程、操作符）的集合。Schema在每个数据库中是唯一的。每个数据库都一个名为public的默认Schema。

如果用户没有创建任何Schema，对象会被创建在public schema中。所有的该数据库角色（用户）都在默认的public schema中拥有CREATE和USAGE特权。

创建Schema

使用CREATE SCHEMA命令来创建一个新的Schema，命令如下：

```
CREATE SCHEMA <schema_name> [AUTHORIZATION <username>]
```

说明：

- <schema_name>：schema名称。
- <username>：如指定authorization username，则创建的schema属于该用户。否则，属于执行该命令的用户。

示例：

```
CREATE SCHEMA myschema;
```

设置Schema搜索路径

数据库的search_path用于配置参数设置Schema的搜索顺序。

使用ALTER DATABASE命令可以设置搜索路径。例如：

```
ALTER DATABASE mydatabase SET search_path TO myschema, public,
pg_catalog;
```

您也可以使用ALTERROLE命令为特定的角色（用户）设置search_path。例如：

```
ALTER ROLE sally SET search_path TO myschema, public, pg_catalog;
```

查看当前Schema

使用current_schema()函数可以查看当前的Schema。例如：

```
SELECT current_schema();
```

您也可以使用SHOW命令查看当前的搜索路径。例如：

```
SHOW search_path;
```

删除Schema

使用DROP SCHEMA命令删除一个空的Schema。例如：

```
DROP SCHEMA myschema;
```

说明：

默认情况下，Schema它必须为空才可以删除。

删除一个Schema连同其中的所有对象（表、数据、函数等等），可以使用：

```
DROP SCHEMA myschema CASCADE;
```

更多信息

详情请参考Pivotal Greenplum 官方文档。

2.4 创建和管理表

AnalyticDB forPostgreSQL数据库中的表与任何一种关系型数据库中的表类似，不同的是表中的行被分布在不同Segment上，表的分布策略决定了在不同Segment上面的分布情况。

创建表

CREATE TABLE命令用于创建一个表，创建表时可以定义以下内容：

- 表的列以及数据类型
- 表的约束
- 表分布键定义
- 表存储格式
- 分区表定义

使用`CREATE TABLE`命令创建表，格式如下：

```
CREATE TABLE table_name (
[ { column_name data_type [ DEFAULT default_expr ]
   [column_constraint [ ... ]
]
   | table_constraint
   | LIKE other_table [{INCLUDING | EXCLUDING}
                      {DEFAULTS | CONSTRAINTS}] ...}
   [, ... ] ]
   )
   [ WITH ( storage_parameter=value [, ... ] )
   [ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]

   [ PARTITION BY partition_type (column)

       [ SUBPARTITION BY partition_type (column) ]
          [ SUBPARTITION TEMPLATE ( template_spec ) ]
       [...]
    ( partition_spec )
        | [ SUBPARTITION BY partition_type (column) ]
          [...]
    ( partition_spec
      [ ( subpartition_spec
           [(...)]
       ) ]
    )
```

示例：

示例中的建表语句创建了一个表，使用trans_id作为分布键，并使用了RANGE分区功能。

```
CREATE TABLE sales (trans_id int,
  date date,
  amount decimal(9,2),
  region text)
  DISTRIBUTED BY (trans_id)
  PARTITION BY RANGE(date)
```

```
  (start (date '2018-01-01') inclusive
   end (date '2019-01-01') exclusive every (interval '1 month'),
   default partition outlying_dates);
```

表的约束定义

您可以在列和表上定义约束来限制表中的数据，但是有以下一些限制：

- CHECK约束引用的列只能在其所在的表中。
- UNIQUE和PRIMARY KEY约束必须包含分布键列，UNIQUE和PRIMARY KEY约束不支持追加优化表和列存表。
- 允许FOREIGN KEY 约束，但实际上并不会做外键约束检查。
- 分区表上的约束必须应用到所有的分区表上，不能只应用于部分分区表。

约束命令格式如下：

```
UNIQUE ( column_name [, ... ] )
   | PRIMARY KEY ( column_name [, ... ] )
   | CHECK ( expression )
   | FOREIGN KEY ( column_name [, ... ] )
            REFERENCES table_name [ ( column_name [, ... ] ) ]
            [ key_match_type ]
            [ key_action ]
            [ key_checking_mode ]
```

检查约束（Check Constraints)

检查约束（Check Constraints）指定列中的值必须满足一个布尔表达式，例如：

```
CREATE TABLE products
            ( product_no integer,
              name text,
              price numeric CHECK (price > 0) );
```

非空约束（Not-Null Constraints)

非空约束（Not-Null Constraints）指定列不能有空值，例如：

```
CREATE TABLE products
       ( product_no integer NOT NULL,
         name text NOT NULL,
         price numeric );
```

唯一约束（Unique Constraints)

唯一约束（Unique Constraints）确保一列或者一组列中包含的数据对于表中所有的行都是唯一的。包含唯一约束的表必须是哈希分布，并且约束列需要包含分布键列，例如：

```
CREATE TABLE products
```

```
        ( product_no integer UNIQUE,
          name text,
          price numeric)
      DISTRIBUTED BY (product_no);
```

主键（Primary Keys）

主键约束（Primary Keys Constraints）是一个UNIQUE约束和一个 NOT NULL约束的组合。包含主键约束的表必须是哈希分布，并且约束列需要包含分布键列。如果一个表具有主键，这个列（或者这一组列）会被默认选中为该表的分布键，例如：

```
CREATE TABLE products
       ( product_no integer PRIMARY KEY,
         name text,
         price numeric)
      DISTRIBUTED BY (product_no);
```

临时表

临时表（Temporary Table）会在会话结束时自动删除，或选择性地在当前事务结束的时候删除。创建临时表的命令如下：

```
CREATE TEMPORARY TABLE table_name(…)
    [ON COMMIT {PRESERVE ROWS | DELETE ROWS | DROP}]
```

说明：

临时表的行为在事务块结束时的行为可以通过上述语句中的ON COMMIT来控制。

- PRESERVE ROWS：在事务结束时候保留数据，这是默认的行为。
- DELETE ROWS：在每个事务块结束时，临时表的所有行都将被删除。
- DROP：在当前事务结束时，会删除临时表。

示例：

创建一个临时表，事务结束时候删除该临时表。

```
CREATE TEMPORARY TABLE temp_foo (a int, b text) ON COMMIT DROP;
```

更多信息

详情请参考Pivotal Greenplum 官方文档。

2.5 表分布键定义

选择表分布策略

AnalyticDB for PostgreSQL 支持两种数据在节点间的分布方式，按分布键的哈希值（HASH）分布或随机（RANDOMLY）分布。`CREATE TABLE`的子句`DISTRIBUTED BY ( column, [ ... ])` 指定数据按分布键的哈希值在节点（segment）间分布，子句`DISTRIBUTED RANDOMLY`指定数据按循环的方式均匀分配在各节点（Segment）间。

示例：

示例中的建表语句创建了一个哈希（Hash）分布的表，数据将按分布键的哈希值被分配到对应的 Segment 数据节点。

```
CREATE TABLE products (name varchar(40),
                       prod_id integer,
                       supplier_id integer)
                       DISTRIBUTED BY (prod_id);
```

示例中的建表语句创建了一个随机（Randomly）分布的表，数据被循环着放置到各个 Segment 数据节点。

```
CREATE TABLE random_stuff (things text,
                           doodads text,
                           etc text)
                           DISTRIBUTED RANDOMLY;
```

对于按分布键的简单查询，包括 UPDATE/DELETE 等语句，AnalyticDB for PostgreSQL 具有按节点的分布键进行数据节点裁剪的功能，例如products表使用prod_id作为分布键，以下查询只会被发送到满足prod_id=101的segment上执行，从而极大提升该 SQL 执行性能：

```
select * from products where prod_id = 101;
```

表分布键选择原则

合理规划分布键，对表查询的性能至关重要，有以下原则需要关注：

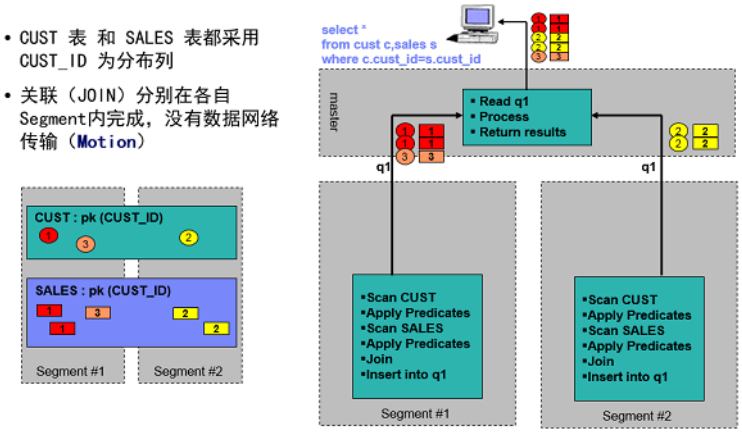
- 尽量选择数据分布均匀的列或者多个列：若选择的分布列数值分布不均匀，则可能导致数据倾斜。某些Segment分区节点存储数据多(查询负载高)。根据木桶原理，时间消耗会卡在数据多的节点上。故不应选择bool类型，时间日期类型数据作为分布键。
- 尽量选择经常需要JOIN的列作为分布键，可以实现图一所示本地关联（Collocated JOIN），即当关联键和分布键一致时，可以在 Segment分区节点内部完成JOIN，否则需要将一个表进行重分布（Redistribute motion）来实现图二所示重分布关联（Redistributed Join）或者广播小表(Broadcast motion)来实现图三所示广播关联（Broadcast Join），后两种方式都会有较大的网络开销。

- 尽量选择高频率出现的查询条件列作为分布键，从而可能实现按分布键做节点 segment 的裁剪。
- 若未指定分布键，默认表的主键为分布键，若表没有主键，则默认将第一列当做分布键。
- 分布键可以被定义为一个或多个列。例如：

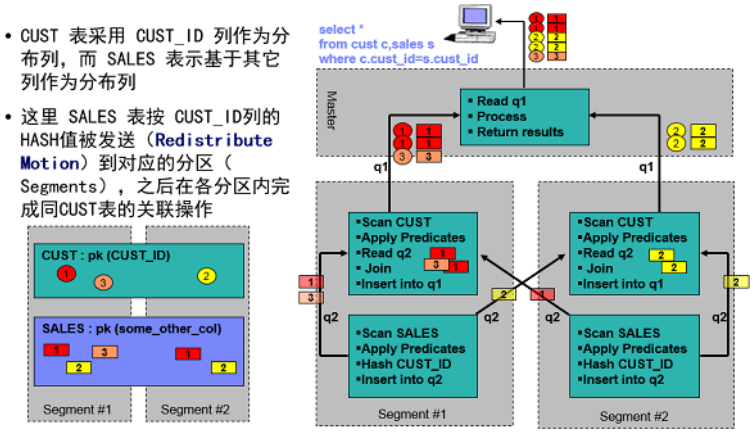
```
create table t1(c1 int, c2 int) distributed by (c1,c2);
```

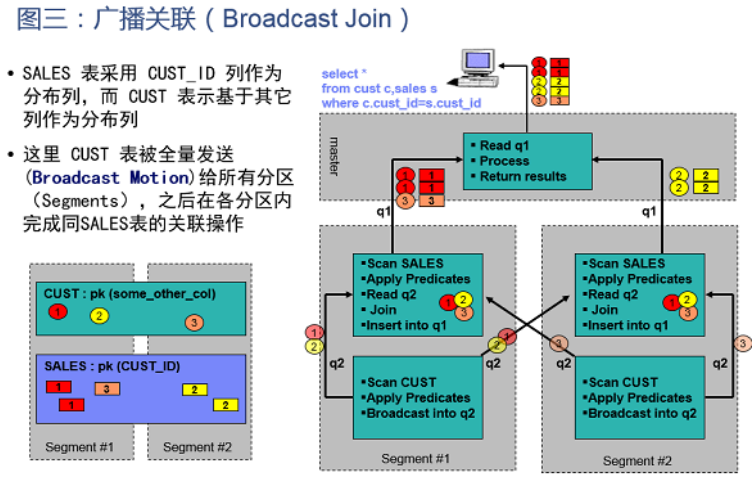
- 谨慎选择随机分布DISTRIBUTED RANDOMLY，这将使得上述本地关联，或者节点裁剪不可能实现。

图一：本地关联 (Collocated JOIN)



图二：重分布关联（Redistuted Join）





表分布键的约束

- 分布键的列不能被更新（UPDATE）。
- 主键和唯一键必须包含分布键。例如：

```
create table t1(c1 int, c2 int, primary key (c1)) distributed by (c2
);
```

说明：

由于主键c1不包含分布键c2，所以建表语句返回失败。

```
ERROR: PRIMARY KEY and DISTRIBUTED BY definitions incompatible
```

- Geometry类型和用户自定义数据类型不能作为分布键。

数据倾斜检查和处理

当某些表上的查询性能差时，可以查看是否是分区键设置不合理造成了数据倾斜，例如：

```
create table t1(c1 int, c2 int) distributed by (c1);
```

您可以通过下述语句来查看表的数据倾斜情况。

```
select gp_segment_id,count(1) from  t1 group by 1 order by 2 desc;
 gp_segment_id | count
---------------+--------
             2 | 131191
             0 |     72
             1 |     68
```

```
(3 rows)
```

如果发现某些segment上存储的数据明显多于其他segment，该表存在数据倾斜。建议选取数据分布平均的列作为分布列，比如通过ALTER TABLE命令更改C2为分布键。

```
alter table t1 set distributed by (c2);
```

表t1的分布键被改为c2，该表的数据按照c2被重新分布，数据不再倾斜。

2.6 表存储格式

AnalyticDB for PostgreSQL支持多种存储格式。当您创建一个表时，可以选择表的存储格式为行存表或者列存表。

行存表

默认情况下，AnalyticDB for PostgreSQL 创建的是行存表（Heap Table），使用的 PostgreSQL 堆存储模型。行存表适合数据更新较频繁的场景，或者采用INSERT方式的实时写入的场景，同时当行存表建有于B-Tree索引时，具备更好的点查询数据检索性能。

示例：

下述语句创建了一个默认堆存储类型的行存表。

```
CREATE TABLE foo (a int, b text) DISTRIBUTED BY (a);
```

列存表

列存表（Column-Oriented Table）的按列存储格式，数据访问只会读取涉及的列，适合少量列的数据查询、聚集等数据仓库应用场景，在此类场景中，列存表能够提供更高效的I/O。但列存表不适合频繁的更新操作或者大批量的INSERT写入场景，这时其效率较低。列存表的数据写入建议采用COPY等批量加载方式。列存表可以提供平均 3-5倍的较高数据压缩率。

示例：

列存表必须是追加优化表。例如，要创建一个列存表，必须指定为 "appendonly=true“ 。

```
CREATE TABLE bar (a int, b text)
    WITH (appendonly=true, orientation=column)
    DISTRIBUTED BY (a);
```

压缩

压缩主要用于列存表或者追加写（"appendonly=true“）的行存表，有以下两种类型的压缩可用。

· 应用于整个表的表级压缩。

· 应用到指定列的列级压缩。用户可以为不同的列应用不同的列级压缩算法。

目前AnalyticDB forPostgreSQL支持zlib和RLE_TYPE压缩算法（如果指定QuickLZ压缩算法，内部会使用zlib算法替换），RLE_TYPE算法只适用于列存表。

示例：

创建一个使用zlib压缩且压缩级别为5的追加优化表。

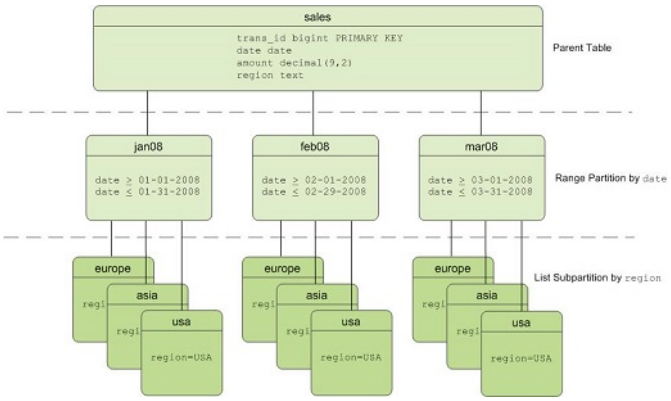
```
CREATE TABLE foo (a int, b text)
   WITH (appendonly=true, compresstype=zlib, compresslevel=5);
```

2.7 分区表定义

将大表定义为分区表，从而将其分成较小的存储单元，根据查询条件，会只扫描满足条件的分区而避免全表扫描，从而显著提升查询性能。

支持的表分区类型

· 范围（RANGE）分区：基于一个数值型范围划分数据，例如按着日期区间定义。
· 值（LIST）分区：基于一个值列表划分数据，例如按着 城市属性定义。
· 多级分区表：上述两种类型的多级组合。



上图为一个多级分区表设计实例，一级分区采用按月的区间（Range）分区，二级分区采用按地区的值（List）分区设计。

创建范围(RANGE)分区表

用户可以通过给出一个START值、一个END值以及一个定义分区增量值的子句让数据库自动产生分区。默认情况下，START值总是被包括在内而END值总是被排除在外，例如：

```
CREATE TABLE sales (id int, date date, amt decimal(10,2))
DISTRIBUTED BY (id)
PARTITION BY RANGE (date)
( START (date '2016-01-01') INCLUSIVE
   END (date '2017-01-01') EXCLUSIVE
```

```
    EVERY (INTERVAL '1 day') );
```

也可以创建一个按数字范围分区的表，使用单个数字数据类型列作为分区键列，例如：

```
CREATE TABLE rank (id int, rank int, year int, gender char(1), count
int)
DISTRIBUTED BY (id)
PARTITION BY RANGE (year)
( START (2006) END (2016) EVERY (1),
  DEFAULT PARTITION extra );
```

创建值（LIST)分区表

一个按列表分区的表可以使用任意允许等值比较的数据类型列作为它的分区键列。对于列表分区，您必须为每一个用户想要创建的分区（列表值）声明一个分区说明，例如：

```
CREATE TABLE rank (id int, rank int, year int, gender
char(1), count int )
DISTRIBUTED BY (id)
PARTITION BY LIST (gender)
( PARTITION girls VALUES ('F'),
  PARTITION boys VALUES ('M'),
  DEFAULT PARTITION other );
```

创建多级分区表

支持创建多级的分区表。下述建表语句创建了具有三级表分区的表。一级分区在year字段上使用RANGE分区，二级分区在month字段上做RANGE分区，三级分区在region上做了LIST分区。

```
CREATE TABLE sales (id int, year int, month int, day int,
region text)
DISTRIBUTED BY (id)
PARTITION BY RANGE (year)
  SUBPARTITION BY RANGE (month)
  SUBPARTITION TEMPLATE (
        START (1) END (13) EVERY (1),
        DEFAULT SUBPARTITION other_months )
  SUBPARTITION BY LIST (region)
    SUBPARTITION TEMPLATE (
        SUBPARTITION usa VALUES ('usa'),
        SUBPARTITION europe VALUES ('europe'),
        SUBPARTITION asia VALUES ('asia'),
        DEFAULT SUBPARTITION other_regions)
( START (2008) END (2016) EVERY (1),
  DEFAULT PARTITION outlying_years);
```

分区定义的粒度

通常分区表的定义都涉及到粒度问题，比如按时间分区，究竟是按天，按周，按月等。粒度越细，每张表的数据就越少，但是分区的数量就越多，反之亦然。关于分区的数量，没有绝对的标准，一般分区的数量在 200 左右已经算是比较多了。分区表数目过多，会有多方面的影响，比如查询优化器生成执行计划较慢，同时很多维护工作也会变慢，比如VACUUM等。

注意:

请对多级分区格外谨慎，因为分区文件的数量可能会增长得非常快。例如，如果一个表被按照月和城市划分并且有24个月以及1,00个城市，那么表分区的总数就是2400。特别对于列存表，会把每一列存在一个物理表中，因此如果这个表有100个列，系统就需要为该表管理十多万个文件。

分区表查询优化

AnalyticDB for PostgreSQL 支持分区表的分区裁剪功能，根据查询条件会只扫描所需的数据分区而避免扫描整个表的全部内容，提升查询性能。例如对于如下查询：

```
select * from sales
  where year = 2008
    and  month = 1
    and  day = 3
    and region = 'usa';
```

由于查询条件落在一级分区2008的二级子分区1的三级子分区 'usa' 上，查询只会扫描读取这一个三级子分区数据。

更多信息

详情请参考Pivotal Greenplum 官方文档。

2.8 使用索引

索引类型

AnalyticDB for PostgreSQL支持B-树索引，GiST索引和位图索引，不支持Hash索引和GIN索引。

说明:

B-树索引是默认的索引类型。位图索引（Bitmap Index）为每一个键值都存储一个位图，位图索引提供了和常规索引相同的功能但索引空间大大减少。对于区分值区间在100至100,000之间的列位图索引表现最好。

创建索引

使用`CREATE INDEX`命令在表上创建一个索引。

示例：

在employee表的gender列上创建一个B-树索引。

```
CREATE INDEX gender_idx ON employee (gender);
```

在films表中的title列上创建一个位图索引。

```
CREATE INDEX title_bmp_idx ON films USING bitmap (title);
```

重建索引

使用`REINDEX INDEX`命令重建索引。

示例：

重建索引my_index。

```
REINDEX INDEX my_index;
```

重建my_table表上的所有索引。

```
REINDEX TABLE my_table;
```

删除索引

使用`DROP INDEX`命令删除一个索引。

示例：

```
DROP INDEX title_idx;
```

说明：

在加载大量数据时，先删除所有索引并载入数据，然后重建索引会更快。

索引的选择原则

- 用户的查询负载。

  索引能改进查询返回单一记录或者非常小的数据集的性能，例如OLTP类型查询。
- 压缩表。

  在被压缩过的追加优化表上，索引也可以提高返回一个目标行集合的查询的性能，只有必要的行才会被解压。
- 避免在频繁更新的列上建立索引。

  在一个被频繁更新的列上建立索引会增加该列被更新时所要求的写操作数据量。

- 创建选择性高的B-树索引。

  例如，如果一个表有1000行并且一个列中有800个不同的值，则该索引的选择度为0.8，索引的选择性会比较高。唯一索引的选择度总是1.0。
- 为低选择度的列使用位图索引。对于区分值区间在100至100,000之间的列位图索引表现最好。
- 索引在连接中用到的列。

  在被用于频繁连接的一个列（例如一个外键列）上的索引能够提升连接性能，因为这让查询优化器有更多的连接方法可以使用。
- 索引在谓词中频繁使用的列。

  频繁地在WHERE子句中被引用的列是索引的首选。
- 避免创建重叠的索引。

  例如在多列索引中，具有相同前导列的索引是冗余的。
- 批量载入前删掉索引。

  对于载入大量数据到一个表中，请考虑先删掉索引并且在数据装载完成后重建它们，这常常比更新索引更快。
- 测试并且比较使用索引和不使用索引的查询性能。

  只有被索引列的查询性能有提升时才增加索引。

更多信息

详情请参考Pivotal Greenplum 官方文档。

2.9 使用视图

视图允许用户保存常用的或者复杂的查询。视图没有物理存储，当用户访问时，视图会作为一个子查询运行。

创建视图

使用`CREATE VIEW`命令创建一个查询的视图。

示例：

```
CREATE VIEW myview AS SELECT * FROM products WHERE kind = 'food';
```

说明：

视图会忽略存储在视图中的ORDERBY以及SORT操作。

删除视图

使用`DROP VIEW`命令删除一个视图。

示例：

```
DROP VIEW myview;
```

更多信息

详情请参考Pivotal Greenplum 官方文档。

3 数据库查询和操作

3.1 数据类型

AnalyticDB for PostgreSQL支持丰富的数据类型，您还可以使用CREATE TYPE命令定义新的数据类型。

AnalyticDB for PostgreSQL内建的数据类型

下表显示了AnalyticDB for PostgreSQL内建的数据类型。

名称	别名	存储大小	范围	描述
bigint	int8	8 bytes	-9223372036854775808 到9223372036854775807	大范围整数
bigserial	serial8	8 bytes	1 到 9223372036854775807	大的自动增量整数
bit [(n)]		n bits	bit 常量	固定长度位串
bit varying [(n)]	varbit	bit实际长度	bit 常量	可变长度位串
boolean	bool	1 byte	true/false, t/f, yes/no, y/n, 1/0	布尔值（true / false）
box		32 bytes	((x1,y1),(x2,y2))	平面中的矩形框 - 分配键列中不允许。
bytea		1 byte + binary string	八进制数序列	可变长度二进制字符串
character [(n)]	char [(n)]	1 byte + n	n长度字符串	定长的空白填充。
character varying [(n)]	varchar [(n)]	1 byte + string size	n长度字符串	受限的可变长度。
cidr		12 or 24 bytes		IPv4和IPv6网络
circle		24 bytes	<(x,y),r> (中心点和半径)	平面的圆 - 不允许在分配键列中。

名称	别名	存储大小	范围	描述
date		4 bytes	4713 BC - 294,277 AD	日历日期（年，月，日）
decimal [(p, s)]	numeric [(p, s)]	variable	无限制	用户指定的精度，精确
double precision	float8	8 bytes	15位数字精度	可变精度，不精确
	float			
inet		12 or 24 bytes		IPv4和IPv6主机和网络
integer	int, int4	4 bytes	-2.1E+09到 + 2147483647	通常选择整数类型
interval [(p)]		12 bytes	-178000000 years - 178000000 years	时间跨度
json		1 byte + json size	Json字符串	不受限制的可变长度
lseg		32 bytes	((x1,y1),(x2,y2))	平面中的线段 - 分配键列中不允许。
macaddr		6 bytes		MAC 地址
money		8 bytes	-92233720368547758.08 到 + 92233720368547758.07	货币金额
path		16+16n bytes	[(x1,y1),...]	平面上的几何路径 - 分布关键列中不允许。
point		16 bytes	(x,y)	平面上的几何点 - 分布关键列中不允许。
polygon		40+16n bytes	((x1,y1),...)	在平面中封闭的几何路径 - 分配关键列中不允许。
real	float4	4 bytes	6位数字精度	可变精度，不准确
serial	serial4	4 bytes	1 到 2147483647	自动增量整数
smallint	int2	2 bytes	-32768 到 +32767	小范围整数

名称	别名	存储大小	范围	描述
text		1 byte + string size	变长字符串	变量无限长
time [(p)] [without time zone]		8 bytes	00:00:00[.000000] - 24:00:00[.000000]	时间只有一天
time [(p)] with time zone	timetz	12 bytes	00:00:00+1359 - 24:00:00-1359	时间只有一天，带时区
timestamp [(p)] [without time zone]		8 bytes	4713 BC - 294,277 AD	日期和时间
timestamp [(p)] with time zone	timestamptz	8 bytes	4713 BC - 294,277 AD	日期和时间，带时区
xml		1 byte + xml size	任意长度xml	变量无限长

更多信息

详情请参考Pivotal Greenplum 官方文档。

3.2 SQL语法参考

本文介绍AnalyticDB for PostgreSQL支持的SQL命令及语法参考。

- ABORT
- ALTER AGGREGATE
- ALTER CONVERSION
- ALTER DATABASE
- ALTER DOMAIN
- ALTER EXTERNAL TABLE
- ALTER FUNCTION
- ALTER GROUP
- ALTER INDEX
- ALTER OPERATOR
- ALTER RESOURCE QUEUE
- ALTER ROLE
- ALTER SCHEMA

- ALTER SEQUENCE
- ALTER TABLE
- ALTER TYPE
- ALTER USER
- ANALYZE
- BEGIN
- CHECKPOINT
- CLOSE
- CLUSTER
- COMMENT
- COMMIT
- COPY
- CREATE AGGREGATE
- CREATE CAST
- CREATE CONVERSION
- CREATE DATABASE
- CREATE DOMAIN
- CREATE EXTENSION
- CREATE EXTERNAL TABLE
- CREATE FUNCTION
- CREATE GROUP
- CREATE INDEX
- CREATE LIBRARY
- CREATE OPERATOR
- CREATE RESOURCE QUEUE
- CREATE ROLE
- CREATE RULE
- CREATE SCHEMA
- CREATE SEQUENCE
- CREATE TABLE
- CREATE TABLE AS
- CREATE TYPE
- CREATE USER

- CREATE VIEW
- DEALLOCATE
- DECLARE
- DELETE
- DROP AGGREGATE
- DROP CAST
- DROP CONVERSION
- DROP DATABASE
- DROP DOMAIN
- DROP EXTENSION
- DROP EXTERNAL TABLE
- DROP FUNCTION
- DROP GROUP
- DROP INDEX
- DROP LIBRARY
- DROP OPERATOR
- DROP OWNED
- DROP RESOURCE QUEUE
- DROP ROLE
- DROP RULE
- DROP SCHEMA
- DROP SEQUENCE
- DROP TABLE
- DROP TYPE
- DROP USER
- DROP VIEW
- END
- EXECUTE
- EXPLAIN
- FETCH
- GRANT
- INSERT
- LOAD

- LOCK
- MOVE
- PREPARE
- REASSIGN OWNED
- REINDEX
- RELEASE SAVEPOINT
- RESET
- REVOKE
- ROLLBACK
- ROLLBACK TO SAVEPOINT
- SAVEPOINT
- SELECT
- SELECT INTO
- SET
- SET ROLE
- SET SESSION AUTHORIZATION
- SET TRANSACTION
- SHOW
- START TRANSACTION
- TRUNCATE
- UDPATE
- VACUUM
- VALUES

ABORT

终止当前事务。

```
ABORT [WORK | TRANSACTION]
```

更多信息请参考ABORT。

ALTER AGGREGATE

改变聚集函数的定义。

```
ALTER AGGREGATE name ( type [ , ... ] ) RENAME TO new_name
ALTER AGGREGATE name ( type [ , ... ] ) OWNER TO new_owner
```

```
ALTER AGGREGATE name ( type [ , ... ] ) SET SCHEMA new_schema
```

更多信息请参考ALTER AGGREGATE。

ALTER CONVERSION

修改转换的定义。

```
ALTER CONVERSION name RENAME TO newname
ALTER CONVERSION name OWNER TO newowner
```

更多信息请参考ALTER CONVERSION。

ALTER DATABASE

修改数据库属性。

```
ALTER DATABASE name [ WITH CONNECTION LIMIT connlimit ]
ALTER DATABASE name SET parameter { TO | = } { value | DEFAULT }
ALTER DATABASE name RESET parameter
ALTER DATABASE name RENAME TO newname
ALTER DATABASE name OWNER TO new_owner
```

更多信息请参考ALTER DATABASE。

ALTER DOMAIN

改变域的定义。

```
ALTER DOMAIN name { SET DEFAULT expression | DROP DEFAULT }
ALTER DOMAIN name { SET | DROP } NOT NULL
ALTER DOMAIN name ADD domain_constraint
ALTER DOMAIN name DROP CONSTRAINT constraint_name [RESTRICT | CASCADE]
ALTER DOMAIN name OWNER TO new_owner
ALTER DOMAIN name SET SCHEMA new_schema
```

更多信息请参阅ALTER DOMAIN。

ALTER EXTERNAL TABLE

改变外部表的定义。

```
ALTER EXTERNAL TABLE name RENAME [COLUMN] column TO new_column
ALTER EXTERNAL TABLE name RENAME TO new_name
ALTER EXTERNAL TABLE name SET SCHEMA new_schema
ALTER EXTERNAL TABLE name action [, ... ]
```

更多信息请参考ALTER EXTERNAL TABLE。

ALTER FUNCTION

改变函数的定义。

```
ALTER FUNCTION name ( [ [argmode] [argname] argtype [, ...] ] )
    action [, ... ] [RESTRICT]
```

```
ALTER FUNCTION name ( [ [argmode] [argname] argtype [, ...] ] )
    RENAME TO new_name
ALTER FUNCTION name ( [ [argmode] [argname] argtype [, ...] ] )
    OWNER TO new_owner
ALTER FUNCTION name ( [ [argmode] [argname] argtype [, ...] ] )
    SET SCHEMA new_schema
```

更多信息请参考ALTER FUNCTION。

ALTER GROUP

改变角色名字或者成员信息。

```
ALTER GROUP groupname ADD USER username [, ... ]
ALTER GROUP groupname DROP USER username [, ... ]
ALTER GROUP groupname RENAME TO newname
```

更多信息请参考ALTER GROUP。

ALTER INDEX

改变索引的定义。

```
ALTER INDEX name RENAME TO new_name
ALTER INDEX name SET TABLESPACE tablespace_name
ALTER INDEX name SET ( FILLFACTOR = value )
ALTER INDEX name RESET ( FILLFACTOR )
```

更多信息请参考ALTER INDEX。

ALTER OPERATOR

改变操作符的定义。

```
ALTER OPERATOR name ( {lefttype | NONE} , {righttype | NONE} )
    OWNER TO newowner
```

更多信息请参考ALTER OPERATOR。

ALTER RESOURCE QUEUE

修改资源队列的限制。

```
ALTER RESOURCE QUEUE name WITH ( queue_attribute=value [, ... ] )
```

更多信息请参考ALTER RESOURCE QUEUE。

ALTER ROLE

```
ALTER ROLE name RENAME TO newname

ALTER ROLE name SET config_parameter {TO | =} {value | DEFAULT}

ALTER ROLE name RESET config_parameter

ALTER ROLE name RESOURCE QUEUE {queue_name | NONE}
```

```
ALTER ROLE name [ [WITH] option [ ... ] ]
```

更多信息请参考ALTER ROLE。

ALTER SCHEMA

改变模式的定义。

```
ALTER SCHEMA name RENAME TO newname

ALTER SCHEMA name OWNER TO newowner
```

更多信息请参考ALTER SCHEMA。

ALTER SEQUENCE

改变序列生成器的定义。

```
ALTER SEQUENCE name [INCREMENT [ BY ] increment]
     [MINVALUE minvalue | NO MINVALUE]
     [MAXVALUE maxvalue | NO MAXVALUE]
     [RESTART [ WITH ] start]
     [CACHE cache] [[ NO ] CYCLE]
     [OWNED BY {table.column | NONE}]
ALTER SEQUENCE name SET SCHEMA new_schema
```

更多信息请参考ALTER SEQUENCE。

ALTER TABLE

改变的表的定义。

```
ALTER TABLE [ONLY] name RENAME [COLUMN] column TO new_column

ALTER TABLE name RENAME TO new_name

ALTER TABLE name SET SCHEMA new_schema

ALTER TABLE [ONLY] name SET
     DISTRIBUTED BY (column, [ ... ] )
   | DISTRIBUTED RANDOMLY
   | WITH (REORGANIZE=true|false)

ALTER TABLE [ONLY] name action [, ... ]

ALTER TABLE name
   [ ALTER PARTITION { partition_name | FOR (RANK(number))
   | FOR (value) } partition_action [...] ]
   partition_action
```

更多信息请参考ALTER TABLE。

ALTER TYPE

改变数据类型的定义。

```
ALTER TYPE name
   OWNER TO new_owner | SET SCHEMA new_schema
```

更多信息请参考ALTER TYPE。

ALTER USER

修改数据库角色（用户）的定义。

```
ALTER USER name RENAME TO newname

ALTER USER name SET config_parameter {TO | =} {value | DEFAULT}

ALTER USER name RESET config_parameter

ALTER USER name [ [WITH] option [ ... ] ]
```

更多信息请参考ALTER USER。

ANALYZE

收集关于数据库的数据。

```
ANALYZE [VERBOSE] [ROOTPARTITION [ALL] ]
   [table [ (column [, ...] ) ]]
```

更多信息请参考 ANALYZE。

BEGIN

启动事务块。

```
BEGIN [WORK | TRANSACTION] [transaction_mode]
      [READ ONLY | READ WRITE]
```

更多信息请参考BEGIN 。

CHECKPOINT

强制事务记录检查点。

```
CHECKPOINT
```

更多信息请参考CHECKPOINT。

CLOSE

关闭游标。

```
CLOSE cursor_name
```

更多信息请参考CLOSE 。

CLUSTER

根据索引对磁盘上的堆存储表进行物理重新排序。不是推荐使用该操作。

```
CLUSTER indexname ON tablename

CLUSTER tablename

CLUSTER
```

更多信息请参考CLUSTER。

COMMENT

定义或者修改对一个对象的注释。

```
COMMENT ON
{ TABLE object_name |
  COLUMN table_name.column_name |
  AGGREGATE agg_name (agg_type [, ...]) |
  CAST (sourcetype AS targettype) |
  CONSTRAINT constraint_name ON table_name |
  CONVERSION object_name |
  DATABASE object_name |
  DOMAIN object_name |
  FILESPACE object_name |
  FUNCTION func_name ([[argmode] [argname] argtype [, ...]]) |
  INDEX object_name |
  LARGE OBJECT large_object_oid |
  OPERATOR op (leftoperand_type, rightoperand_type) |
  OPERATOR CLASS object_name USING index_method |
  [PROCEDURAL] LANGUAGE object_name |
  RESOURCE QUEUE object_name |
  ROLE object_name |
  RULE rule_name ON table_name |
  SCHEMA object_name |
  SEQUENCE object_name |
  TABLESPACE object_name |
  TRIGGER trigger_name ON table_name |
  TYPE object_name |
  VIEW object_name }
IS 'text'
```

更多信息请参考COMMENT。

COMMIT

提交当前事务。

```
COMMIT [WORK | TRANSACTION]
```

更多信息请参考COMMIT。

COPY

在文件和表之间拷贝数据。

```
COPY table [(column [, ...])] FROM {'file' | STDIN}
     [ [WITH]
       [BINARY]
       [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]
       SEGMENT REJECT LIMIT count [ROWS | PERCENT] ]

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

更多信息请参考COPY 。

CREATE AGGREGATE

定义一个新的聚集函数。

```
CREATE [ORDERED] AGGREGATE name (input_data_type [ , ... ])
      ( SFUNC = sfunc,
        STYPE = state_data_type
        [, PREFUNC = prefunc]
        [, FINALFUNC = ffunc]
        [, INITCOND = initial_condition]
        [, SORTOP = sort_operator] )
```

更多信息请参考CREATE AGGREGATE。

CREATE CAST

定义一个新的CAST。

```
CREATE CAST (sourcetype AS targettype)
       WITH FUNCTION funcname (argtypes)
       [AS ASSIGNMENT | AS IMPLICIT]

CREATE CAST (sourcetype AS targettype) WITHOUT FUNCTION
       [AS ASSIGNMENT | AS IMPLICIT]
```

更多信息请参考CREATE CAST。

CREATE CONVERSION

定义一个新的编码转换。

```
CREATE [DEFAULT] CONVERSION name FOR source_encoding TO
     dest_encoding FROM funcname
```

更多信息请参考CREATE CONVERSION。

CREATE DATABASE

创建一个新的数据库。

```
CREATE DATABASE name [ [WITH] [OWNER [=] dbowner]
                     [TEMPLATE [=] template]
                     [ENCODING [=] encoding]
                     [CONNECTION LIMIT [=] connlimit ] ]
```

更多信息请参考CREATE DATABASE。

CREATE DOMAIN

定义一个新的域。

```
CREATE DOMAIN name [AS] data_type [DEFAULT expression]
       [CONSTRAINT constraint_name
       | NOT NULL | NULL
       | CHECK (expression) [...]]
```

更多信息请参考CREATE DOMAIN。

CREATE EXTENSION

在数据库中注册一个EXTENSION。

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
  [ WITH ] [ SCHEMA schema_name ]
           [ VERSION version ]
           [ FROM old_version ]
           [ CASCADE ]
```

更多信息请参考CREATE EXTENSION。

CREATE EXTERNAL TABLE

定义一张外部表。

```
CREATE [READABLE] EXTERNAL TABLE tablename
( columnname datatype [, ...] | LIKE othertable )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
            [( [HEADER]
               [DELIMITER [AS] 'delimiter' | 'OFF']
               [NULL [AS] 'null string']
               [ESCAPE [AS] 'escape' | 'OFF']
               [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
               [FILL MISSING FIELDS] )]
          | 'CSV'
            [( [HEADER]
               [QUOTE [AS] 'quote']
               [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [FORCE NOT NULL column [, ...]]
               [ESCAPE [AS] 'escape']
               [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
               [FILL MISSING FIELDS] )]
[ ENCODING 'encoding' ]
[ [LOG ERRORS [INTO error_table]] SEGMENT REJECT LIMIT count
       [ROWS | PERCENT] ]
CREATE WRITABLE EXTERNAL TABLE table_name
( column_name data_type [, ...] | LIKE other_table )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
               [( [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [ESCAPE [AS] 'escape' | 'OFF'] )]
          | 'CSV'
               [([QUOTE [AS] 'quote']
               [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [FORCE QUOTE column [, ...]] ]
               [ESCAPE [AS] 'escape'] )]
[ ENCODING 'encoding' ]
[ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
ossprotocol:
   oss://oss_endpoint prefix=prefix_name
    id=userossid key=userosskey bucket=ossbucket compressiontype=[none
|gzip] async=[true|false]
ossprotocol:
   oss://oss_endpoint dir=[folder/[folder/]...]/file_name
    id=userossid key=userosskey bucket=ossbucket compressiontype=[none
|gzip] async=[true|false]
ossprotocol:
   oss://oss_endpoint filepath=[folder/[folder/]...]/file_name
id=userossid key=userosskey bucket=ossbucket compressiontype=[none|
gzip] async=[true|false]
```

更多信息请参考CREATE EXTERNAL TABLE。

CREATE FUNCTION

定义一个新的函数。

```
CREATE [OR REPLACE] FUNCTION name
```

```
    ( [ [argmode] [argname] argtype [ { DEFAULT | = } defexpr ]
[, ...] ] )
      [ RETURNS { [ SETOF ] rettype
        | TABLE ([{ argname argtype | LIKE other table }
          [, ...]])
        } ]
    { LANGUAGE langname
    | IMMUTABLE | STABLE | VOLATILE
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [EXTERNAL] SECURITY INVOKER | [EXTERNAL] SECURITY DEFINE
    | COST execution_cost
    | SET configuration_parameter { TO value | = value | FROM CURRENT
}
    | AS 'definition'
    | AS 'obj_file', 'link_symbol' } ...
    [ WITH ({ DESCRIBE = describe_function
           } [, ...] ) ]
```

更多信息请参考CREATE FUNCTION。

CREATE GROUP

定义一个新的数据库角色。

```
CREATE GROUP name [ [WITH] option [ ... ] ]
```

更多信息请参考CREATE GROUP。

CREATE INDEX

定义一个新的索引。

```
CREATE [UNIQUE] INDEX name ON table
       [USING btree|bitmap|gist]
       ( {column | (expression)} [opclass] [, ...] )
       [ WITH ( FILLFACTOR = value ) ]
       [TABLESPACE tablespace]
       [WHERE predicate]
```

更多信息请参考CREATE INDEX。

CREATE LIBRARY

定义一个用户自定义软件表。

```
CREATE LIBRARY library_name LANGUAGE [JAVA] FROM oss_location OWNER
ownername
CREATE LIBRARY library_name LANGUAGE [JAVA] VALUES file_content_hex
OWNER ownername
```

更多信息请参考CREATE LIBRARY。

CREATE OPERATOR

定义一个新的操作符。

```
CREATE OPERATOR name (
```

```
        PROCEDURE = funcname
        [, LEFTARG = lefttype] [, RIGHTARG = righttype]
        [, COMMUTATOR = com_op] [, NEGATOR = neg_op]
        [, RESTRICT = res_proc] [, JOIN = join_proc]
        [, HASHES] [, MERGES]
        [, SORT1 = left_sort_op] [, SORT2 = right_sort_op]
        [, LTCMP = less_than_op] [, GTCMP = greater_than_op] )
```

更多信息请参考CREATE OPERATOR。

CREATE RESOURCE QUEUE

定义一个新的资源队列。

```
CREATE RESOURCE QUEUE name WITH (queue_attribute=value [, ... ])
```

更多信息请参考CREATE RESOURCE QUEUE。

CREATE ROLE

定义一个新的数据库角色（用户或组）。

```
CREATE ROLE name [[WITH] option [ ... ]]
```

更多信息请参考CREATE ROLE。

CREATE RULE

定义一个新的重写规则。

```
CREATE [OR REPLACE] RULE name AS ON event
  TO table [WHERE condition]
  DO [ALSO | INSTEAD] { NOTHING | command | (command; command
  ...) }
```

更多信息请参考CREATE RULE。

CREATE SCHEMA

定义一个新的SCHEMA。

```
CREATE SCHEMA schema_name [AUTHORIZATION username]
   [schema_element [ ... ]]

CREATE SCHEMA AUTHORIZATION rolename [schema_element [ ... ]]
```

更多信息请参考CREATE SCHEMA。

CREATE SEQUENCE

定义一个新的序列生成器。

```
CREATE [TEMPORARY | TEMP] SEQUENCE name
       [INCREMENT [BY] value]
       [MINVALUE minvalue | NO MINVALUE]
```

```
        [MAXVALUE maxvalue | NO MAXVALUE]
        [START [ WITH ] start]
        [CACHE cache]
        [[NO] CYCLE]
        [OWNED BY { table.column | NONE }]
```

更多信息请参考CREATE SEQUENCE。

CREATE TABLE

定义一个新的表。

```
CREATE [[GLOBAL | LOCAL] {TEMPORARY | TEMP}] TABLE table_name (
[ { column_name data_type [ DEFAULT default_expr ]
   [column_constraint [ ... ]
[ ENCODING ( storage_directive [,...] ) ]
]
   | table_constraint
   | LIKE other_table [{INCLUDING | EXCLUDING}
                       {DEFAULTS | CONSTRAINTS}] ...}
   [, ... ] ]
   )
   [ INHERITS ( parent_table [, ... ] ) ]
   [ WITH ( storage_parameter=value [, ... ] )
   [ ON COMMIT {PRESERVE ROWS | DELETE ROWS | DROP} ]
   [ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
   [ PARTITION BY partition_type (column)
       [ SUBPARTITION BY partition_type (column) ]
          [ SUBPARTITION TEMPLATE ( template_spec ) ]
       [...]
    ( partition_spec )
        | [ SUBPARTITION BY partition_type (column) ]
          [...]
    ( partition_spec
      [ ( subpartition_spec
            [(...)]
        ) ]
    )
```

更多信息请参考CREATE TABLE。

CREATE TABLE AS

从查询的结果中定义一个新的表。

```
CREATE [ [GLOBAL | LOCAL] {TEMPORARY | TEMP} ] TABLE table_name
   [(column_name [, ...] )]
   [ WITH ( storage_parameter=value [, ... ] ) ]
   [ON COMMIT {PRESERVE ROWS | DELETE ROWS | DROP}]
   [TABLESPACE tablespace]
   AS query
   [DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY]
```

更多信息请参考CREATE TABLE AS。

CREATE TYPE

定义一个新的类型。

```
CREATE TYPE name AS ( attribute_name data_type [, ... ] )

CREATE TYPE name AS ENUM ( 'label' [, ... ] )

CREATE TYPE name (
    INPUT = input_function,
    OUTPUT = output_function
    [, RECEIVE = receive_function]
    [, SEND = send_function]
    [, TYPMOD_IN = type_modifier_input_function ]
    [, TYPMOD_OUT = type_modifier_output_function ]
    [, INTERNALLENGTH = {internallength | VARIABLE}]
    [, PASSEDBYVALUE]
    [, ALIGNMENT = alignment]
    [, STORAGE = storage]
    [, DEFAULT = default]
    [, ELEMENT = element]
    [, DELIMITER = delimiter] )

CREATE TYPE name
```

更多信息请参考CREATE TYPE。

CREATE USER

定义一个默认带有LOGIN权限的数据库角色。

```
CREATE USER name [ [WITH] option [ ... ] ]
```

更多信息请参考CREATE USER。

CREATE VIEW

定义一个新的视图。

```
CREATE [OR REPLACE] [TEMP | TEMPORARY] VIEW name
       [ ( column_name [, ...] ) ]
       AS query
```

更多信息请参考CREATE VIEW。

DEALLOCATE

取消分配一个预编译的语句。

```
DEALLOCATE [PREPARE] name
```

更多信息请参考DEALLOCATE。

DECLARE

定义一个游标。

```
DECLARE name [BINARY] [INSENSITIVE] [NO SCROLL] CURSOR
     [{WITH | WITHOUT} HOLD]
     FOR query [FOR READ ONLY]
```

更多信息请参考DECLARE。

DELETE

从表中删除行。

```
DELETE FROM [ONLY] table [[AS] alias]
      [USING usinglist]
      [WHERE condition | WHERE CURRENT OF cursor_name ]
```

更多信息请参考DELETE。

DROP AGGREGATE

删除聚集函数。

```
DROP AGGREGATE [IF EXISTS] name ( type [, ...] ) [CASCADE | RESTRICT]
```

更多信息请参考DROP AGGREGATE。

DROP CAST

删除一个CAST。

```
DROP CAST [IF EXISTS] (sourcetype AS targettype) [CASCADE | RESTRICT]
```

更多信息请参考DROP CAST。

DROP CONVERSION

删除一个转换。

```
DROP CONVERSION [IF EXISTS] name [CASCADE | RESTRICT]
```

更多信息请参考DROP CONVERSION。

DROP DATABASE

删除一个数据库。

```
DROP DATABASE [IF EXISTS] name
```

更多信息请参考DROP DATABASE。

DROP DOMAIN

删除一个域。

```
DROP DOMAIN [IF EXISTS] name [, ...]  [CASCADE | RESTRICT]
```

更多信息请参考DROP DOMAIN。

DROP EXTENSION

从数据库中删除一个扩展。

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

更多信息请参考DROP EXTENSION。

DROP EXTERNAL TABLE

删除一个外部表定义。

```
DROP EXTERNAL [WEB] TABLE [IF EXISTS] name [CASCADE | RESTRICT]
```

更多信息请参考DROP EXTERNAL TABLE。

DROP FUNCTION

删除一个函数。

```
DROP FUNCTION [IF EXISTS] name ( [ [argmode] [argname] argtype
    [, ...] ] ) [CASCADE | RESTRICT]
```

更多信息请参考DROP FUNCTION。

DROP GROUP

删除一个数据库角色。

```
DROP GROUP [IF EXISTS] name [, ...]
```

更多信息请参考DROP GROUP。

DROP INDEX

删除一个索引。

```
DROP INDEX [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考DROP INDEX。

DROP LIBRARY

删除一个用户定义软件包。

```
DROP LIBRARY library_name
```

更多信息请参考DROP LIBRARY。

DROP OPERATOR

删除一个操作符。

```
DROP OPERATOR [IF EXISTS] name ( {lefttype | NONE} ,
    {righttype | NONE} ) [CASCADE | RESTRICT]
```

更多信息请参考DROP OPERATOR。

DROP OWNED

删除数据库角色所拥有的数据库对象。

```
DROP OWNED BY name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考DROP OWNED。

DROP RESOURCE QUEUE

删除一个资源队列。

```
DROP RESOURCE QUEUE queue_name
```

更多信息请参考DROP RESOURCE QUEUE。

DROP ROLE

删除一个数据库角色。

```
DROP ROLE [IF EXISTS] name [, ...]
```

更多信息请参考DROP ROLE。

DROP RULE

删除一个重写规则。

```
DROP RULE [IF EXISTS] name ON relation [CASCADE | RESTRICT]
```

更多信息请参考DROP RULE。

DROP SCHEMA

删除一个SCHEMA。

```
DROP SCHEMA [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考DROP SCHEMA。

DROP SEQUENCE

删除一个序列。

```
DROP SEQUENCE [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考DROP SEQUENCE。

DROP TABLE

删除一个表。

```
DROP TABLE [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考DROP TABLE。

DROP TYPE

删除一个数据类型。

```
DROP TYPE [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考 DROP TYPE。

DROP USER

删除一个数据库角色。

```
DROP USER [IF EXISTS] name [, ...]
```

更多信息请参考DROP USER。

DROP VIEW

删除一个视图。

```
DROP VIEW [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考DROP VIEW。

END

提交当前事务。

```
END [WORK | TRANSACTION]
```

更多信息请参考END。

EXECUTE

执行一个已经准备好的SQL语句。

```
EXECUTE name [ (parameter [, ...] ) ]
```

更多信息请参考EXECUTE。

EXPLAIN

展示语句的查询计划。

```
EXPLAIN [ANALYZE] [VERBOSE] statement
```

更多信息请参考EXPLAIN。

FETCH

使用游标获取查询结果的行。

```
FETCH [ forward_direction { FROM | IN } ] cursorname
```

更多信息请参考FETCH 。

GRANT

定义一个访问权限。

```
GRANT { {SELECT | INSERT | UPDATE | DELETE | REFERENCES |
TRIGGER | TRUNCATE } [,...] | ALL [PRIVILEGES] }
    ON [TABLE] tablename [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]

GRANT { {USAGE | SELECT | UPDATE} [,...] | ALL [PRIVILEGES] }
    ON SEQUENCE sequencename [, ...]
    TO { rolename | PUBLIC } [, ...] [WITH GRANT OPTION]

GRANT { {CREATE | CONNECT | TEMPORARY | TEMP} [,...] | ALL
[PRIVILEGES] }
    ON DATABASE dbname [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]

GRANT { EXECUTE | ALL [PRIVILEGES] }
    ON FUNCTION funcname ( [ [argmode] [argname] argtype [, ...]
] ) [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]

GRANT { USAGE | ALL [PRIVILEGES] }
```

```
    ON LANGUAGE langname [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]

GRANT { {CREATE | USAGE} [,...] | ALL [PRIVILEGES] }
    ON SCHEMA schemaname [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]

GRANT { CREATE | ALL [PRIVILEGES] }
    ON TABLESPACE tablespacename [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]

GRANT parent_role [, ...]
    TO member_role [, ...] [WITH ADMIN OPTION]

GRANT { SELECT | INSERT | ALL [PRIVILEGES] }
    ON PROTOCOL protocolname
    TO username
```

更多信息请参考GRANT。

INSERT

在表中创建新的行。

```
INSERT INTO table [( column [, ...] )]
   {DEFAULT VALUES | VALUES ( {expression | DEFAULT} [, ...] )
   [, ...] | query}
```

更多信息请参考INSERT。

LOAD

加载或重新加载共享库文件。

```
LOAD 'filename'
```

更多信息请参考 LOAD。

LOCK

锁住一张表。

```
LOCK [TABLE] name [, ...] [IN lockmode MODE] [NOWAIT]
```

更多信息请参考LOCK。

MOVE

放置一个游标。

```
MOVE [ forward_direction {FROM | IN} ] cursorname
```

更多信息请参考MOVE。

PREPARE

准备一个执行的语句。

```
PREPARE name [ (datatype [, ...] ) ] AS statement
```

更多信息请参考PREPARE。

REASSIGN OWNED

改变数据库角色所拥有的数据库对象的所有权。

```
REASSIGN OWNED BY old_role [, ...] TO new_role
```

更多信息请参考REASSIGN OWNED。

REINDEX

重新构建索引。

```
REINDEX {INDEX | TABLE | DATABASE | SYSTEM} name
```

更多信息请参考REINDEX。

RELEASE SAVEPOINT

销毁一个之前定义过的SAVEPOINT。

```
RELEASE [SAVEPOINT] savepoint_name
```

更多信息请参考RELEASE SAVEPOINT。

RESET

恢复系统配置参数的值为默认值。

```
RESET configuration_parameter

RESET ALL
```

更多信息请参考RESET。

REVOKE

撤销访问权限。

```
REVOKE [GRANT OPTION FOR] { {SELECT | INSERT | UPDATE | DELETE
       | REFERENCES | TRIGGER | TRUNCATE } [,...] | ALL [PRIVILEGES] }
       ON [TABLE] tablename [, ...]
       FROM {rolename | PUBLIC} [, ...]
       [CASCADE | RESTRICT]

REVOKE [GRANT OPTION FOR] { {USAGE | SELECT | UPDATE} [,...]
       | ALL [PRIVILEGES] }
```

```
        ON SEQUENCE sequencename [, ...]
        FROM { rolename | PUBLIC } [, ...]
        [CASCADE | RESTRICT]

REVOKE [GRANT OPTION FOR] { {CREATE | CONNECT
        | TEMPORARY | TEMP} [,...] | ALL [PRIVILEGES] }
        ON DATABASE dbname [, ...]
        FROM {rolename | PUBLIC} [, ...]
        [CASCADE | RESTRICT]

REVOKE [GRANT OPTION FOR] {EXECUTE | ALL [PRIVILEGES]}
        ON FUNCTION funcname ( [[argmode] [argname] argtype
                               [, ...]] ) [, ...]
        FROM {rolename | PUBLIC} [, ...]
        [CASCADE | RESTRICT]

REVOKE [GRANT OPTION FOR] {USAGE | ALL [PRIVILEGES]}
        ON LANGUAGE langname [, ...]
        FROM {rolename | PUBLIC} [, ...]
        [ CASCADE | RESTRICT ]

REVOKE [GRANT OPTION FOR] { {CREATE | USAGE} [,...]
        | ALL [PRIVILEGES] }
        ON SCHEMA schemaname [, ...]
        FROM {rolename | PUBLIC} [, ...]
        [CASCADE | RESTRICT]

REVOKE [GRANT OPTION FOR] { CREATE | ALL [PRIVILEGES] }
        ON TABLESPACE tablespacename [, ...]
        FROM { rolename | PUBLIC } [, ...]
        [CASCADE | RESTRICT]

REVOKE [ADMIN OPTION FOR] parent_role [, ...]
        FROM member_role [, ...]
        [CASCADE | RESTRICT]
```

更多信息请参考REVOKE。

ROLLBACK

中止当前事务。

```
ROLLBACK [WORK | TRANSACTION]
```

更多信息请参考ROLLBACK。

ROLLBACK TO SAVEPOINT

将当前事务回滚到某个SAVEPOINT。

```
ROLLBACK [WORK | TRANSACTION] TO [SAVEPOINT] savepoint_name
```

更多信息请参考ROLLBACK TO SAVEPOINT。

SAVEPOINT

在当前事务定义一个新的savepoint。

```
SAVEPOINT savepoint_name
```

更多信息请参考SAVEPOINT。

SELECT

从表或者视图中检索行。

```
[ WITH with_query [, ...] ]
SELECT [ALL | DISTINCT [ON (expression [, ...])]]
  * | expression [[AS] output_name] [, ...]
  [FROM from_item [, ...]]
  [WHERE condition]
  [GROUP BY grouping_element [, ...]]
  [HAVING condition [, ...]]
  [WINDOW window_name AS (window_specification)]
  [{UNION | INTERSECT | EXCEPT} [ALL] select]
  [ORDER BY expression [ASC | DESC | USING operator] [NULLS {FIRST |
LAST}] [, ...]]
  [LIMIT {count | ALL}]
  [OFFSET start]
  [FOR {UPDATE | SHARE} [OF table_name [, ...]] [NOWAIT] [...]]
```

更多信息请参考SELECT。

SELECT INTO

从查询结果中定义一个新的表。

```
[ WITH with_query [, ...] ]
SELECT [ALL | DISTINCT [ON ( expression [, ...] )]]
    * | expression [AS output_name] [, ...]
    INTO [TEMPORARY | TEMP] [TABLE] new_table
    [FROM from_item [, ...]]
    [WHERE condition]
    [GROUP BY expression [, ...]]
    [HAVING condition [, ...]]
    [{UNION | INTERSECT | EXCEPT} [ALL] select]
    [ORDER BY expression [ASC | DESC | USING operator] [NULLS {FIRST
 | LAST}] [, ...]]
    [LIMIT {count | ALL}]
    [OFFSET start]
    [FOR {UPDATE | SHARE} [OF table_name [, ...]] [NOWAIT]
    [...]]
```

更多信息请参考SELECT INTO。

SET

改变数据库配置参数的值。

```
SET [SESSION | LOCAL] configuration_parameter {TO | =} value |
    'value' | DEFAULT}
```

```
SET [SESSION | LOCAL] TIME ZONE {timezone | LOCAL | DEFAULT}
```

更多信息请参考SET。

SET ROLE

设置当前会话当前角色的标识符。

```
SET [SESSION | LOCAL] ROLE rolename

SET [SESSION | LOCAL] ROLE NONE

RESET ROLE
```

更多信息请参考SET ROLE。

SET SESSION AUTHORIZATION

设置会话角色标识符和当前会话当前角色的标识符。

```
SET [SESSION | LOCAL] SESSION AUTHORIZATION rolename

SET [SESSION | LOCAL] SESSION AUTHORIZATION DEFAULT

RESET SESSION AUTHORIZATION
```

更多信息请参考SET SESSION AUTHORIZATION。

SET TRANSACTION

设置当前事务的特征。

```
SET TRANSACTION [transaction_mode] [READ ONLY | READ WRITE]

SET SESSION CHARACTERISTICS AS TRANSACTION transaction_mode
     [READ ONLY | READ WRITE]
```

更多信息请参考SET TRANSACTION。

SHOW

显示当前系统配置参数的值。

```
SHOW configuration_parameter

SHOW ALL
```

更多信息请参考SHOW。

START TRANSACTION

开始一个事务块。

```
START TRANSACTION [SERIALIZABLE | READ COMMITTED | READ UNCOMMITTED]
```

```
                    [READ WRITE | READ ONLY]
```

更多信息请参考START TRANSACTION。

TRUNCATE

清空表的所有行。

```
TRUNCATE [TABLE] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考TRUNCATE。

UDPATE

更新表的行。

```
UPDATE [ONLY] table [[AS] alias]
   SET {column = {expression | DEFAULT} |
   (column [, ...]) = ({expression | DEFAULT} [, ...])} [, ...]
   [FROM fromlist]
   [WHERE condition | WHERE CURRENT OF cursor_name ]
```

更多信息请参考UPDATE。

VACUUM

垃圾收集和选择性分析数据库。

```
VACUUM [FULL] [FREEZE] [VERBOSE] [table]

VACUUM [FULL] [FREEZE] [VERBOSE] ANALYZE
              [table [(column [, ...] )]]
```

更多信息请参考VACUUM。

VALUES

计算一组行。

```
VALUES ( expression [, ...] ) [, ...]
   [ORDER BY sort_expression [ASC | DESC | USING operator] [, ...]]
   [LIMIT {count | ALL}] [OFFSET start]
```

更多信息请参考VALUES。

3.3 用户权限管理

用户管理

实例创建过程中，会提示用户指定初始用户名和密码，这个初始用户为“根用户”。实例创建好后，用户可以使用该根用户连接数据库。使用psql（PostgreSQL 或 Greenplum 的客户端工具）连接数据库后，通过`\du+`命令可以查看所有用户的信息，示例如下：

注意:

除了根用户外，还有其他内部管理用户被创建。

```
postgres=> \du+
                                   List of roles
  Role name   |            Attributes             | Member of |
Description
--------------+-----------------------------------+-----------
+---------------
 root_user    |                                   |           |
rds_superuser
 ...
```

目前，AnalyticDB for PostgreSQL 没有开放 SUPERUSER 权限，对应的是 RDS_SUPERUSER，这一点与云数据库 RDS（PostgreSQL）中的权限体系一致。所以，根用户（如上面的示例中的 root_user）具有 RDS_SUPERUSER 权限，这个权限属性只能通过查看用户的描述（Description）来识别。根用户具有如下权限：

- 具有 CREATEROLE、CREATEDB 和 LOGIN 权限，即可以用来创建数据库和用户，但没有 SUPERUSER 权限。
- 查看和修改其它非超级用户的数据表，执行 SELECT、UPDATE、DELTE 或更改所有者（Owner）等操作。
- 查看其它非超级用户的连接信息、撤销（Cancel）其 SQL 或终止（Kill）其连接。
- 执行 CREATE EXTENSION 和 DROP EXTENSION 命令，创建和删除插件。
- 创建其他具有 RDS_SUPERUSER 权限的用户，示例如下：

```
CRATE ROLE root_user2 RDS_SUPERUSER LOGIN PASSWORD 'xyz';
```

权限管理

用户可以在数据库（Database）、模式（Schema）、表等多个层次管理权限。例如，赋予一个用户表上的读取权限，但收回修改权限，示例如下。

```
GRANT SELECT ON TABLE t1 TO normal_user1;
REVOKE UPDATE ON TABLE t1 FROM normal_user1;
```

```
REVOKE DELETE ON TABLE t1 FROM normal_user1;
```

参考文档

关于具体的用户与权限管理方法，请参见Managing Roles and Privileges。

3.4 插入、更新、删除数据

本文介绍在AnalyticDB for PostgreSQL数据库中，如何插入、更新、删除数据。

插入行（INSERT）

使用INSERT命令在一个表中插入行。命令格式如下：

```
INSERT INTO table [( column [, ...] )]
   {DEFAULT VALUES | VALUES ( {expression | DEFAULT} [, ...] )
   [, ...] | query}
```

更多信息请参考INSERT。

示例：

插入单行数据。

```
INSERT INTO products (name, price, product_no) VALUES ('Cheese', 9.99
, 1);
```

在一条命令中插入多行数据。

```
INSERT INTO products (product_no, name, price) VALUES
    (1, 'Cheese', 9.99),
    (2, 'Bread', 1.99),
    (3, 'Milk', 2.99);
```

使用标量表达式插入数据。

```
INSERT INTO films SELECT * FROM tmp_films WHERE date_prod <
'2016-05-07';
```

说明：

如果要插入大量数据，推荐使用外部表或者COPY命令，比INSERT性能好。

更新行（UPDATE）

使用UPDATE命令在一个表中更新行。命令格式如下：

```
UPDATE [ONLY] table [[AS] alias]
   SET {column = {expression | DEFAULT} |
   (column [, ...]) = ({expression | DEFAULT} [, ...])} [, ...]
   [FROM fromlist]
```

```
    [WHERE condition | WHERE CURRENT OF cursor_name ]
```

更多信息请参考UPDATE。

使用限制：

- 分布键列不能被更新。
- 分区键列不能被更新。
- 不能在UPDATE语句中使用STABLE或VOLATILE函数。
- 不支持RETURNING子句。

示例：

将所有价格为5的产品更新价格为10。

```
UPDATE products SET price = 10 WHERE price = 5;
```

删除行（DELETE）

使用DELETE命令从一个表中删除行。命令格式如下：

```
DELETE FROM [ONLY] table [[AS] alias]
      [USING usinglist]
      [WHERE condition | WHERE CURRENT OF cursor_name ]
```

更多信息请参考DELETE。

使用限制：

- 不能在DELETE语句中使用STABLE或VOLATILE函数。
- 不支持RETURNING子句。

示例：

从产品表中删除所有价格为10的行。

```
DELETE FROM products WHERE price = 10;
```

从表中删除所有行。

```
DELETE FROM products;
```

截断表（TRUNCATE）

使用TRUNCATE命令可以快速地移除一个表中的所有行。命令格式如下：

```
TRUNCATE [TABLE] name [, ...] [CASCADE | RESTRICT]
```

更多信息请参考TRUNCATE。

示例：

清空mytable表中的所有行，TRUNCATE不扫描该表，因此它不会处理继承的子表或者ON DELETE的重写规则，只会截断命令中的表中的行。

```
TRUNCATE mytable;
```

3.5 事务管理

事务管理

AnalyticDB for PostgreSQL提供了下列事务管理相关的SQL命令：

- **BEGIN或者START TRANSACTION 开始一个事务块。**
- **END或者COMMIT提交一个事务的结果。**
- **ROLLBACK放弃一个事务而不做任何更改。**
- **SAVEPOINT在一个事务中标记一个位置并且允许做部分回滚。用户可以回滚在一个保存点之后执行的命令但保留该保存点之前执行的命令。**
- **ROLLBACK TO SAVEPOINT 回滚一个事务到一个保存点。**
- **RELEASE SAVEPOINT 销毁一个事务内的保存点。**

示例：

在事务中建立一个保存点，后来撤销在它建立之后执行的所有命令的效果：

```
BEGIN;
    INSERT INTO table1 VALUES (1);
    SAVEPOINT my_savepoint;
    INSERT INTO table1 VALUES (2);
    ROLLBACK TO SAVEPOINT my_savepoint;
    INSERT INTO table1 VALUES (3);
COMMIT;
```

上面的事务将插入值 1 和 3，但不会插入 2。

要建立并且稍后销毁一个保存点：

```
BEGIN;
    INSERT INTO table1 VALUES (3);
    SAVEPOINT my_savepoint;
    INSERT INTO table1 VALUES (4);
    RELEASE SAVEPOINT my_savepoint;
COMMIT;
```

上面的事务将插入值 3 和 4。

事务隔离级别

AnalyticDB for PostgreSQL提供以下两种SQL事务级别，默认为读已提交（READ COMMITTED）。

- - 读已提交（READ COMMITTED）：SQL标准中的读未提交（READ UNCOMMITED）和读已提交（READ COMMITTED）使用读已提交（READ COMMITTED）级别。
  - 可序列化（SERIALIZABLE）：SQL标准中的可重复读（REPEATABLE READ）和可序列化（SERIALIZABLE）使用可序列化（SERIALIZABLE）级别。

示例：

使用可序列化（SERIALIZABLE）隔离级别开始事务块：

```
BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

3.6 JSON 数据类型操作

JSON 类型几乎已成为互联网及物联网（IoT）的基础数据类型，具体协议请参见 JSON 官网。AnalyticDB for PostgreSQL数据库对JSON数据类型做了完善的支持。这部分介绍对JSON数据类型的操作，包括：

- JSON输入输出语法
- JSON操作符
- JSON创建函数
- JSON处理函数

JSON输入输出语法

AnalyticDB for PostgreSQL支持的JSON数据类型的输入和输出语法详见RFC 7159。

一个JSON数值可以是一个简单值（数字、字符串、true/null/false），数组，对象。下列都是合法的json表达式：

```
-- 简单标量/简单值
-- 简单值可以是数字、带引号的字符串、true、false或者null
SELECT '5'::json;

-- 零个或者更多个元素的数组（元素类型可以不同）
SELECT '[1, 2, "foo", null]'::json;

-- 含有键/值对的对象
-- 注意对象的键必须总是带引号的字符串
SELECT '{"bar": "baz", "balance": 7.77, "active": false}'::json;

-- 数组和对象可以任意嵌套
```

```
SELECT '{"foo": [true, "bar"], "tags": {"a": 1, "b": null}}'::json;
```

JSON操作符

下表说明了可以用于JSON数据类型的操作符。

操作符	右操作数类型	描述	例子	结果
->	int	获得JSON数组元素（索引从零开始）。	`'[{"a":"foo"}, {"b":"bar"}, {"c":"baz"}]'::json->2`	`{"c":"baz"}`
->	text	根据键获得JSON对象的域。	`'{"a": {"b":"foo"}}'::json->'a'`	`{"b":"foo"}`
->>	int	获得JSON数组元素的文本形式。	`'[1,2,3]'::json->>2`	`3`
->>	text	获得JSON对象域的文本形式。	`'{"a":1,"b":2}'::json->>'b'`	`2`
#>	text[]	获得在指定路径上的JSON对象。	`'{"a": {"b":{"c": "foo"}}}'::json#>'{a,b}'`	`{"c": "foo"}`
#>>	text[]	获得在指定路径上的JSON对象的文本形式。	`'{"a":[1,2,3], "b":[4,5,6]}'::json#>>'{a,2}'`	`3`

JSON创建函数

下表说明了用于创建JSON值的函数。

函数	描述	例子	结果
to_json (anyelement)	返回该值作为一个合法的JSON对象。数组和组合会被递归处理并且转换成数组和对象。如果输入包含一个从该类型到JSON的造型，会使用该cast函数来执行转换，否则将会产生一个JSON标量值。对于任何非数字、布尔值或空值的标量类型，会使用其文本表示，并且加上适当的引号和转义让它变成一个合法的JSON字符串。	to_json ('Fred said "Hi."'::text)	"Fred said \"Hi.\""
array_to_json (anyarray [, pretty_bool])	返回该数组为一个JSON数组。一个多维数组会变成一个JSON数组的数组。 说明: 如果pretty_bool为true，在第一维元素之间会增加换行。	array_to_json ('{{1,5},{99,100}}'::int[])	[[1,5],[99,100]]
row_to_json (record [, pretty_bool])	返回该行为一个JSON对象。 说明: 如果pretty_bool为true，在第一级别元素之间会增加换行。	row_to_json (row(1,'foo'))	{"f1":1,"f2":"foo"}

JSON处理函数

下表说明了处理JSON值的函数。

函数	返回类型	描述	例子	例子结果
`json_each(json)`	`setof key text, value json setof key text, value jsonb`	把最外层的JSON对象展开成键/值对的集合。	`select * from json_each('{"a":"foo", "b":"bar"}')`	`key | value -----+------- a | "foo" b | "bar"`
`json_each_text(json)`	`setof key text, value text`	把最外层的JSON对象展开成键/值对的集合。返回值的类型是text。	`select * from json_each_text('{"a":"foo", "b":"bar"}')`	`key | value -----+------- a | foo b | bar`
`json_extract_path(from_json json, VARIADIC path_elems text[])`	`json`	返回path_elems指定的JSON值。等效于#>操作符。	`json_extract_path('{"f2":{"f3":1},"f4":{"f5":99,"f6":"foo"}}','f4')`	`{"f5":99,"f6":"foo"}`
`json_extract_path_text(from_json json, VARIADIC path_elems text[])`	`text`	返回path_elems指定的JSON值为文本。等效于#>>操作符。	`json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"foo"}}','f4','f6')`	`foo`
`json_object_keys(json)`	`setof text`	返回最外层JSON对象中的键集合。	`json_object_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}}')`	`json_object_keys ------------------ f1 f2`
`json_populate_record(base anyelement, from_json json)`	`anyelement`	把Expands the object in from_json中的对象展开成一行，其中的列匹配由base定义的记录类型。	`select * from json_populate_record(null::myrowtype, '{"a":1,"b":2}')`	`a | b ---+--- 1 | 2`
`json_popul`	`setof`	将from_json中	`select * from`	

```
=> insert into tj(ary, obj, num) values('{1,5}'::int[], '{"obj":1}', 5
);
INSERT 0 1
=> select row_to_json(q) from (select id, ary, obj, num from tj) as q;
                row_to_json
-------------------------------------------
 {"f1":1,"f2":[1,5],"f3":{"obj":1},"f4":5}
(1 row)
=> insert into tj(ary, obj, num) values('{2,5}'::int[], '{"obj":2}', 5
);
INSERT 0 1
=> select row_to_json(q) from (select id, ary, obj, num from tj) as q;
                row_to_json
-------------------------------------------
 {"f1":1,"f2":[1,5],"f3":{"obj":1},"f4":5}
 {"f1":2,"f2":[2,5],"f3":{"obj":2},"f4":5}
(2 rows)
```

说明:

JSON 类型不能支持作为分布键来使用；也不支持 JSON 聚合函数。

多表 JOIN

```
create table tj2(id serial, ary int[], obj json, num integer);
=> insert into tj2(ary, obj, num) values('{2,5}'::int[], '{"obj":2}',
5);
INSERT 0 1
=> select * from tj, tj2 where tj.obj->>'obj' = tj2.obj->>'obj';
 id |  ary  |    obj    | num | id |  ary  |    obj    | num
----+-------+-----------+-----+----+-------+-----------+-----
  2 | {2,5} | {"obj":2} |   5 |  1 | {2,5} | {"obj":2} |   5
(1 row)
=> select * from tj, tj2 where json_object_field_text(tj.obj, 'obj')
 = json_object_field_text(tj2.obj, 'obj');
 id |  ary  |    obj    | num | id |  ary  |    obj    | num
----+-------+-----------+-----+----+-------+-----------+-----
  2 | {2,5} | {"obj":2} |   5 |  1 | {2,5} | {"obj":2} |   5
(1 row)
```

JSON 函数索引

```
CREATE TEMP TABLE test_json (
       json_type text,
       obj json
);
=> insert into test_json values('aa', '{"f2":{"f3":1},"f4":{"f5":99,"
f6":"foo"}}');
INSERT 0 1
=> insert into test_json values('cc', '{"f7":{"f3":1},"f8":{"f5":99,"
f6":"foo"}}');
INSERT 0 1
=> select obj->'f2' from test_json where json_type = 'aa';
 ?column?
----------
 {"f3":1}
(1 row)
=> create index i on test_json (json_extract_path_text(obj, '{f4}'));
CREATE INDEX
=> select * from test_json where json_extract_path_text(obj, '{f4}') =
 '{"f5":99,"f6":"foo"}';
```

```
 json_type |                  obj
-----------+-----------------------------------------
 aa        | {"f2":{"f3":1},"f4":{"f5":99,"f6":"foo"}}
(1 row)
```

下面是 Python 访问的一个例子：

```
#! /bin/env python
import time
import json
import psycopg2
def gpquery(sql):
    conn = None
    try:
        conn = psycopg2.connect("dbname=sanity1x2")
        conn.autocommit = True
        cur = conn.cursor()
        cur.execute(sql)
        return cur.fetchall()
    except Exception as e:
        if conn:
            try:
                conn.close()
            except:
                pass
            time.sleep(10)
        print e
    return None
def main():
    sql = "select obj from tj;"
    #rows = Connection(host, port, user, pwd, dbname).query(sql)
    rows = gpquery(sql)
    for row in rows:
        print json.loads(row[0])
if __name__ == "__main__":
    main()
```

3.7 列存表排序键（SortKey）定义

排序键（SortKey）是列存表的一种可选属性，可以将数据按照排序键顺序存储在磁盘文件中。使用排序键的优势主要有：

- 加速列存优化，收集的min、max元信息很少有重叠，过滤性好。
- 避免对含有order by和group by等需要排序的SQL数据再次排序，从磁盘中直接读取出来的数据就是满足条件的有序数据。

本文介绍了排序键在不同场景下的用法。

创建表时定义排序键

使用命令`CREATE TABLE`定义一个新的表。语法如下：

```
CREATE [[GLOBAL | LOCAL] {TEMPORARY | TEMP}] TABLE table_name (
[ { column_name data_type [ DEFAULT default_expr ]     [column_con
straint [ ... ]
```

```
[ ENCODING ( storage_directive [,...] ) ]
]
   | table_constraint
   | LIKE other_table [{INCLUDING | EXCLUDING}
                      {DEFAULTS | CONSTRAINTS}] ...}
   [, ... ] ]
   [column_reference_storage_directive [, ] ]
   )
   [ INHERITS ( parent_table [, ... ] ) ]
   [ WITH ( storage_parameter=value [, ... ] )
   [ ON COMMIT {PRESERVE ROWS | DELETE ROWS | DROP} ]
   [ TABLESPACE tablespace ]
   [ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
   [ SORTKEY (column, [ ... ] )]
   [ PARTITION BY partition_type (column)
       [ SUBPARTITION BY partition_type (column) ]
          [ SUBPARTITION TEMPLATE ( template_spec ) ]
       [...]
    ( partition_spec )
        | [ SUBPARTITION BY partition_type (column) ]
          [...]
    ( partition_spec
      [ ( subpartition_spec
           [(...)]
       ) ]
    )
```

样例

```
create table test(date text, time text, open float, high float, low
float, volume int) with(APPENDONLY=true,ORIENTATION=column) sortkey (
volume);
```

对表进行排序

命令如下：

```
VACUUM SORT ONLY [tablename]
```

修改排序键

命令如下：

```
ALTER [[GLOBAL | LOCAL] {TEMPORARY | TEMP}] TABLE table_name SET
SORTKEY (column, [ ... ] )
```

注意：

这个命令只会修改catalog，不会对数据立即排序，需要使用`VACUUM SORT ONLY`命令排序。

样例

```
alter table test set sortkey (high,low);
```

注意事项

一旦对表进行了更新（例如Insert、Update、Delete），表的数据将被视为未按排序键排序，查询中不会将直接从磁盘读出的数据视为有序数据。此时，需要重新执行`VACUUM SORT ONLY`命令重新整理表数据。

4 数据库的维护

4.1 空间回收 VACUUM

背景信息

表中的数据被删除或更新后（UPDATE/DELTE），物理存储层面并不会直接删除数据，而是标记这些数据不可见，所以会在数据页中留下很多“空洞”，在读取数据时，这些“空洞”会随数据页一起加载，拖慢数据扫描速度，需要定期回收删除的空间。

空间回收方法

使用VACUUM命令，可以对表进行重新整理，回收空间，以便获取更好的数据读取性能。VACUUM命令如下：

```
VACUUM [FULL] [FREEZE] [VERBOSE] [table];
```

VACUUM 会在页内进行整理，VACUUM FULL会跨数据页移动数据。 VACUUM执行速度更快， VACUUM FULL执行地更彻底，但会请求排他锁。建议定期对系统表进行VACUUM（每周一次）。

使用建议

什么情况下做VACUUM?

- 不锁表回收空间，只能回收部分空间。
- 频率：对于有较多实时更新的表，每天做一次。
- 如果更新是每天一次批量进行的，可以在每天批量更新后做一次。
- 对系统影响：不会锁表，表可以正常读写。会导致CPU、I/O使用率增加，可能影响查询的性能。

什么情况下做VACUUM FULL?

- 锁表，通过重建表回收空间,可回收所有空洞空间。对做了大量更新后的表，建议尽快执行VACUUM FULL。
- 频率：至少每周执行一次。如果每天会更新几乎所有数据，需要每天做一次。
- 对系统影响：会对正在进行vacuum full的表锁定，无法读写。会导致CPU、I/O使用率增加。建议在维护窗口进行操作。

查询需要执行VACUUM的表

AnalyticDB for PostgreSQL提供了一个gp_bloat_diag视图，统计当前页数和实际需要页数的比例。通过analyze table来收集统计信息之后，查看该视图。

```
gpadmin=# SELECT * FROM gp_toolkit.gp_bloat_diag;
bdirelid | bdinspname | bdirelname | bdirelpages | bdiexppages |
          bdidiag
----------+------------+------------+-------------+-------------
+---------------------------------------
    21488 | public     | t1         |          97 |           1 |
significant amount of bloat suspected
(1 row)
```

其结果只包括发生了中度或者显著膨胀的表。当实际页面数和预期页面的比率超过4但小于10时，就会报告为中度膨胀。当该比率超过10时就会报告显著膨胀。对于这些表，可以考虑进行VACUUM FULL来回收空间。

VACUUM FREEZE的使用

AnalyticDB for PostgreSQL执行的所有事务都有唯一的事务ID(XID)，XID是单调递增的，上限是20亿。

随着数据库执行事务的增多，为防止XID超过极限，在XID接近xid_stop_limit-xid_warn_limit(默认500000000)时， ADB for PG会对执行事务的sql返回warning信息，提醒用户：

```
WARNING: database "database_name" must be vacuumed within number_of_
transactions transactions
```

用户可通过手动执行VACUUM FREEZE当前database来缩小XID。

如果忽略这个warning信息，在XID继续增长到超过xid_stop_limit(默认1000000000)时，ADB for PG会拒绝新的事务执行，并返回报错信息：

```
FATAL: database is not accepting commands to avoid wraparound data
loss in database "database_name"
```

此时，您需要通过提交工单联系阿里云工程师来解决此问题。

5 数据写入

5.1 数据写入方式概述

AnalyticDB for PostgreSQL 提供三种数据写入/加载方式，分别是采用 INSERT 语句，COPY 命令及OSS外表的并行写入。其中 INSERT 语句和 COPY 命令的写入数据均通过 Master 节点，INSERT 语句每秒支持最高 3MB数据写入量， COPY 命令的写入性能快于 INSERT，最高可以达到每秒 30MB。OSS 外表为各个数据分区(Segment)并行读取，单分区从 OSS 加载的能力即可达到 30MB，整体加载速率随分区数线性扩展，为性能最高的数据写入加载方式。

- 方式一：INSERT 语句

  通过 INSERT 语句写入数据，为提升写入速度，建议单条 INSERT 语句拼装多个值 VALUE 执行，最高可实现每秒 3MB 数据量写入。

  ```
  INSERT INTO performers (name, specialty) VALUES (‘Sinatra’, ‘Singer
  ’), …;
  ```

  注意:

  执行高吞吐的写入时，建议关闭 ORCA SQL 优化器。ORCA 适合复杂查询优化，但对简单语句，会带来不必要的解析和优化CPU计算代价。Session 会话级别关闭ORCA优化器命令：“set optimizer = off;”；或提工单在实例级别关闭。

- 方式二：COPY命令

  使用\COPY命令，可以将本地的文本文件数据导入 AnalyticDB for PostgreSQL，本地的文本文件要求是格式化的，如通过逗号、分号或特有符号作为分割符号的文件。同时也支持用户使用 JDBC 执行 COPY 语句，JDBC 中封装了 CopyIn 方法。采用 COPY 方式，最高支持每秒 30MB 的数据写入或导入速度。

  COPY命令的操作细节可以参见#unique_68

- 方式三：OSS 外表导入和导出

  通过 OSS 外表，数据可以从各个节点（Segment ）并行的写入或导出。整体写入或导出速度，随节点数线性扩展，单节点每秒最高支持 30MB 的数据写入。

  建立OSS外表及数据写入和导出操作，可以参见OSS 外部表的使用

5.2 COPY 命令的使用

您可以直接使用\COPY命令，将本地的文本文件数据导入 AnalyticDB for PostgreSQL数据库实例，或者将数据从AnalyticDB for PostgreSQL中导出到本地文件。这里本地的文本文件要求是格式化的，如通过逗号、分号或特有符号作为分割符号的文件。

注意:

- 由于\COPY命令需要通过 Master 节点进行串行数据写入处理，因此无法实现并行写入大批量数据。如果要进行大量数据的并行写入，请使用基于 OSS 的数据导入方式。
- \COPY命令是 psql 的操作指令，如果您使用的不是\COPY，而是数据库指令COPY，则需要注意只支持 STDIN，不支持 file，因为“根用户”并没有 superuser 权限，不可以进行 file 文件操作。

\COPY导入数据的操作命令参考如下:

```
\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导出数据的操作命令参考如下:

```
\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 ]
```

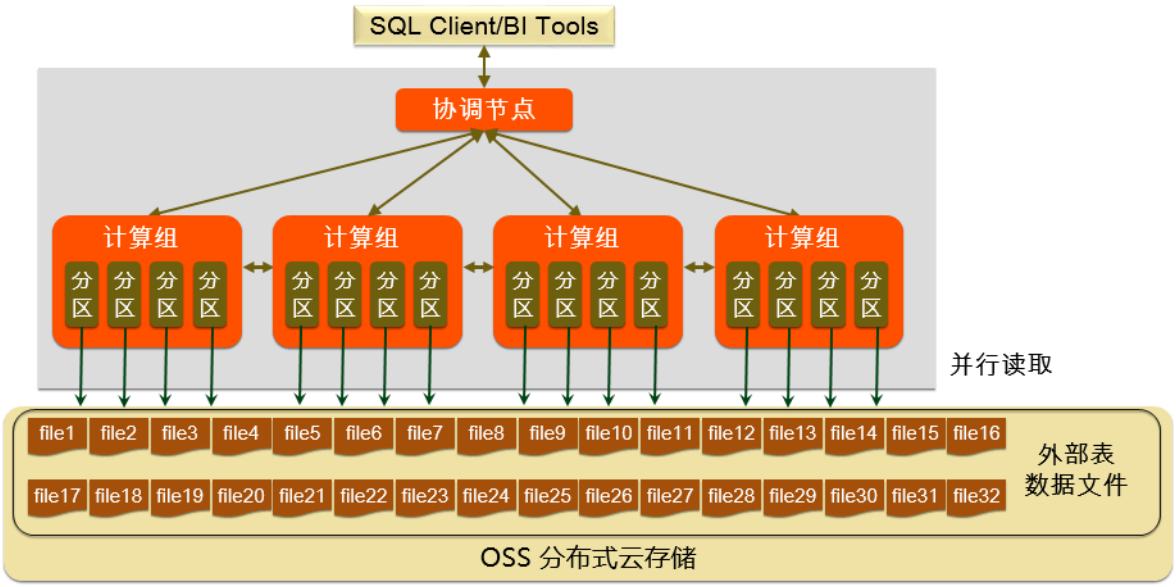
注意:

- 云数据库 AnalyticDB for PostgreSQL 还支持用户使用 JDBC 执行 COPY 语句，JDBC 中封装了 CopyIn 方法，详细用法请参见文档Interface CopyIn。
- COPY 命令使用方法请参见文档COPY。

5.3 OSS 外部表的使用

云数据库 AnalyticDB for PostgreSQL 支持通过 OSS 外部表（即 gpossext 功能），将数据并行从 OSS 导入或导出到 OSS，并支持通过 gzip 进行 OSS 外部表文件压缩，大量节省存储空间及成本。

目前的 gpossext 支持读写text/csv格式的文件或者gzip 压缩格式的 text/csv 文件。



本文内容包括:

- 操作说明
- 参数释义
- 使用示例
- 注意事项
- TEXT/CSV格式说明
- SDK错误处理
- 常见问题
- 参考文档

操作说明

通过 AnalyticDB for PostgreSQL 使用 OSS 外部表，主要涉及以下操作。

- 创建OSS外部表插件（oss_ext）
- 并行导入数据
- 并行导出数据
- 创建 OSS 外部表语法

创建 OSS 外部表插件（oss_ext）

使用 OSS 外部表时，需要在 AnalyticDB for PostgreSQL 中先创建 OSS 外部表插件（每个数据库需要单独创建）。

- 创建命令为：`CREATE EXTENSION IF NOT EXISTS oss_ext;`
- 删除命令为：`DROP EXTENSION IF EXISTS oss_ext;`

并行导入数据

导入数据时，请执行如下步骤：

1. 将数据均匀分散存储在多个 OSS 文件中。

   注意：

   AnalyticDB for PostgreSQL的每个数据分区（segment）将按轮询方式并行对OSS上的数据文件进行读取，文件的数目建议为 数据节点数（Segment 个数）的整数倍，从而提升读取效率。

2. 在 AnalyticDB for PostgreSQL 中，创建 READABLE 外部表。
3. 执行如下操作，并行导入数据。

```
INSERT INTO <目标表> SELECT * FROM <外部表>
```

并行导出数据

导出数据时，请执行如下步骤：

1. 在 AnalyticDB for PostgreSQL 中，创建 WRITABLE 外部表。
2. 执行如下操作，并行把数据导出到 OSS 中。

```
INSERT INTO <外部表> SELECT * FROM <源表>
```

创建 OSS 外部表语法

创建 OSS 外部表语法，请执行如下命令：

```
CREATE [READABLE] EXTERNAL TABLE tablename
( columnname datatype [, ...] | LIKE othertable )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
            [( [HEADER]
               [DELIMITER [AS] 'delimiter' | 'OFF']
               [NULL [AS] 'null string']
               [ESCAPE [AS] 'escape' | 'OFF']
               [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
               [FILL MISSING FIELDS] )]
          | 'CSV'
            [( [HEADER]
               [QUOTE [AS] 'quote']
               [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [FORCE NOT NULL column [, ...]]
               [ESCAPE [AS] 'escape']
               [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
               [FILL MISSING FIELDS] )]
[ ENCODING 'encoding' ]
[ [LOG ERRORS [INTO error_table]] SEGMENT REJECT LIMIT count
       [ROWS | PERCENT] ]
CREATE WRITABLE EXTERNAL TABLE table_name
( column_name data_type [, ...] | LIKE other_table )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
               [( [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [ESCAPE [AS] 'escape' | 'OFF'] )]
          | 'CSV'
               [([QUOTE [AS] 'quote']
               [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [FORCE QUOTE column [, ...]] ]
               [ESCAPE [AS] 'escape'] )]
[ ENCODING 'encoding' ]
[ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
ossprotocol:
   oss://oss_endpoint prefix=prefix_name
    id=userossid key=userosskey bucket=ossbucket compressiontype=[none
|gzip] async=[true|false]
ossprotocol:
   oss://oss_endpoint dir=[folder/[folder/]...]/file_name
    id=userossid key=userosskey bucket=ossbucket compressiontype=[none
|gzip] async=[true|false]
ossprotocol:
   oss://oss_endpoint filepath=[folder/[folder/]...]/file_name
    id=userossid key=userosskey bucket=ossbucket compressiontype=[none
|gzip] async=[true|false]
```

参数释义

该部分介绍各操作中用到的参数定义，涉及到参数包括：

- **常用参数**
- **导入模式参数**
- **导出模式参数**

- 其他通用参数

常用参数

- 协议和 endpoint：格式为“协议名://oss_endpoint”，其中协议名为 oss，oss_endpoint 为 OSS 对应区域的域名。

  注意:

  如果是从阿里云的主机访问数据库，应该使用内网域名（即带有“internal”的域名），避免产生公网流量。

- id：OSS 账号的 ID。
- key：OSS 账号的 key。
- bucket：指定数据文件所在的 bucket，需要通过 OSS 预先创建。
- prefix：指定数据文件对应路径名的前缀，不支持正则表达式，仅是匹配前缀，且与 filepath、dir 互斥，三者只能设置其中一个。
  - 如果创建的是用于数据导入的 READABLE 外部表，则在导入时含有这一前缀的所有 OSS 文件都会被导入。
    - 如果指定 prefix=test/filename，以下文件都会被导入：
      - test/filename
      - test/filenamexxx
      - test/filename/aa
      - test/filenameyyy/aa
      - test/filenameyyy/bb/aa
    - 如果指定 prefix=test/filename/，只有以下文件会被导入（上面列的其他文件不会被导入）：
      - test/filename/aa
  - 如果创建的是用于数据导出的 WRITABLE 外部表，在导出数据时，将根据该前缀自动生成一个唯一的文件名来给导出文件命名。

    注意:

    导出文件将不止有一个，每个数据节点都会导出一个或多个文件。导出文件名格式为 `prefix_tablename_uuid.x`，其中 uuid 是生成的 int64 整型值（精度为微秒的时间戳），x 为节点 ID。支持使用同一外部表多次导出，每次导出的文件将通过 uuid 区分，而同一次导出的文件 uuid 相同。

- dir：OSS 中的虚拟文件夹路径，与 prefix、filepath 互斥，三者只能设置其中一个。
  - 文件夹路径需要以“/”结尾，如`test/mydir/`。
  - 在导入数据时，使用此参数创建外部表，会导入指定虚拟目录下的所有文件，但不包括它子目录和子目录下的文件。与 filepath 不同，dir 下的文件没有命名要求。
  - 在导出数据时，使用此参数创建外部表，所有数据会导出到此目录下的多个文件中，输出文件名的形式为`filename.x`，x 为数字，但可能不是连续的。
- filepath：OSS 中包含路径的文件名称，与 prefix、dir 互斥，三者只能设置其中一个，并且这个参数只能在创建 READABLE 外部表时指定（即只支持在导入数据时使用）。
  - 该文件名称包含该路径，但不包含 bucket 名。
  - 在导入数据时，文件命名方式必须为 `filename` 或 `filename.x`，x 要求从 1 开始，且是连续的。例如，如果指定 filepath = filename，而 OSS 中含有如下文件：

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

    则将被导入的文件有 filename、filename.1 和 filename.2。而因为 filename.3 不存在，所以 filename.4 不会被导入。

导入模式参数

- async：是否启用异步模式导入数据。
  - 开启辅助线程从 OSS 导入数据，加速导入性能。
  - 默认情况下异步模式是打开的，如果需要关掉，可以使用参数 async = false 或 async = f。
  - 异步模式和普通模式比，会消耗更多的硬件资源。
- compressiontype：导入的文件的压缩格式。
  - 指定为 none（缺省值），说明导入的文件没经过压缩。
  - 指定为 gzip，则导入的格式为 gzip。目前仅支持 gzip 压缩格式。
- compressionlevel：设置写入 OSS 的文件的压缩等级，取值范围为 1 - 9，默认值为 6

导出模式参数

- oss_flush_block_size：单次刷出数据到 OSS 的 buffer 大小，默认为 32 MB，可选范围是 1 到 128 MB。
- oss_file_max_size：设置写入到 OSS 的最大文件大小，超出之后会切换到另一个文件继续写。默认为 1024 MB，可选范围是 8 MB 到 4000 MB。

- num_parallel_worker： 设置写入 OSS 的压缩数据的并行压缩线程个数，取值范围为 1 - 8，默认值为3。
- compressiontype：导出文件的压缩格式。
  - 指定为 none（缺省值），说明导出的文件没经过压缩。
  - 指定为 gzip，则导出的格式为 gzip。目前仅支持 gzip 压缩格式。

另外，针对导出模式，有如下注意事项：

- WRITABLE 是导出模式外部表的关键字，创建外部表时需要明确指明。
- 导出模式目前只支持 prefix 和 dir 参数模式，不支持 filepath。
- 导出模式的 DISTRIBUTED BY 子句可以使数据节点（Segment）按指定的分布键将数据写入 OSS。

其他通用参数

针对导入模式和导出模式，还有下列容错相关的参数：

- oss_connect_timeout：设置链接超时，单位为秒，默认是 10 秒。
- oss_dns_cache_timeout：设置 DNS 超时，单位为秒，默认是 60 秒。
- oss_speed_limit：设置能容忍的最小速率，默认是 1024，即 1 KB。
- oss_speed_time：设置能容忍的最长时间，默认是 15 秒。

上述参数如果使用默认值，则如果连续 15 秒的传输速率小于 1 KB，就会触发超时。详细描述请参见 OSS SDK 错误处理。

其他参数兼容 Greenplum EXTERNAL TABLE 的原有语法，具体语法解释请参见 Greenplum 外部表语法官方文档。这部分参数主要有：

- FORMAT：支持文件格式，支持 text、csv 等。
- ENCODING：文件中数据的编码格式，如 utf8。
- LOG ERRORS：指定该子句可以忽略掉导入中出错的数据，将这些数据写入error_table，并可以使用count参数指定报错的阈值。

使用示例

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

```
        (date text, time text, open float, high float,
        low float, volume int)
        location('oss://oss-cn-hangzhou.aliyuncs.com
        dir=osstest/ id=XXX
        key=XXX bucket=testbucket')
        FORMAT 'csv'
        LOG ERRORS SEGMENT REJECT LIMIT 5;
create readable external table ossexample
        (date text, time text, open float, high float,
        low float, volume int)
        location('oss://oss-cn-hangzhou.aliyuncs.com
        filepath=osstest/example.csv id=XXX
        key=XXX bucket=testbucket')
        FORMAT 'csv'
        LOG ERRORS SEGMENT REJECT LIMIT 5;
# 创建 OSS 导出外部表
create WRITABLE external table ossexample_exp
        (date text, time text, open float, high float,
        low float, volume int)
        location('oss://oss-cn-hangzhou.aliyuncs.com
        prefix=osstest/exp/outfromhdb id=XXX
        key=XXX bucket=testbucket') FORMAT 'csv'
        DISTRIBUTED BY (date);
create WRITABLE external table ossexample_exp
        (date text, time text, open float, high float,
        low float, volume int)
        location('oss://oss-cn-hangzhou.aliyuncs.com
        dir=osstest/exp/ id=XXX
        key=XXX bucket=testbucket') FORMAT 'csv'
        DISTRIBUTED BY (date);
# 创建堆表，数据就装载到这张表中
create table example
        (date text, time text, open float,
         high float, low float, volume int)
         DISTRIBUTED BY (date);
# 数据并行地从 ossexample 装载到 example 中
insert into example select * from ossexample;
# 数据并行地从 example 导出到 oss
insert into ossexample_exp select * from example;
# 从下面的执行计划中可以看出，每个 Segment 都会参与工作。
# 每个 Segment 从 OSS 并行拉取数据，然后通过 Redistribution Motion 这个执
行节点将拿到的数据 HASH 计算后分发给对应的 Segment，接受数据的 Segment 通过
Insert 执行节点进行入库。
explain insert into example select * from ossexample;
                                          QUERY PLAN
------------------------------------------------------------------------------
 Insert (slice0; segments: 4)  (rows=250000 width=92)
   ->  Redistribute Motion 4:4  (slice1; segments: 4)  (cost=0.00..
11000.00 rows=250000 width=92)
         Hash Key: ossexample.date
         ->  External Scan on ossexample  (cost=0.00..11000.00 rows=
250000 width=92)
(4 rows)
# 从下面的查询计划可以看到，Segment 把本地数据直接导出到 OSS ，没有进行数据重分布
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)
```

注意事项

- 创建和使用外部表的语法，除了 location 相关的参数，其余部分和 Greenplum 相同。
- 数据导入的性能和 AnalyticDB for PostgreSQL 集群的资源（CPU、IO、内存、网络等）相关，也和 OSS 相关。为了获取最大的导入性能，建议在创建表时，使用列式存储 + 压缩功能。例如，指定子句“WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib, COMPRESSLEVEL=5, BLOCKSIZE=1048576)”，详情请参见 Greenplum Database 表创建语法官方文档。
- 为了保证数据导入的性能，ossendpoint Region 需要匹配 AnalyticDB for PostgreSQL 云上所在 Region，建议 OSS AnalyticDB for PostgreSQL 在同一个 Region 内以获得最好的性能。相关信息请参见 OSS endpoint 信息。

TEXT/CSV 格式说明

下列几个参数可以在外表 DDL 参数中指定，用于规定读写 OSS 的文件格式：

- TEXT/CSV 行分割符号是 ‘\n’ ，也就是换行符。
- DELIMITER 用于定义列的分割符：
  - 当用户数据中包括 DELIMITER 时，则需要和 QUOTE 参数一同使用。
  - 推荐的列分割符有 ‘,’ 、 ‘\t‘ 、 ‘|’ 或一些不常出现的字符。
- QUOTE 以列为单位包裹有特殊字符的用户数据。
  - 用户包含有特殊字符的字符串会被 QUOTE 包裹，用于区分用户数据和控制字符。
  - 如果不必要，例如整数，基于优化效率的考虑，不必使用 QUOTE 包裹数据。
  - QUOTE 不能和 DELIMITER 相同，默认 QUOTE 是双引号。
  - 当用户数据中包含了 QUOTE 字符，则需要使用转义字符 ESCAPE 加以区分。
- ESCAPE 特殊字符转义
  - 转义字符出现在需要转义的特殊字符前，表示它不是一个特殊字符。
  - ESCAPE 默认和 QUOTE 相同，也就是双引号。
  - 也支持设置成 ‘\’(MySQL 默认的转义字符)或别的字符。

典型的 TEXT/CSV 默认控制字符

控制字符 \ 格式	TEXT	CSV
DELIMITER（列分割符）	\t （tab）	，（comma）
QUOTE（摘引）	" （double-quote）	"（double-quote）
ESCAPE（转义）	（不适用）	和 QUOTE 相同

控制字符 \ 格式	TEXT	CSV
NULL（空值）	\N （backslash-N）	（无引号的空字符串）

说明:

所有的控制字符都必须是单字节字符。

SDK 错误处理

当导入或导出操作出错时，错误日志可能会出现如下信息：

- code：出错请求的 HTTP 状态码。
- error_code：OSS 的错误码。
- error_msg：OSS 的错误信息。
- req_id：标识该次请求的 UUID。当您无法解决问题时，可以凭 req_id 来请求 OSS 开发工程师的帮助。

详情请参见OSS API 错误响应，超时相关的错误可以使用 oss_ext 相关参数处理。

常见问题

如果导入过慢，请参见上面“注意事项”中关于导入性能的描述。

参考文档

- OSS endpoint 信息
- OSS help 页面
- OSS SDK 错误处理
- OSS API 错误响应
- Greenplum Database 外部表语法官方文档
- Greenplum Database 表创建语法官方文档

6 数据迁移及同步

6.1 数据迁移及同步方案综述

AnalyticDB for PostgreSQL 提供了多种数据迁移方案，可满足不同的数据同步或迁移的业务需求，使您可以在不影响业务的情况下，平滑地与各种数据库类型实例之间进行迁移或数据同步，包括 RDS MySQL， RDS PostgreSQL、RDS PPAS、MaxCompute、Greenplum Database、以及自建MySQL 、 PostgreSQL 或 Amazon Redshift。

AnalyticDB for PostgreSQL支持的各种数据迁移应用场景及相关操作如下：

操作	类型	场景
OSS 外部表的使用	迁移	通过OSS外部表将数据在AnalyticDB for PostgreSQL和OSS之间进行导入或者导出。
使用数据集成迁移及批量同步数据	同步/迁移	通过数据集成（Data Integration）在AnalyticDB for PostgreSQL中进行数据的导入或者导出。
使用 COPY 命令迁移数据到AnalyticDB for PostgreSQL	迁移	通过\COPY命令，将本地的文本文件的数据导入到AnalyticDB for PostgreSQL中。
使用DTS将RDS PostgreSQL实时同步数据至AnalyticDB for PostgreSQL	同步/迁移	通过数据传输服务（DTS）同步RDS PostgreSQL 数据到AnalyticDB for PostgreSQL。
使用DTS 将RDS MySQL数据同步至AnalyticDB for PostgreSQL	同步/迁移	通过数据传输服务（DTS）同步RDS MySQL数据到AnalyticDB for PostgreSQL。
使用DTS 将ECS的自建MySQL数据同步至AnalyticDB for PostgreSQL	同步/迁移	通过数据传输服务（DTS）同步ECS自建MySQL数据到AnalyticDB for PostgreSQL

操作	类型	场景
使用DTS将通过专线/VPN网关/智能网关接入的自建MySQL数据同步至AnalyticDB for PostgreSQL	同步/迁移	通过数据传输服务（DTS）同步专线/VPN网关/智能网关接入的自建MySQL数据同步至AnalyticDB for PostgreSQL
Amazon Redshift迁移数据到AnalyticDB for PostgreSQL	第三方迁移	通过Amazon S3和阿里云OSS将Amazon Redshift的数据导入到AnalyticDB for PostgreSQL中。
使用 rds_dbsync 迁移/同步MySQL 数据到AnalyticDB for PostgreSQL	同步/迁移	通过rds_dbsync的mysql2pgsql工具将本地MySQL中的表导入到AnalyticDB for PostgreSQL中。
使用 rds_dbsync 迁移/同步PostgreSQL数据到AnalyticDB for PostgreSQL	同步/迁移	通过rds_dbsync的pgsql2pgsql工具将AnalyticDB for PostgreSQL/Greenplum Database/PostgreSQL/PPAS中的表导AnalyticDB for PostgreSQL中。

1. 迁移：是指将各种数据库实例或者本地存储的数据迁移到AnalyticDB for PostgreSQL。
2. 同步：是指将其他数据库中的数据实时同步到AnalyticDB for PostgreSQL。

6.2 使用数据集成迁移及批量同步数据

数据集成（Data Integration）是阿里巴巴集团提供的数据同步平台。该平台具备可跨异构数据存储系统、可靠、安全、低成本、可弹性扩展等特点，可为20多种数据源提供不同网络环境下的离线（全量/增量）数据进出通道。详情请参见 支持的数据源类型。

本文将为您介绍如何通过数据集成在 AnalyticDB for PostgreSQL 中进行数据的 导入 和 导出 操作。

应用场景

- AnalyticDB for PostgreSQL 可以通过数据集成的同步任务将数据同步到到其他的数据源中，并对数据进行相应的处理。
- 可以通过数据集成的同步任务将处理好的其他数据源数据同步到 AnalyticDB for PostgreSQL 。

流程概述

1. 数据源端新建表。
2. 新增数据源。
3. 向导模式或脚本模式配置同步任务。
4. 运行同步任务，检查目标端的数据质量。

准备工作

1. 开通阿里云主账号，并建好账号的访问秘钥，即 AccessKeys。
2. 使用主账号登录 DataWorks Dataworks。
3. 创建项目。您可以在项目中协作完成工作流，共同维护数据和任务等，因此使用 DataWorks 之前需要先创建一个项目。

> 注意:
>
> 如果您想通过子账号创建数据集成任务，可以赋予其相应的权限。详情请参见准备 RAM 子账号和项目成员管理。

AnalyticDB for PostgreSQL 准备工作

1. 进行数据导入操作前，首先通过 PostgreSQL 客户端创建好 AnalyticDB for PostgreSQL 中需要迁入数据的目标数据库和表。
2. 当需要迁出数据的源数据库为AnalyticDB for PostgreSQL时，请AnalyticDB for PostgreSQL的管理控制台进行 IP 白名单设置，详情请参见 添加白名单。

   通过 AnalyticDB for PostgreSQL 的管理控制台进入白名单设置添加下列 IP 地址:

```
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
```

   如下图所示:

> 注意:
>
> 若使用自定义资源组调度 AnalyticDB for PostgreSQL 数据同步任务，必须把自定义资源组的机器 IP 也加AnalyticDB for PostgreSQL 的白名单中。

新增数据源

> 注意:
>
> 只有项目管理员角色才能够新建数据源，其他角色的成员仅能查看数据源。

本文以添加 AnalyticDB for PostgreSQL 的数据源为例：

1. 以开发者身份进入 DataWorks 管理控制台，单击对应项目操作栏中的 进入工作区。
2. 单击顶部菜单栏中的 数据集成，导航至 数据源 页面。
3. 单击 新增数据源，弹出支持的数据源类型。如下图：

4. 选择数据源类型为 PostgreSQL。

   图 6-1: 有公网IP

   配置项说明：

   - 数据源类型：有公网 IP。
   - 数据源名称：由英文字母、数字、下划线组成且需以字符或下划线开头，长度不超过 60 个字符。
   - 数据源描述：对数据源进行简单描述，不得超过 80 个字符。
   - JDBC URL：JDBC 连接信息，格式为：`jdbc:postgresql://ServerIP:Port/Database`。
   - 用户名/密码：数据库对应的用户名和密码。
5. 单击 测试连通性。
6. 测试连通性通过后，单击 确定。

向导模式或脚本模式配置同步任务

通过数据集成向导模式导入数据

以向导模式将 MaxCompute（原 ODPS） 数据同步到 AnalyticDB for PostgreSQL 为例：

1. 进入 数据集成 页面，新建同步任务，如下图所示：

   - 向导模式：向导模式是可视化界面配置同步任务，共五步：选择来源，选择目标，字段映射，通道控制，预览保存。在每个不同的数据源之间，这几步的界面可能有不同的内容，向导模式可以转换成脚本模式。
   - 脚本模式：进入脚本界面你可以选择相应的模板，此模板包含了同步任务的主要参数，将相关的信息填写完整，但是脚本模式不能转化成向导模式。

2. 选择来源。

   选择odps数据源及源头表hpg，数据浏览默认是收起的，选择后单击 下一步，如下图所示：

3. 选择目标。

   选择AnalyticDB for PostgreSQL数据源及目标表public.person，选择后单击下一步，如下图所示：

   - 导入前准备语句：执行数据同步任务之前率先执行的 SQL 语句，目前向导模式只允许执行一条 SQL 语句，脚本模式可以支持多条 SQL 语句，例如清除旧数据。
   - 导入后准备语句：执行数据同步任务之后执行的 SQL 语句，目前向导模式只允许执行一条 SQL 语句，脚本模式可以支持多条 SQL 语句，例如加上某一个时间戳。
   - 主键冲突：`insert into` 指当主键/唯一性索引冲突，数据集成视为脏数据进行处理。

4. 映射字段。

   单击 下一步，选择字段的映射关系。需对字段映射关系进行配置，左侧 源头表字段 和右侧 目标表字段 为一一对应的关系，如下图所示：

   映射字段数据源端 添加一行 的功能，如下所示：

   - 可以输入常量，输入的值需要使用英文单引号包括，如’abc’、’123’等。
   - 可以配合调度参数使用，如 `${bdp.system.bizdate}` 等。
   - 可以输入关系数据库支持的函数，如now()、count(1)等。
   - 如果您输入的值无法解析，则类型显示为’-’。
   - 不支持配置 MaxCompute 函数。

5. 通道控制。

   单击 下一步，配置作业速率上限和脏数据检查规则，如下图所示：

   - 作业速率上限：指数据同步作业可能达到的最高速率，其最终实际速率受网络环境、数据库配置等的影响。
   - 作业并发数：作业速率上限=作业并发数*单并发的传输速率。

     注意：

     当作业速率上限已选定的情况下，应该如何选择作业并发数？

     - 如果你的数据源是线上的业务库，建议您不要将并发数设置过大，以防对线上库造成影响。
     - 如果您对数据同步速率特别在意，建议您选择最大作业速率上限和较大的作业并发数。

6. 预览保存。

   完成上述配置后，上下滚动鼠标可查看任务配置，如若无误，单击 保存，如下图所示：

脚本模式配置导入同步任务

```
{
  "configuration": {
```

```
    "reader": {
      "plugin": "odps",
      "parameter": {
        "partition": "pt=${bdp.system.bizdate}",//分区信息
        "datasource": "odps_first",//数据源名，建议数据源都先添加数据源后再配
置同步任务，此配置项填写的内容必须要与添加的数据源名称保持一致
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "table": "hpg"//源端表名
      }
    },
    "writer": {
      "plugin": "postgresql",
      "parameter": {
        "postSql": [],//导入后准备语句
        "datasource": "l_PostGreSql",//数据源名，建议数据源都先添加数据源后
再配置同步任务，此配置项填写的内容必须要与添加的数据源名称保持一致
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "table": "public.person",//目标表名
        "preSql": []//导入前准备语句
      }
    },
    "setting": {
      "speed": {
        "concurrent": 7,//并发数
        "mbps": 7//数率最高上限
      }
    }
  },
  "type": "job",
  "version": "1.0"
```

通过数据集成向导模式导出数据

以向导模式配置 AnalyticDB for PostgreSQL 同步到 MaxCompute 为例：

1. 进入 数据集成 页面，新建同步任务，如下图所示：

2. 选择来源。

   选择AnalyticDB for PostgreSQL数据源及源头表public.person，数据浏览默认是收起的，选择后单击下一步，如下图所示：

   - 数据过滤：PostgreSQLReader 根据指定的 column、table、where 条件拼接 SQL，并根据这个 SQL 进行数据抽取。例如在做测试时，可以根据实际业务场景指定where，往往会选择当天的数据进行同步，可以将 where 条件指定为 id > 2 and sex = 1。

     where 条件可以有效地进行业务增量同步。where条件不配置或者为空，视作全表同步数据。
   - 切分键：PostgreSQL Reader 进行数据抽取时，如果指定 splitPk，表示您希望使用 splitPk 代表的字段进行数据分片，数据同步因此会启动并发任务进行数据同步，这样可以大大提供数据同步的效能。
     - 推荐 splitPk 用户使用表主键，因为表主键通常情况下比较均匀，因此切分出来的分片也不容易出现数据热点。
     - 目前 splitPk 仅支持整型数据切分，不支持字符串、浮点、日期等其他类型。如果您指定其他非支持类型，忽略 plitPk 功能，使用单通道进行同步。
     - 如果 splitPk 不填写，包括不提供 splitPk 或者 splitPk 值为空，数据同步视作使用单通道同步该表数据 。

3. 选择目标。

   选择odps数据源及目标表hpg，选择后单击下一步，如下图所示：

4. 映射字段。

   单击 下一步，选择字段的映射关系。需对字段映射关系进行配置，左侧 源头表字段 和右侧 目标表字段 为一一对应的关系，如下图所示：

5. 通道控制。

   单击 下一步，配置作业速率上限和脏数据检查规则，如下图所示：

6. 预览保存。

完成上述配置后，上下滚动鼠标可查看任务配置，如若无误，单击 保存。

脚本模式配置导出同步任务

```
{
  "configuration": {
    "reader": {
      "plugin": "postgresql",
      "parameter": {
        "datasource": "l_PostGreSql",//数据源名，建议数据源都先添加数据源后再配置同步任务，此配置项填写的内容必须要与添加的数据源名称保持一致
        "table": "public.person",//源端表名
        "where": "",//过滤条件
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "splitPk": ""//切分键
      }
    },
    "writer": {
      "plugin": "odps",
      "parameter": {
        "datasource": "odps_first",//数据源名，建议数据源都先添加数据源后再配置同步任务，此配置项填写的内容必须要与添加的数据源名称保持一致
        "column": [
          "id",
          "name",
          "year",
          "birthdate",
          "ismarried",
          "interest",
          "salary"
        ],
        "table": "hpg",//目标表名
        "truncate": true,
        "partition": "pt=${bdp.system.bizdate}"//分区信息
      }
    },
    "setting": {
      "speed": {
        "mbps": 5,//数率最高上限
        "concurrent": 5//并发数
      }
    }
  },
  "type": "job",
  "version": "1.0"
}
```

运行同步任务，检查目标端的数据质量

运行结果，如下图所示：

> **注意:**
>
> 同步任务保存后，直接单击 运行，任务会立刻运行或单击右边的 提交，将同步任务提交到调度系统中，调度系统会按照配置属性在从第二天开始自动定时执行，相关调度的配置请参见 调度配置介绍。

参考信息

其他的配置同步任务详细信息请参见下述文档:

- 各数据源 Reader 的配置。
- 各数据源 Writer 的配置。

6.3 使用 rds_dbsync 迁移/同步 MySQL 数据到AnalyticDB for PostgreSQL

rds_dbsync

rds_dbsync 为开源的数据同步/迁移工具，其 mysql2pgsql 功能支持不落地的把MySQL中的表迁移到AnalyticDB for PostgreSQL/Greenplum Database/PostgreSQL/PPAS。此工具的原理是，同时连接源端MySQL数据库和目的端数据库，从MySQL库中通过查询得到要导出的数据，然后通过 COPY命令导入到目的端。此工具支持多线程导入（每个工作线程负责导入一部分数据库表）。

参数配置

修改配置文件my.cfg、配置源和目的库连接信息。

- 源库MySQL的连接信息如下:

> **注意:**
>
> 源库MySQL的连接信息中，用户需要有对所有用户表的读权限。

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

```
encoding = "utf8"
```

- 目的库pgsql（包括Postgresql、PPAS和AnalyticDB for PostgreSQL）的连接信息如下：

  注意：

  目的库pgsql的连接信息，用户需要对目标表有写的权限。

```
[desc.pgsql]
connect_string = "host=192.168.1.2 dbname=test port=3432  user=test
 password=pgsql"
```

mysql2pgsql用法

mysql2pgsql的用法如下所示：

```
./mysql2pgsql -l <tables_list_file> -d -n -j <number of threads> -s <
schema of target table>
```

参数说明：

- -l：可选参数，指定一个文本文件，文件中含有需要同步的表；如果不指定此参数，则同步配置文件中指定数据库下的所有表。`<tables_list_file>`为一个文件名，里面含有需要同步的表集合以及表上查询的条件，其内容格式示例如下：

  ```
  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：可选参数，表示只生成目的表的建表DDL语句，不实际进行数据同步。
- -n：可选参数，需要与-d一起使用，指定在DDL语句中不包含表分区定义。
- -j：可选参数，指定使用多少线程进行数据同步；如果不指定此参数，会使用5个线程并发。
- -s：可选参数，指定目标表的schema，目前仅支持设定为public。

典型用法

全库迁移

全库迁移的操作步骤如下所示：

1. 通过如下命令，获取目的端对应表的DDL。

   ```
   ./mysql2pgsql -d
   ```

2. 根据这些DDL，再加入Distribution Key等信息，在目的端创建表。

3. 执行如下命令，同步所有表：

```
./mysql2pgsql
```

此命令会把配置文件中所指定数据库中的所有MySQL 表数据迁移到目的端。过程中使用5个线程（即缺省线程数为5），读取和导入所有涉及的表数据。

部分表迁移

1. 编辑一个新文件 tab_list.txt，放入如下内容：

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

2. 执行如下命令，同步指定的t1和t2表（注意t2表只迁移符合c1 > 138888条件的数据）：

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

下载与说明

- 下载mysql2pgsql二进制安装包下载，请单击这里。
- 查看mysql2pgsql源码编译说明，请单击这里。

6.4 使用 rds_dbsync 迁移/同步PostgreSQL数据到AnalyticDB for PostgreSQL

开源工具 rds_dbsync的pgsql2pgsql功能，支持把AnalyticDB for PostgreSQL/Greenplum Database/PostgreSQL/PPAS中的表迁AnalyticDB for PostgreSQL/Greenplum Database/PostgreSQL/PPAS。

pgsql2pgsql支持的功能

pgsql2pgsql支持如下功能：

- PostgreSQL/PPAS/Greenplum Database/AnalyticDB for PostgreSQL全量数据迁移到PostgreSQL/PPAS/Greenplum Database/AnalyticDB for PostgreSQL。
- PostgreSQL/PPAS（版本大于9.4）全量+增量迁移到PostgreSQL/PPAS。

参数配置

修改配置文件my.cfg、配置源和目的库连接信息。

- 源库pgsql连接信息如下所示：

注意：

源库pgsql的连接信息中，用户最好是对应DB的owner。

```
[src.pgsql]
connect_string = "host=192.168.1.1 dbname=test port=3432 user=test
password=pgsql"
```

- 本地临时Database pgsql连接信息如下所示：

```
[local.pgsql]
connect_string = "host=192.168.1.2 dbname=test port=3432 user=test2
password=pgsql"
```

- 目的库pgsql连接信息如下所示：

注意：

目的库pgsql的连接信息，用户需要对目标表有写权限。

```
[desc.pgsql]
connect_string = "host=192.168.1.3 dbname=test port=3432 user=test3
password=pgsql"
```

注意：

- 如果要做增量数据同步，连接源库需要有创建replication slot的权限。
- 由于PostgreSQL 9.4及以上版本支持逻辑流复制，所以支持作为数据源的增量迁移。打开下列内核参数才能让内核支持逻辑流复制功能。

```
wal_level = logical
max_wal_senders = 6
max_replication_slots = 6
```

pgsql2pgsql用法

全库迁移

进行全库迁移，请执行如下命令：

```
./pgsql2pgsql
```

迁移程序会默认把对应pgsql库中所有用户的表数据将迁移到pgsql。

状态信息查询

连接本地临时Database，可以查看到单次迁移过程中的状态信息。这些信息被放在表db_sync_status中，包括全量迁移的开始和结束时间、增量迁移的开始时间和增量同步的数据情况。

下载与说明

- 下载rds_dbsync二进制安装包，请单击这里。
- 查看rds_dbsync源码编译说明，请单击这里。

6.5 Amazon Redshift迁移数据到AnalyticDB for PostgreSQL

本文描述从Amazon Redshift迁移数据到AnalyticDB for PostgreSQL的整体过程。

总体步骤

从Redshift迁移数据到AnalyticDB for PostgreSQL包含如下步骤：

1. 资源和环境准备，执行操作前需提前准备Amazon Redshift、Amazon S3（Amazon Simple Storage Service）、AnalyticDB for PostgreSQL和阿里云对象存储服务（OSS）的相关资源。
2. 将Redshift的数据导入到Amazon S3中。
3. 使用OSSImport将Amazon S3中CSV格式的数据文件导入到OSS。
4. 在目标AnalyticDB for PostgreSQL中创建和源Redshift对应的对象，包括模式（Schema）、表（Table）、视图（View）和函数（Function）。
5. 使用OSS外部表将数据导入到AnalyticDB for PostgreSQL。

整体迁移路径如下：

AWS上的准备工作

准备用户访问S3的安全凭证

包括如下信息：

- 访问密钥ID（AccessKeyID）和秘密访问密钥（Secret AccessKey）。
- S3的Endpoint，例如s3.ap-southeast-2.amazonaws.com。
- S3的Bucket名称，例如alibaba-hybrid-export。

导出数据格式约定

- 导出文件为CSV格式。
- 导出文件不得大于50MB。
- 导出文件中列的顺序必须和建表语句中列的顺序一致。
- 导出文件的数量最好和AnalyticDB for PostgreSQL计算组的数量一致或者是计算组数量的整数倍。

推荐的Redshift UNLOAD命令选项

经过大量的实践，我们建议使用类似如下的Redshift UNLOAD选项将数据导入到S3中：

```
unload ('select * from test')
to 's3://xxx-poc/test_export_'
```

```
access_key_id '<Your access key id>'
secret_access_key '<Your access key secret>'
DELIMITER AS ','
ADDQUOTES
ESCAPE
NULL AS 'NULL'
MAXFILESIZE 50 mb;
```

在上述样例中，推荐使用如下选项：

```
DELIMITER AS ','
ADDQUOTES
ESCAPE
NULL AS 'NULL'
MAXFILESIZE 50 mb
```

从Redshift导出DDL语句

从Redshift导出所有的DDL语句，包括但不限于创建模式、创建表、创建视图和创建函数的语句。

阿里云上的准备工作

准备阿里云RAM账户

- RAM账户ID
- RAM账户密码
- RAM账户AccessKeyID
- RAM账户AccessKeySecret

创建OSS存储空间（Bucket）

在AWS S3 Bucket所在地域，比如华北2（北京），创建一个OSS存储空间。OSS存储空间创建完成后，可以从OSS的控制台获取存储空间的访问域名，本文中会使用到ESC的VPC网络访问（内网）的访问域名信息。使用内网传输，可以保障数据传输的速度和安全性。

下载安装OSSImport

1. 在AWS S3 Bucket所在地域，创建一个ECS实例。我们选择创建带宽为100Mbps，系统镜像为Windows X64的ECS实例。
2. 在ECS系统中下载并安装单机模式的OSSImport。OSSImport的最新版本，可从此处获取。
3. 单机模式的OSSImport软件包解压后，软件的文件结构如下：

```
ossimport
├── bin
│   └── ossimport2.jar  # 包括Master、Worker、Tracker、Console四个模块的总
jar
├── conf
│   ├── local_job.cfg   # 单机Job配置文件
│   └── sys.properties  # 系统运行参数配置文件
├── console.bat           # Windows命令行，可以分布执行调入任务
```

```
├── console.sh              # Linux命令行，可以分布执行调入任务
├── import.bat              # Windows一键导入，执行配置文件为conf/local_job.
cfg配置的数据迁移任务，包括启动、迁移、校验、重试
├── import.sh               # Linux一键导入，执行配置文件为conf/local_job.cfg
配置的数据迁移任务，包括启动、迁移、校验、重试
├── logs                    # 日志目录
└── README.md               # 说明文档，强烈建议使用前仔细阅读
```

使用OSSImport将数据从S3导入到OSS中

配置OSSImport

在本文中，我们采用单机模式的OSSImport。请参考如下样例修改配置文件conf/local_job.cfg，请确保仅修改本样例提及的参数。关于OSSImport配置的详细信息，请参考说明及配置。

```
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=
isSkipExistFile=true
```

启动OSSImport迁移任务

在单机模式的OSSImport中，执行import.bat批处理文件启动迁移任务。

查看迁移任务的状态

在数据迁移过程中，您可以通过命令执行窗口查看任务的执行状态。另外，你还可通过Windows系统的任务管理器查看带宽的占用情况。

在本样例中，ECS和OSS存储空间位于相同的地域，采用内网传输，不受网速的限制；S3到ECS采用外网传输，有网速限制。数据的上传速度受限于下载速度，因此从ECS到OSS存储空间的上传速度几乎和S3到ECS的下载速度相同。

任务失败重试（可选）

由于网络或者其他因素，迁移任务可能失败。在ECS Windows系统中的CMD命令窗口中执行concole.bat retry命令重试任务。失败任务重试仅仅重新执行失败的子任务，不会重试已成功的子任务。

检查OSS存储空间的文件（可选）

您可在OSS控制台检查导入的数据文件。我们推荐使用ossbrowser客户端工具来查看和管理OSS存储空间中的文件。ossbrowser可从此处获取。

整理CSV文件中的数据（可选）

> 说明:
> 本操作仅仅提供一个参考样例，为可选步骤，您可以根据您的业务需求整理CSV文件。

- 将CSV文件中的`NULL`替换成空格。
- 将CSV文件中的`\,`替换成`,`。

推荐在本地进行数据整理。您先通过ossbrowser将文件下载到ECS中，进行数据整理。然后再将整理后的文件上传到一个新建的OSS存储空间中，以区别于原来从S3下载的CSV文件。无论是上传还是下载CSV文件，我们都推荐使用OSS的内网Endpoint，以降低内网流量的开销。

将Redshift的DDL语句转换成AnalyticDB for PostgreSQL的DDL语句

在创建AnalyticDB for PostgreSQL数据库对象之前，我们需要做一些必要的准备工作。主要是将上述步骤导出的Redshift DDL语句转换成AnalyticDB for PostgreSQL语法的DDL语句。下面我们将简要地介绍这些转换规则。

CREATE SCHEMA

按照AnalyticDB for PostgreSQL语法标准创建模式，将其保存为*create_schema.sql*。如下为具体的样例：

```
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_external_table
  AUTHORIZATION xxxpoc;
```

CREATE FUNCTION

由于AnalyticDB for PostgreSQL不兼容Redshift的某些SQL函数，因此你需要定制或者重写这些函数。涉及的函数举例如下：

- `CONVERT_TIMEZONE(a,b,c)`，使用如下语句替换：

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

- `GETDATE()`，使用如下语句替换：

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

- 替换或优化用户定义函数（UDF）。

  例如，Redshift的SQL函数如下：

```
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;
```

  替换成如下AnalyticDB for PostgreSQL函数，可以提升性能。

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

- 其他Redshift标准的函数。

  在具体的实践中，您可以在Functions and Operators in PostgreSQL8.2中查询标准的PostgreSQL函数的用法，修改或者自行实现Redshift和AnalyticDB for PostgreSQL不兼容的函数。如下为一些相关资源：

  - ISNULL()
  - DATEADD()
  - DATEDIFF()
  - REGEXP_COUNT()
  - LEFT()
  - RIGHT()

CREATE TABLE

- 修改压缩编码。AnalyticDB for PostgreSQL目前并不支持所有的Redshift压缩编码。不支持的压缩编码包括：
  - BYTEDICT
  - DELTA
  - DELTA32K
  - LZO
  - MOSTLY8
  - MOSTLY16
  - MOSTLY32
  - RAW (no compression)
  - RUNLENGTH
  - TEXT255
  - TEXT32K
  - ZSTD

  必须删除Redshift建表语句中的`ENCODE XXX`，用如下子句代替。

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

- 修改分布键。Redshift支持三种分布键（分配），具体请参见分配方式。您需按照如下规则修改分布键。
  - EVEN分配（DISTSTYLE EVEN）：用`distributed randomly`代替。
  - KEY分配（DISKEY）：用`distributed by (colname1,...)`代替。
  - ALL分配（ALL）：不支持，直接删除。
- 修改排序键（SortKey）。删除Redshift的排序键子句`[ COMPOUND | INTERLEAVED ] SORTKEY (column_name [, ...] ) ]`中的`COMPOUND`或者`INTERLEAVED`选项，使用如下子句代替：

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

  样例1

  Redshift的CREATE TABLE语句：

  ```
  CREATE TABLE schema1.table1
  (
   filed1 VARCHAR(100) ENCODE lzo,
   filed2 INTEGER DISTKEY,
   filed3 INTEGER,
  ```

```
 filed4 BIGINT ENCODE lzo,
 filed5 INTEGER,
)
INTERLEAVED SORTKEY
(
 filed1,
 filed2
);
```

转换成AnalyticDB for PostgreSQL的CREATE TABLE语句：

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

样例2

Redshift的CREATE TABLE语句，包含ENCODE和SORTKEY选项：

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

转换成AnalyticDB for PostgreSQL的CREATE TABLE语句：

```
CREATE TABLE schema2.table2
(
 filed1 VARCHAR(50),
 filed2 VARCHAR(50),
 filed3 VARCHAR(20),
)
WITH(APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib)
DISTRIBUTED randomly
SORTKEY
(
 filed1
);
```

CREATE VIEW

同样需要将Redshift的CREATE VIEW语句转换成符合AnalyticDB for PostgreSQL语法的SQL语句，转换规则和CREATE TABLE的转换规则类似。

创建和配置AnalyticDB for PostgreSQL实例

根据如下内容，创建并配置实例。

- 创建实例
- 设置白名单
- 设置账号

创建数据库对象

参考客户端访问实例，使用psql或者pgAdmin III 1.6.3客户端，登录数据库。

按照上述转换规则，将Redshift的DDL语句转换成符合AnalyticDB for PostgreSQL语法规则的DDL语句。然后执行这些DDL语句创建数据库对象。

CREATE EXTERNAL TABLE

AnalyticDB for PostgreSQL支持通过OSS外部表（即gpossext功能），将数据并行从OSS导入或导出到OSS，并支持通过gzip进行OSS外部表文件压缩，大量地节省存储空间及成本。请参考OSS 外部表的使用，创建OSS外部表。

使用INSERT INTO脚本导入数据

在完成OSS外部表和目标数据库各个对象的创建后，您需要准备`INSERT`脚本，用于将OSS外部表的数据插入到AnalyticDB for PostgreSQL目标表中。请将该脚本保存为*insert.sql*文件，并执行该脚本。

插入语句的格式为：`INSERT INTO <TABLE NAME> SELECT * FROM <OSS EXTERNAL TABLE NAME>;`

例如：

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

导入数据后，请使用SELECT语句查询导入后的数据并和源数据进行对比分析，验证导入前后数据的一致性。

使用VACUUM脚本清理数据库

在将OSS外部表导入AnalyticDB for PostgreSQL后，您还需要使用`VACUUM`命令清理一下数据库，请将VACUUM脚本存储为`vacuum.sql`文件。关于VACUUM的用法，请参见VACUUM。

6.6 使用DTS 将RDS MySQL数据同步至AnalyticDB for PostgreSQL

数据传输服务（Data Transmission Service，简称DTS）支持将MySQL数据同步至AnalyticDB for PostgreSQL。通过DTS提供的数据同步功能，可以轻松实现数据的流转，将企业数据集中分析。

前提条件

- 数据同步的源RDS实例的数据库类型为MySQL，数据库引擎版本为5.1、5.5、5.6或5.7版本。
- 源库中待同步的数据表，必须有主键。

数据同步功能限制

- 同步对象仅支持数据表，暂不支持非数据表的对象。
- 暂不支持结构迁移功能。
- 不支持JSON、GEOMETRY、CURVE、SURFACE、MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTION、BYTEA类型的数据同步。

支持的同步语法

- DML操作：INSERT、UPDATE、DELETE。
- DDL操作：ALTER TABLE。目前仅支持ADD COLUMN、DROP COLUMN、MODIFY COLUMN、CHANGE COLUMN。

> 说明：
>
> - 不支持COMMENT内容的同步。
> - 不支持CREATE TABLE和DROP TABLE操作。如您需要新增表，则需要通过修改同步对象操作来新增对应表的同步，详情请参考新增同步对象。

支持的同步架构

- 1对1单向同步。
- 1对多单向同步。
- 多对1单向同步。

术语/概念对应关系

MySQL中的术语/概念	AnalyticDB for PostgreSQL中的术语/概念
Database	Schema
Table	Table

操作步骤一 在目标实例中创建对应的数据结构

根据源RDS实例中待迁移数据表的数据结构，在目标AnalyticDB for PostgreSQL实例中创建数据库、Schema及数据表，详情请参考AnalyticDB for PostgreSQL基础操作。

说明:
MySQL的timestamp类型对应AnalyticDB for PostgreSQL的timestamp with time zone类型。

操作步骤二 购买数据同步实例

1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 在页面右上角，单击创建同步作业。
4. 在数据传输服务购买页面，选择付费类型为预付费或按量付费。
   - 预付费：属于预付费，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。
   - 按量付费：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。
5. 选择数据同步实例的参数配置信息，参数说明如下表所示。

参数配置区	参数项	说明
基本配置	功能	选择数据同步。
	源实例	选择MySQL。
	源实例地域	选择数据同步链路中源RDS实例的地域。  说明: 订购后不支持更换地域，请谨慎选择。
	目标实例	选择AnalyticDB for PostgreSQL。

参数配置区	参数项	说明
	目标实例地域	选择数据同步链路中目标AnalyticDB for PostgreSQL实例的地域。 说明: 订购后不支持更换地域，请谨慎选择。
	同步拓扑	数据同步支持的拓扑类型，选择单向同步。 说明: 当前仅支持单向同步。
	网络类型	数据同步服务使用的网络类型，目前固定为专线。
	同步链路规格	数据传输为您提供了不同性能的链路规格，以同步的记录数为衡量标准。详情请参考数据同步规格说明。 说明: 建议生产环境选择small及以上规格。
购买量	购买数量	一次性购买数据同步实例的数量，默认为1，如果购买的是按量付费实例，一次最多购买9 条链路。

6. 单击立即购买，根据提示完成支付流程。

操作步骤三 配置数据同步

1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 定位至已购买的数据同步实例，单击配置同步链路。
4. 配置同步通道的源实例及目标实例信息。

配置项目	配置选项	配置说明
任务名称	-	· DTS为每个任务自动生成一个任务名称，任务名称没有唯一性要求。 · 您可以根据需要修改任务名称，建议为任务配置具有业务意义的名称，便于后续的任务识别。
源实例信息	实例类型	选择RDS实例。
	实例地区	购买数据同步实例时选择的源实例地域信息，不可变更。
	实例ID	选择作为数据同步源的RDS实例ID。

配置项目	配置选项	配置说明
	数据库账号	填入源RDS的数据库账号。 说明: 当源RDS实例的数据库类型为 MySQL 5.5或MySQL 5.6时，没有 数据库账号和 数据库密码配置选项。
	数据库密码	填入数据库账号对应的密码。
	连接方式	根据需求选择非加密连接或SSL安全连接，本案例选择为非加密连接。 说明: 选择SSL安全连接时，需要提前开启RDS实例的SSL加密功能，详情请参考设置SSL加密。
目标实例信息	实例类型	固定为AnalyticDB for PostgreSQL，无需设置。
	实例地区	购买数据同步实例时选择的目标实例地域信息，不可变更。
	实例ID	选择作为数据同步目标的AnalyticDB for PostgreSQL实例ID。
	数据库名称	填入同步目标表所属的数据库名称。
	数据库账号	填入目标AnalyticDB for PostgreSQL实例的数据库账号。 说明: 数据库账号须具备SELECT、INSERT、UPDATE、DELETE、COPY、TRUNCATE、TABLE权限。
	数据库密码	填入数据库账号对应的密码。

5. 单击页面右下角的授权白名单并进入下一步。
6. 配置同步策略及对象信息。

配置项目	配置选项	配置说明
同步策略配置	同步初始化	选择全量数据初始化。 说明: 将源实例中已经存在同步对象的数据在目标实例中初始化，作为后续增量同步数据的基线数据。

配置项目	配置选项	配置说明
	目标已存在表的处理模式	· 预检查检测并拦截（默认勾选） 在预检查阶段执行目标表是否为空的检查项目，如果有数据直接在预检查的目标表是否为空的检查项中检测并拦截报错。 · 清空目标表的数据 在预检查阶段跳过目标表是否为空的检查项目。全量初始化之前将目标表的数据清空。适用于完成同步任务测试后的正式同步场景。 · 不做任何操作 在预检查阶段跳过目标表是否为空的检查项目。全量初始化时直接追加迁移数据。适用于多张表同步到一张表的汇总同步场景。
	同步操作类型	· Insert · Update · Delete · AlterTable 说明: 根据业务需求选择数据同步的操作类型。
选择同步对象	-	同步对象的选择粒度为表。 如果需要目标表中列信息与源表不同，则需要使用DTS的字段映射功能，详情请参考库表列映射。 说明: 不支持CREATE TABLE操作，您需要通过修改同步对象操作来新增对应表的同步，详情请参考新增同步对象。

7. 上述配置完成后单击页面右下角的预检查并启动。

说明:

· 在数据同步任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动数据同步任务。

- 如果预检查失败，单击具体检查项后的，查看具体的失败详情。根据失败原因修复后，重新进行预检查。

8. 在预检查对话框中显示预检查通过后，关闭预检查对话框，该同步作业的同步任务正式开始。
9. 等待该同步作业的链路初始化完成，直至状态处于同步中。

   您可以在 数据同步页面，查看数据同步状态。

6.7 使用DTS 将ECS的自建MySQL数据同步至AnalyticDB for PostgreSQL

数据传输服务（Data Transmission Service，简称DTS）支持将ECS上的自建MySQL数据同步至AnalyticDB for PostgreSQL。通过DTS提供的数据同步功能，可以轻松实现数据的流转，将企业数据集中分析。

前提条件

- ECS上的自建MySQL数据库版本为5.1、5.5、5.6或5.7版本。
- 源库中待同步的数据表，必须有主键。

数据同步功能限制

- 同步对象仅支持数据表，暂不支持非数据表的对象。
- 暂不支持结构迁移功能。
- 不支持JSON、GEOMETRY、CURVE、SURFACE、MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTION、BYTEA类型的数据同步。

支持的同步语法

- DML操作：INSERT、UPDATE、DELETE。
- DDL操作：ALTER TABLE、ADD COLUMN、DROP COLUMN、RENAME COLUMN。

> 说明:
> 不支持CREATE TABLE和DROP TABLE操作。如您需要新增表，则需要通过修改同步对象操作来新增对应表的同步，详情请参考新增同步对象。

支持的同步架构

- 1对1单向同步。
- 1对多单向同步。
- 多对1单向同步。

术语/概念对应关系

MySQL中的术语/概念	AnalyticDB for PostgreSQL中的术语/概念
Database	Schema
Table	Table

数据同步前准备工作

在正式操作数据同步之前，需要在ECS上的自建MySQL数据库上进行账号与Binlog的配置。

1. 在ECS上的MySQL数据库中创建用于数据同步的账号。

```
CREATE USER 'username'@'host' IDENTIFIED BY 'password';
```

参数说明：

- username：要创建的账号。
- host：指定该账号登录数据库的主机。如果是本地用户可以使用 localhost，如需该用户从任意主机登录，可以使用百分号（%）。
- password：该账号的登录密码。

例如，创建账号为dtsmigration，密码为Dts123456的账号从任意主机登录本地数据库，命令如下。

```
CREATE USER 'dtsmigration'@'%' IDENTIFIED BY 'Dts123456';
```

2. 给用于数据同步的数据库账号进行授权操作。

```
GRANT privileges ON databasename.tablename TO 'username'@'host' WITH
 GRANT OPTION;
```

参数说明：

- privileges：该账号的操作权限，如SELECT、INSERT、UPDATE等。如果要授权该账号所有权限，则使用ALL。
- databasename：数据库名。如果要授权该账号所有的数据库权限，则使用星号（*）。
- tablename：表名。如果要授权该账号所有的表权限，则使用星号（*）。
- username：要授权的账号名。
- host：授权登录数据库的主机名。如果是本地用户可以使用 localhost，如果想让该用户从任意主机登录，可以使用百分号（%）。
- WITH GRANT OPTION：授权该账号能使用GRANT命令，该参数为可选。

例如，授权账号dtsmigration对所有数据库和表的所有权限，并可以从任意主机登录本地数据库，命令如下。

```
GRANT ALL ON *.* TO 'dtsmigration'@'%';
```

3. 开启ECS上的MySQL数据库的binlog。

说明：

您可以使用如下命令查询数据库是否开启了binlog。如果查询结果为 log_bin=ON，那么数据库已开启binlog，您可以跳过本步骤。

```
show global variables like "log_bin";
```

a. 修改配置文件my.cnf中的如下参数。

```
log_bin=mysql_bin
binlog_format=row
server_id=大于 1 的整数
binlog_row_image=full //当本地 MySQL 版本大于 5.6 时，则需设置该项。
```

b. 修改完成后，重启MySQL进程。

```
$mysql_dir/bin/mysqladmin -u root -p shutdown
$mysql_dir/bin/safe_mysqld &
```

说明:

“mysql_dir”替换为您MySQL实际的安装目录。

操作步骤一 在目标实例中创建对应的数据结构

根据ECS上自建的MySQL数据库中待迁移数据表的数据结构，在目标AnalyticDB for PostgreSQL实例中创建数据库、Schema及数据表，详情请参考AnalyticDB for PostgreSQL基础操作。

操作步骤二 购买数据同步实例

1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 在页面右上角，单击创建同步作业。
4. 在数据传输服务购买页面，选择付费类型为预付费或按量付费。
   - 预付费：属于预付费，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。
   - 按量付费：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。

说明:

DTS产品价格请参考产品定价。

5. 选择数据同步实例的参数配置信息，参数说明如下表所示。

参数配置区	参数项	说明
基本配置	功能	选择数据同步。
	源实例	选择MySQL。

参数配置区	参数项	说明
	源实例地域	选择数据同步链路中源ECS实例所属地域。  说明: 订购后不支持更换地域，请谨慎选择。
	目标实例	选择AnalyticDB for PostgreSQL。
	目标实例地域	选择数据同步链路中目标AnalyticDB for PostgreSQL实例的地域。  说明: 订购后不支持更换地域，请谨慎选择。
	同步拓扑	数据同步支持的拓扑类型，选择单向同步。  说明: 当前仅支持单向同步。
	网络类型	数据同步服务使用的网络类型，目前固定为专线。
	同步链路规格	数据传输为您提供了不同性能的链路规格，以同步的记录数为衡量标准。详情请参考数据同步规格说明。  说明: 建议生产环境选择small及以上规格。
购买量	购买数量	一次性购买数据同步实例的数量，默认为1，如果购买的是按量付费实例，一次最多购买9 条链路。

6. 单击立即购买，根据提示完成支付流程。

操作步骤三 配置数据同步

1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 定位至已购买的数据同步实例，单击配置同步链路。

4. 配置同步通道的源实例及目标实例信息。

配置项目	配置选项	配置说明
任务名称	-	· DTS为每个任务自动生成一个任务名称，任务名称没有唯一性要求。 · 您可以根据需要修改任务名称，建议为任务配置具有业务意义的名称，便于后续的任务识别。
源实例信息	实例类型	选择ECS上的自建数据库。
	实例地区	购买数据同步实例时选择的源实例地域信息，不可变更。
	实例ID	选择作为同步数据源的ECS实例ID。
	数据库类型	购买数据同步实例时选择的数据库类型：MySQL，不可变更。
	端口	填入自建数据库的服务端口，默认为3306。
	数据库账号	填入连接ECS上自建MySQL数据库的账号。 说明: 需要具备 Replication slave, Replication client 及所有同步对象的 Select 权限。
	数据库密码	填入连接ECS上自建MySQL数据库账号对应的密码。
目标实例信息	实例类型	固定为AnalyticDB for PostgreSQL，无需设置。
	实例地区	购买数据同步实例时选择的目标实例地域信息，不可变更。
	实例ID	选择作为数据同步目标的AnalyticDB for PostgreSQL实例ID。
	数据库名称	填入同步目标表所属的数据库名称。
	数据库账号	填入目标AnalyticDB for PostgreSQL实例的数据库账号。 说明: 数据库账号须具备SELECT、INSERT、UPDATE、DELETE、COPY、TRUNCATE、ALTER TABLE权限。
	数据库密码	填入数据库账号对应的密码。

5. 单击页面右下角的授权白名单并进入下一步。

6. 配置同步策略及对象信息。

配置项目	配置选项	配置说明
同步策略配置	同步初始化	选择全量数据初始化。 说明: 将源实例中已经存在同步对象的数据在目标实例中初始化，作为后续增量同步数据的基线数据。
	目标已存在表的处理模式	· 预检查检测并拦截（默认勾选） 在预检查阶段执行目标表是否为空的检查项目，如果有数据直接在预检查的目标表是否为空的检查项中检测并拦截报错。 · 清空目标表的数据 在预检查阶段跳过目标表是否为空的检查项目。全量初始化之前将目标表的数据清空。适用于完成同步任务测试后的正式同步场景。 · 不做任何操作 在预检查阶段跳过目标表是否为空的检查项目。全量初始化时直接追加迁移数据。适用于多张表同步到一张表的汇总同步场景。
	同步操作类型	· Insert · Update · Delete · AlterTable 说明: 根据业务需求选择数据同步的操作类型。
选择同步对象	-	同步对象的选择粒度为表。 如果需要目标表中列信息与源表不同，则需要使用DTS的字段映射功能，详情请参考库表列映射。 说明: 不支持CREATE TABLE操作，您需要通过修改同步对象操作来新增对应表的同步，详情请参考新增同步对象。

7. 上述配置完成后单击页面右下角的预检查并启动。

> 说明:
> - 在数据同步任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动数据同步任务。
> - 如果预检查失败，单击具体检查项后的，查看具体的失败详情。根据失败原因修复后，重新进行预检查。

8. 在预检查对话框中显示预检查通过后，关闭预检查对话框，该同步作业的同步任务正式开始。
9. 等待该同步作业的链路初始化完成，直至状态处于同步中。

   您可以在 数据同步页面，查看数据同步状态。

6.8 使用DTS将通过专线/VPN网关/智能网关接入的自建MySQL数据同步至AnalyticDB for PostgreSQL

数据传输服务（Data Transmission Service，简称DTS）支持将通过专线/VPN网关/智能网关接入的自建MySQL数据同步至AnalyticDB for PostgreSQL。通过DTS提供的数据同步功能，可以轻松实现数据的流转，将企业数据集中分析。

前提条件

- 自建MySQL数据库版本为5.1、5.5、5.6或5.7版本。
- 源库中待同步的数据表，必须有主键。

数据同步功能限制

- 同步对象仅支持数据表，暂不支持非数据表的对象。
- 暂不支持结构迁移功能。
- 不支持JSON、GEOMETRY、CURVE、SURFACE、MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTION、BYTEA类型的数据同步。

支持的同步语法

- DML操作：INSERT、UPDATE、DELETE。
- DDL操作：ALTER TABLE、ADD COLUMN、DROP COLUMN、RENAME COLUMN。

> 说明:
> 不支持CREATE TABLE和DROP TABLE操作。如您需要新增表，则需要通过修改同步对象操作来新增对应表的同步，详情请参考新增同步对象。

支持的同步架构

- 1对1单向同步。
- 1对多单向同步。
- 多对1单向同步。

术语/概念对应关系

MySQL中的术语/概念	AnalyticDB for PostgreSQL中的术语/概念
Database	Schema
Table	Table

数据同步前准备工作

在正式操作数据同步之前，需要在自建MySQL数据库上进行账号与Binlog的配置。

1. 在自建MySQL数据库中创建用于数据同步的账号。

```
CREATE USER 'username'@'host' IDENTIFIED BY 'password';
```

参数说明：

- username：要创建的账号。
- host：指定该账号登录数据库的主机。如果是本地用户可以使用 localhost，如需该用户从任意主机登录，可以使用百分号（%）。
- password：该账号的登录密码。

例如，创建账号为dtsmigration，密码为Dts123456的账号从任意主机登录本地数据库，命令如下。

```
CREATE USER 'dtsmigration'@'%' IDENTIFIED BY 'Dts123456';
```

2. 给用于数据同步的数据库账号进行授权操作。

```
GRANT privileges ON databasename.tablename TO 'username'@'host' WITH
 GRANT OPTION;
```

参数说明：

- privileges：该账号的操作权限，如SELECT、INSERT、UPDATE等。如果要授权该账号所有权限，则使用ALL。
- databasename：数据库名。如果要授权该账号所有的数据库权限，则使用星号（*）。
- tablename：表名。如果要授权该账号所有的表权限，则使用星号（*）。
- username：要授权的账号名。
- host：授权登录数据库的主机名。如果是本地用户可以使用 localhost，如果想让该用户从任意主机登录，可以使用百分号（%）。
- WITH GRANT OPTION：授权该账号能使用GRANT命令，该参数为可选。

例如，授权账号dtsmigration对所有数据库和表的所有权限，并可以从任意主机登录本地数据库，命令如下。

```
GRANT ALL ON *.* TO 'dtsmigration'@'%';
```

3. 开启自建MySQL数据库的binlog。

说明：

您可以使用如下命令查询数据库是否开启了binlog。如果查询结果为 log_bin=ON，那么数据库已开启binlog，您可以跳过本步骤。

```
show global variables like "log_bin";
```

a. 修改配置文件my.cnf中的如下参数。

```
log_bin=mysql_bin
binlog_format=row
server_id=大于 1 的整数
binlog_row_image=full //当本地 MySQL 版本大于 5.6 时，则需设置该项。
```

b. 修改完成后，重启MySQL进程。

```
$mysql_dir/bin/mysqladmin -u root -p shutdown
$mysql_dir/bin/safe_mysqld &
```

说明:

“mysql_dir”替换为您MySQL实际的安装目录。

操作步骤一 在目标实例中创建对应的数据结构

根据自建MySQL数据库中待迁移数据表的数据结构，在目标AnalyticDB for PostgreSQL实例中创建数据库、Schema及数据表，详情请参考AnalyticDB for PostgreSQL基础操作。

说明:

MySQL的timestamp类型对应AnalyticDB for PostgreSQL的timestamp with time zone类型。

操作步骤二 购买数据同步实例

1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 在页面右上角，单击创建同步作业。
4. 在数据传输服务购买页面，选择付费类型为预付费或按量付费。
   - 预付费：属于预付费，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。
   - 按量付费：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。

说明:

DTS产品价格请参考产品定价。

5. 选择数据同步实例的参数配置信息，参数说明如下表所示。

参数配置区	参数项	说明
基本配置	功能	选择数据同步。
	源实例	选择MySQL。
	源实例地域	选择数据同步链路中作为数据源的自建MySQL数据库的专有网络所属地域。 说明: 订购后不支持更换地域，请谨慎选择。
	目标实例	选择AnalyticDB for PostgreSQL。
	目标实例地域	选择数据同步链路中目标AnalyticDB for PostgreSQL实例的地域。 说明: 订购后不支持更换地域，请谨慎选择。
	同步拓扑	数据同步支持的拓扑类型，选择单向同步。 说明: 当前仅支持单向同步。
	网络类型	数据同步服务使用的网络类型，目前固定为专线。
	同步链路规格	数据传输为您提供了不同性能的链路规格，以同步的记录数为衡量标准。详情请参考数据同步规格说明。 说明: 建议生产环境选择small及以上规格。
购买量	购买数量	一次性购买数据同步实例的数量，默认为1，如果购买的是按量付费实例，一次最多购买9 条链路。

6. 单击立即购买，根据提示完成支付流程。

操作步骤三 配置数据同步

1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 定位至已购买的数据同步实例，单击配置同步链路。

4. 配置同步通道的源实例及目标实例信息。

配置项目	配置选项	配置说明
任务名称	-	· DTS为每个任务自动生成一个任务名称，任务名称没有唯一性要求。 · 您可以根据需要修改任务名称，建议为任务配置具有业务意义的名称，便于后续的任务识别。
源实例信息	实例类型	选择通过专线/VPN网关/智能网关接入的自建数据库。
	实例地区	购买数据同步实例时选择的源实例地域信息，不可变更。
	对端专有网络	选择自建数据库接入的VPC ID。
	数据库类型	购买数据同步实例时选择的数据库类型：MySQL，不可变更。
	IP地址	填入自建MySQL数据库的服务器IP地址。
	端口	填入自建数据库的服务端口，默认为3306。
	数据库账号	填入连接自建MySQL数据库的账号。 说明: 需要具备 Replication slave, Replication client 及所有同步对象的 Select 权限。
	数据库密码	填入连接自建MySQL数据库账号对应的密码。
目标实例信息	实例类型	固定为AnalyticDB for PostgreSQL，无需设置。
	实例地区	购买数据同步实例时选择的目标实例地域信息，不可变更。
	实例ID	选择作为数据同步目标的AnalyticDB for PostgreSQL实例ID。
	数据库名称	填入同步目标表所属的数据库名称。
	数据库账号	填入目标AnalyticDB for PostgreSQL实例的数据库账号。 说明: 数据库账号须具备SELECT、INSERT、UPDATE、DELETE、COPY、TRUNCATE、ALTER TABLE权限。
	数据库密码	填入数据库账号对应的密码。

5. 单击页面右下角的授权白名单并进入下一步。

6. 配置同步策略及对象信息。

配置项目	配置选项	配置说明
同步策略配置	同步初始化	选择全量数据初始化。 说明: 将源实例中已经存在同步对象的数据在目标实例中初始化，作为后续增量同步数据的基线数据。
	目标已存在表的处理模式	· 预检查检测并拦截（默认勾选） 在预检查阶段执行目标表是否为空的检查项目，如果有数据直接在预检查的目标表是否为空的检查项中检测并拦截报错。 · 清空目标表的数据 在预检查阶段跳过目标表是否为空的检查项目。全量初始化之前将目标表的数据清空。适用于完成同步任务测试后的正式同步场景。 · 不做任何操作 在预检查阶段跳过目标表是否为空的检查项目。全量初始化时直接追加迁移数据。适用于多张表同步到一张表的汇总同步场景。
	同步操作类型	· Insert · Update · Delete · AlterTable 说明: 根据业务需求选择数据同步的操作类型。
选择同步对象	-	同步对象的选择粒度为表。 如果需要目标表中列信息与源表不同，则需要使用DTS的字段映射功能，详情请参考库表列映射。 说明: 不支持CREATE TABLE操作，您需要通过修改同步对象操作来新增对应表的同步，详情请参考新增同步对象。

7. 上述配置完成后单击页面右下角的预检查并启动。

   说明:

   - 在数据同步任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动数据同步任务。
   - 如果预检查失败，单击具体检查项后的，查看具体的失败详情。根据失败原因修复后，重新进行预检查。

8. 在预检查对话框中显示预检查通过后，关闭预检查对话框，该同步作业的同步任务正式开始。
9. 等待该同步作业的链路初始化完成，直至状态处于同步中。

   您可以在 数据同步页面，查看数据同步状态。

6.9 使用DTS将RDS PostgreSQL实时同步数据至AnalyticDB for PostgreSQL

数据传输服务（Data Transmission Service，简称DTS）支持将RDS PostgreSQL数据同步至AnalyticDB for PostgreSQL。通过DTS提供的数据同步功能，可以轻松实现数据的流转，将企业数据集中分析。

前提条件

- 源RDS PostgreSQL和目标AnalyticDB for PostgreSQL的数据表必须建有主键。
- 源库需为RDS PostgreSQL 9.4或10版本。

注意事项

- 不支持迁移使用C语言编写的function。
- 一个数据同步任务只能对一个数据库进行数据同步，如果有多个数据库需要同步，则需要为每个数据库创建数据同步任务。
- 如果同步对象为数据库，在同步过程中，在待同步的数据库中创建新表时，需要在RDS PostgreSQL中对新创建的表执行如下语句：

  ```
  ALTER TABLE schema.table REPLICA IDENTITY FULL;
  ```

  说明:

  将schema和table替换成真实的schema名和表名。

- 目前不支持BIT、VARBIT、JSON、GEOMETRY、ARRAY、UUID、TSQUERY、TSVECTOR、TXID_SNAPSHOT类型的数据同步。

- RDS PostgreSQL的JSON类型字段可以同步为AnalyticDB for PostgreSQL的VARCHAR类型。
- 仅支持INSERT、UPDATE、DELETE语句的数据同步；不支持DDL及其它DML 语句同步。
- 同步过程中，若源库有DDL操作，需要手工在目标库中进行对应的DDL操作，然后重启DTS任务。

迁移前准备工作

- 在同步启动前，需要在AnalyticDB for PostgreSQL中创建好对应的schema和table。

  
  说明:
  暂不支持表结构的同步。

- 源RDS PostgreSQL需要将`wal_level`参数值改为`logical`，操作步骤如下。

  1. 登录RDS管理控制台。
  2. 单击源实例ID或操作栏中的管理按钮，进入基本信息页面。
  3. 在左侧导航栏，单击参数设置。
  4. 在参数设置页面找到`wal_level`参数，将参数值改为`logical`。

     
     说明:
     修改`wal_level`参数后需要重启实例才能生效，请评估对业务的影响，在合适的时间进行修改。

操作步骤

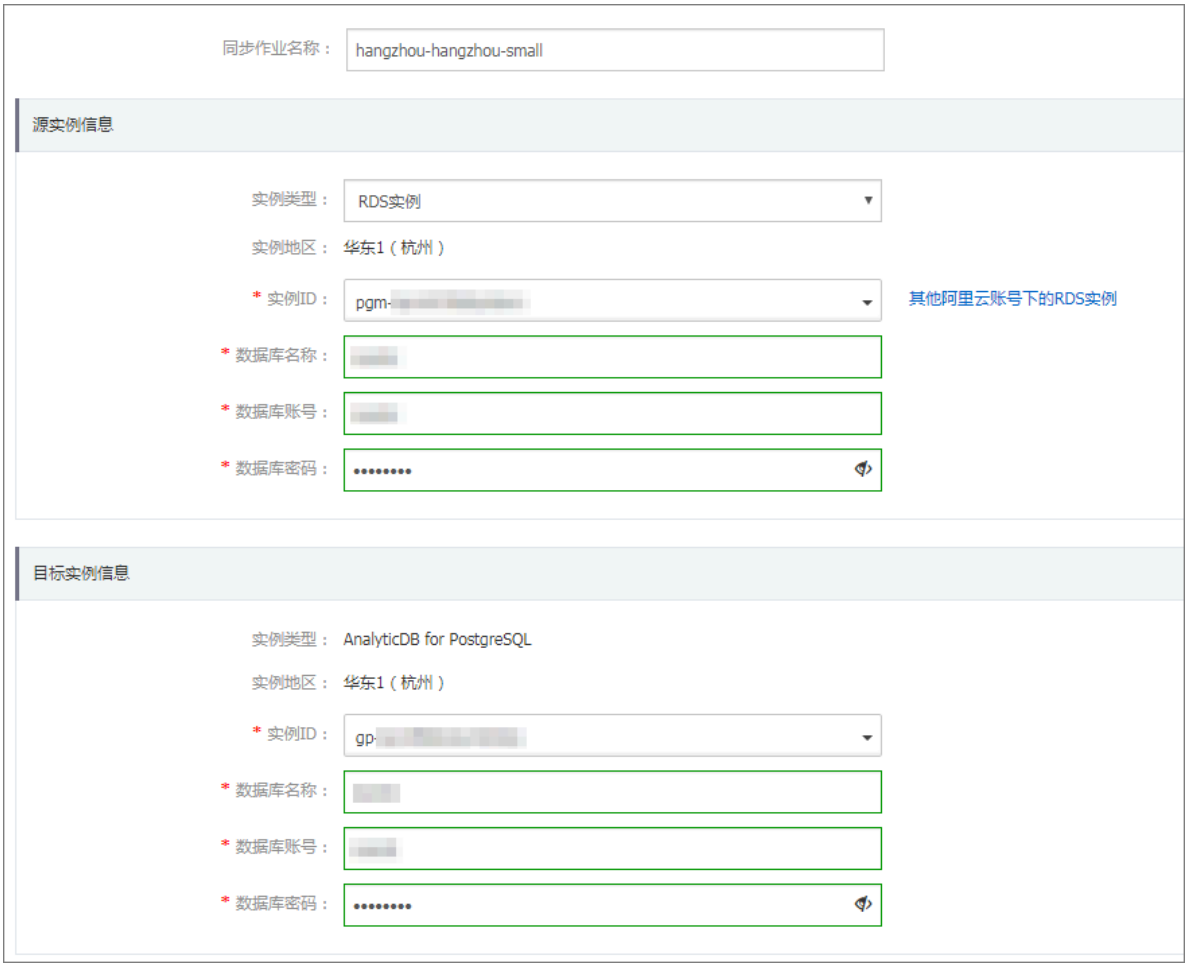
1. 登录数据传输服务DTS控制台。
2. 在左侧导航栏，单击数据同步。
3. 在页面右上角，单击创建同步作业。
4. 在数据传输服务购买页面，选择付费类型为预付费或按量付费。
   - 预付费：属于预付费，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。
   - 按量付费：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。
5. 选择数据同步实例的参数配置信息，参数说明如下表所示。

配置项目	配置选项	配置说明
基本配置	功能	选择数据同步。
	源实例	选择PostgreSQL。

配置项目	配置选项	配置说明
	源实例地域	选择数据同步链路中源RDS实例的地域。  说明: 订购后不支持更换地域，请谨慎选择。
	目标实例	选择AnalyticDB for PostgreSQL。
	目标实例地域	选择数据同步链路中目标AnalyticDB for PostgreSQL实例的地域。  说明: 订购后不支持更换地域，请谨慎选择。
	同步拓扑	数据同步支持的拓扑类型，选择单向同步。  说明: 目前仅支持单向同步。
	同步链路规格	数据传输为您提供了不同性能的链路规格，以同步的记录数为衡量标准。详情请参见数据同步规格说明。  说明: 建议生产环境选择small及以上规格。
	网络类型	数据同步服务使用的网络类型，目前固定为专线。
购买量	购买数量	一次性购买数据同步实例的数量，默认为1，如果购买的是按量付费实例，一次最多购买9 条链路。

6. 单击立即购买，根据提示完成支付流程。
7. 购买成功后单击管理控制台。
8. 在左侧导航栏，单击数据同步。
9. 定位至已购买的数据同步实例，单击配置同步链路。

10.配置同步通道的源实例及目标实例信息。

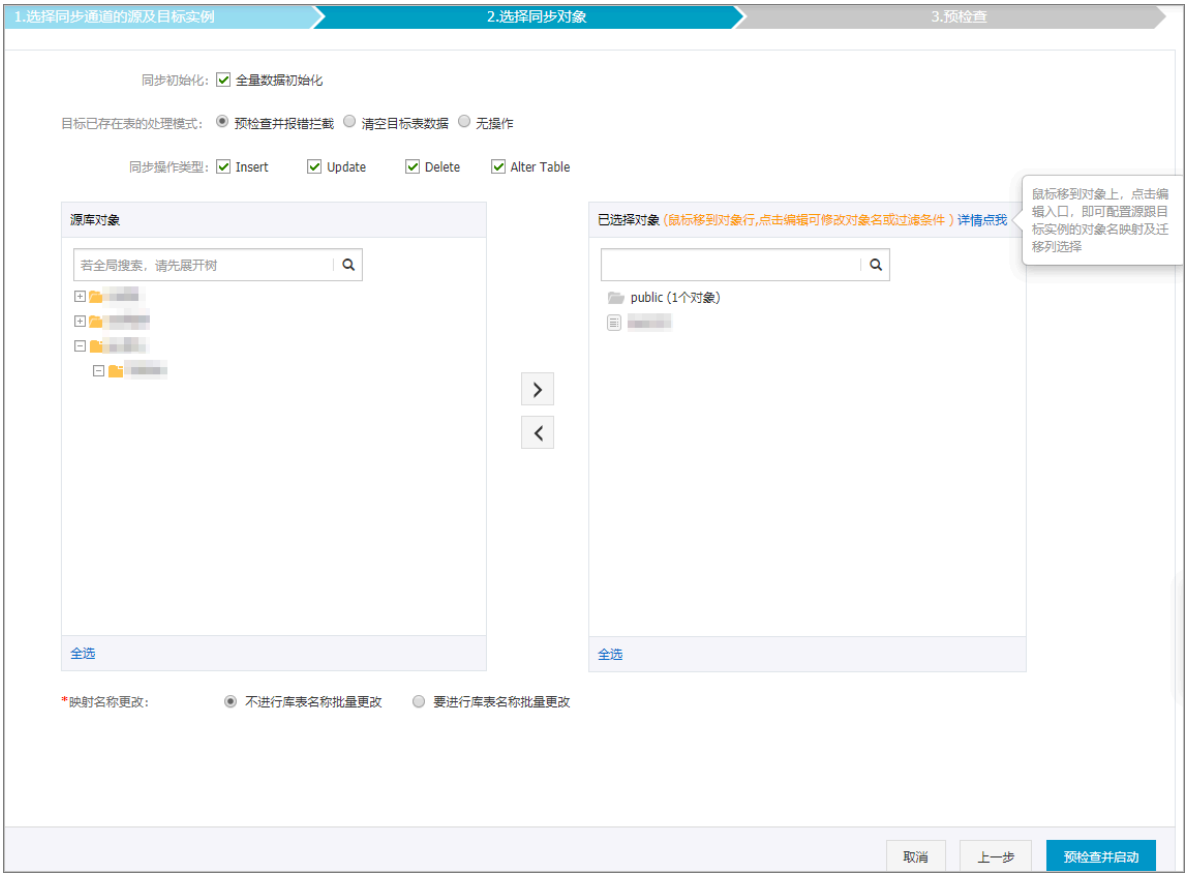


配置项目	配置选项	配置说明
同步作业名称	-	· DTS为每个任务自动生成一个任务名称，任务名称没有唯一性要求。 · 您可以根据需要修改任务名称，建议为任务配置具有业务意义的名称，便于后续的任务识别。
源实例信息	实例类型	选择RDS实例。
	实例地区	购买数据同步实例时选择的源实例地域信息，不可变更。
	实例ID	选择作为数据同步源的RDS实例ID。
	数据库名称	填入同步源表所属的数据库名称。
	数据库账号	填入源RDS PostgreSQL实例的数据库账号。 说明: 数据库账号须具备schema的owner权限。
	数据库密码	填入数据库账号对应的密码。

配置项目	配置选项	配置说明
目标实例信息	实例类型	固定为AnalyticDB for PostgreSQL，无需设置。
	实例地区	购买数据同步实例时选择的目标实例地域信息，不可变更。
	实例ID	选择作为数据同步目标的AnalyticDB for PostgreSQL实例ID。
	数据库名称	填入同步目标表所属的数据库名称。
	数据库账号	填入目标AnalyticDB for PostgreSQL实例的数据库账号。 说明: 数据库账号须具备SELECT、INSERT、UPDATE、DELETE、COPY、TRUNCATE TABLE权限。
	数据库密码	填入数据库账号对应的密码。

11.单击页面右下角的授权白名单并进入下一步。

12.配置同步策略及对象信息。



配置项目	配置参数	配置说明
同步策略配置	同步初始化	选择全量数据初始化。 **说明:** 将源实例中已经存在同步对象的数据在目标实例中初始化，作为后续增量同步数据的基线数据。
	目标已存在表的处理模式	· 预检查并报错拦截：如果发现目标表中已有数据，报错停止。 · 清空目标表数据：如果发现目标表中已有数据，清空表数据。 · 无操作：无论目标表是否有数据，不做任何操作。

配置项目	配置参数	配置说明
	同步操作类型	· Insert · Update · Delete · AlterTable 说明: · RDS PostgreSQL迁移到AnalyticDB for PostgreSQL不支持AlterTable同步操作。 · 根据业务需求选择数据同步的操作类型。
选择同步对象	-	同步对象的选择粒度为表。 如果需要目标表中列信息与源表不同，则需要使用DTS的字段映射功能，详情请参见库表列映射。 说明: 不支持CREATE TABLE操作，您需要通过修改同步对象操作来新增对应表的同步，详情请参见新增同步对象。

13.上述配置完成后单击页面右下角的预检查并启动。

说明:

· 在数据同步任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动数据同步任务。

· 如果预检查失败，请查看具体的失败详情。根据失败原因修复后，重新进行预检查。

14.在预检查对话框中显示预检查通过后，关闭预检查对话框，该同步作业的同步任务正式开始。

15.等待该同步作业的链路初始化完成，直至状态处于同步中。

您可以在 数据同步页面，查看数据同步状态。

7 高级扩展插件（EXTENSION）

7.1 扩展插件(EXTENSION)使用

云数据库 AnalyticDB for PostgreSQL 基于 Greenplum Database 开源数据库项目开发，由阿里云深度扩展，是一种在线的分布式云数据库，由多个计算组组成，可提供大规模并行处理（MPP）数据仓库的服务。

插件类型

云数据库 AnalyticDB for PostgreSQL 支持如下插件(EXTENSION)：

- PostGIS：支持地理信息数据。
- MADlib：机器学习方面的函数库。
- fuzzystrmatch：字符串模糊匹配。
- orafunc：兼容 Oracle 的部分函数。
- oss_ext：支持从 OSS 读取数据。
- hll：支持用 HyperLogLog 算法进行统计。
- pljava：支持使用 PL/Java 语言编写用户自定义函数（UDF）。
- pgcrypto：支持加密函数。
- intarray：整数数组相关的函数、操作符和索引支持。
- roaringbitmap：采用Roaring Bitmap高效压缩算法的位图运算插件。

创建插件

创建插件的方法如下所示：

```
CREATE EXTENSION <extension name>;
CREATE SCHEMA <schema name>;
CREATE EXTENSION IF NOT EXISTS <extension name> WITH SCHEMA <schema
name>;
```

 注意：

创建 MADlib 插件时，需要先创建 plpythonu 插件，如下所示：

```
CREATE EXTENSION plpythonu;
CREATE EXTENSION madlib;
```

删除插件

删除插件的方法如下所示：

注意:

如果插件被其它对象依赖，需要加入 CASCADE（级联）关键字，删除所有依赖对象。

```
DROP EXTENSION <extension name>;
DROP EXTENSION IF EXISTS <extension name> CASCADE;
```

7.2 HyperLogLog 的使用

阿里云深度优化云数据库 AnalyticDB for PostgreSQL，除原生 Greenplum Database 功能外，还支持 HyperLogLog，为互联网广告分析及有类似预估分析计算需求的行业提供解决方案，以便于快速预估 PV、UV 等业务指标。

创建 HyperLogLog 插件

执行如下命令，创建 HyperLogLog 插件：

```
CREATE EXTENSION hll;
```

基本类型

- **执行如下命令，创建一个含有 hll 字段的表：**

```
create table agg (id int primary key,userids hll);
```

- **执行如下命令，进行 int 转 hll_hashval：**

```
select 1::hll_hashval;
```

基本操作符

- **hll 类型支持 =、!=、<>、|| 和 #。**

```
select hll_add_agg(1::hll_hashval) = hll_add_agg(2::hll_hashval);
select hll_add_agg(1::hll_hashval) || hll_add_agg(2::hll_hashval);
select #hll_add_agg(1::hll_hashval);
```

- **hll_hashval 类型支持 =、!= 和 <>。**

```
select 1::hll_hashval = 2::hll_hashval;
select 1::hll_hashval <> 2::hll_hashval;
```

基本函数

- **hll_hash_boolean、hll_hash_smallint 和 hll_hash_bigint 等 hash 函数。**

```
select hll_hash_boolean(true);
```

```
select hll_hash_integer(1);
```

- hll_add_agg：可以将 int 转 hll 格式。

```
select hll_add_agg(1::hll_hashval);
```

- hll_union：hll 并集。

```
select hll_union(hll_add_agg(1::hll_hashval),hll_add_agg(2::
hll_hashval));
```

- hll_set_defaults：设置精度。

```
select hll_set_defaults(15,5,-1,1);
```

- hll_print：用于 debug 信息。

```
select hll_print(hll_add_agg(1::hll_hashval));
```

示例

```
create table access_date (acc_date date unique, userids hll);
insert into access_date select current_date, hll_add_agg(hll_hash_i
nteger(user_id)) from generate_series(1,10000) t(user_id);
insert into access_date select current_date-1, hll_add_agg(hll_hash_i
nteger(user_id)) from generate_series(5000,20000) t(user_id);
insert into access_date select current_date-2, hll_add_agg(hll_hash_i
nteger(user_id)) from generate_series(9000,40000) t(user_id);
postgres=# select #userids from access_date where acc_date=current_da
te;
     ?column?
------------------
 9725.85273370708
(1 row)
postgres=# select #userids from access_date where acc_date=current_da
te-1;
     ?column?
------------------
 14968.6596883279
(1 row)
postgres=# select #userids from access_date where acc_date=current_da
te-2;
     ?column?
------------------
 29361.5209149911
(1 row)
```

7.3 使用压缩位图索引（Roaring Bitmap）

位图（bitmap）是一种常用的数据结构，位图为每个列所有可能的值创建一个单独的位图（0和1的系列），每个位（bit）对应一个数据行是否存在对应的值。通过位图能够快速定位一个数值是否在存在，并利用计算机位级计算的快速特性（AND，OR和ANDNOT等运算），可以显著加快位图的相关计算，非常适合大数据查询和关联计算，常用于去重、标签筛选、时间序列等计算中。

传统的位图会占用大量内存，一般需要对位图进行压缩处理。Roaring Bitmap是一种高效优秀的位图压缩算法，目前已被广泛应用在各种语言和各种大数据平台上。

Roaring Bitmap压缩算法简介

Roaring Bitmap的算法是将整数的32-bit的范围 ([0, n]) 划分为 2^16 个数据块（Chunk），每一个数据块对应整数的高16位，并使用一个容器（Container）来存放一个数值的低16位。Roaring Bitmap将这些容器保存在一个动态数组中，作为一级索引。容器使用两种不同的结构：数组容器（Array Container）和 位图容器（Bitmap Container）。数组容器存放稀疏的数据，位图容器存放稠密的数据。如果一个容器里面的整数数量小于4096，就用数组容器来存储值。若大于4096，就用位图容器来存储值。

采用这种存储结构，Roaring Bitmap可以快速检索一个特定的值。在做位图计算（AND，OR，XOR）时，Roaring Bitmap提供了相应的算法来高效地实现在两种容器之间的运算。使得Roaring Bitmap无论在存储和计算性能上都变现优秀。

更多关于Roaring Bitmap的介绍信息，请参考官方网站

使用Roaring Bitmap位图计算

使用Roaring Bitmap位图计算功能之前，首先需要在数据库中创建RoaringBitmap扩展插件：

```
CREATE EXTENSION roaringbitmap;
```

创建带有RoaringBitmap数据类型的表：

```
CREATE TABLE t1 (id integer, bitmap roaringbitmap);
```

使用rb_build函数插入roaringbitmap的数据：

```
--数组位置对应的BIT值为1
INSERT INTO t1 SELECT 1,RB_BUILD(ARRAY[1,2,3,4,5,6,7,8,9,200]);
--将输入的多条记录的值对应位置的BIT值设置为1，最后聚合为一个roaringbitmap
INSERT INTO t1 SELECT 2,RB_BUILD_AGG(e) FROM GENERATE_SERIES(1,100) e;
```

Bitmap计算(OR, AND, XOR, ANDNOT)

```
SELECT RB_OR(a.bitmap,b.bitmap) FROM (SELECT bitmap FROM t1 WHERE id
 = 1) AS a,(SELECT bitmap FROM t1 WHERE id = 2) AS b;
```

Bitmap聚合计算(OR, AND, XOR, BUILD)，并生成新的roaringbitmap类型

```
SELECT RB_OR_AGG(bitmap) FROM t1;
SELECT RB_AND_AGG(bitmap) FROM t1;
SELECT RB_XOR_AGG(bitmap) FROM t1;
```

```
SELECT RB_BUILD_AGG(e) FROM GENERATE_SERIES(1,100) e;
```

统计基数（Cardinality）,即统计roaringbitmap中包含多少个位置为1的BIT位

```
SELECT RB_CARDINALITY(bitmap) FROM t1;
```

从roaringbitmap中返回位置为1的BIT下标（位置值）。

```
SELECT RB_ITERATE(bitmap) FROM t1 WHERE id = 1;
```

Bitmap函数列表

函数名	输入	输出	描述	示例
rb_build	integer[]	roaringbitmap	通过数组创建一个Bitmap。	`rb_build('{1,2,3,4,5}')`
rb_and	roaringbitmap，roaringbitmap	roaringbitmap	And计算。	`rb_and(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_or	roaringbitmap，roaringbitmap	roaringbitmap	Or计算。	`rb_or(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_xor	roaringbitmap，roaringbitmap	roaringbitmap	Xor计算。	`rb_xor(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_andnot	roaringbitmap，roaringbitmap	roaringbitmap	AndNot计算。	`rb_andnot(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_cardinality	roaringbitmap	integer	统计基数	`rb_cardinality(rb_build('{1,2,3,4,5}'))`
rb_and_cardinality	roaringbitmap，roaringbitmap	integer	And计算并返回基数。	`rb_and_cardinality(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_or_card	roaringbitmap	integer	Or计算并返回基	`rb_or_card`

8 PL/Java的使用

8.1 使用 PL ／ Java UDF

AnalyticDB for PostgreSQL 支持用户使用 PL/Java 语言，编写并上传 jar 软件包，并利用这些 jar 包创建用户自定义函数（UDF）。该功能支持的 PL/Java 语言版本为社区版 PL/Java 1.5.0，使用的 JVM 版本为 1.8。

本文介绍了创建 PL/Java UDF 的示例步骤。更多的 PL/Java 样例，请参见 PL/Java代码（查看 编译方法）。

操作步骤

1. 在 AnalyticDB for PostgreSQL 中，执行如下命令，创建 PL/Java 插件（每个数据库只需执行一次）。

```
create extension pljava;
```

2. 根据业务需要，编写自定义函数。例如，使用如下代码编写 `Test.java` 文件：

```
public class Test
{
    public static String substring(String text, int beginIndex,
                int endIndex)
            {
                    return text.substring(beginIndex, endIndex);
            }
}
```

3. 编写 `manifest.txt` 文件。

```
Manifest-Version: 1.0
Main-Class: Test
Specification-Title: "Test"
Specification-Version: "1.0"
Created-By: 1.7.0_99
Build-Date: 01/20/2016 21:00 AM
```

4. 执行如下命令，将程序编译打包。

```
javac Test.java
```

```
jar cfm analytics.jar manifest.txt Test.class
```

5. 将步骤 4 生成的 `analytics.jar` 文件，通过 OSS 控制台命令上传到 OSS。

```
osscmd put analytics.jar oss://zzz
```

6. 在 AnalyticDB for PostgreSQL 中，执行 Create Library 命令，将文件导入到 AnalyticDB for PostgreSQL 中。

> **注意:**
> Create Library 只支持 filepath，一次导入一个文件。另外，Create Library 还支持字节流形式，可以不通过 OSS 直接导入，详情请参见Create Library命令的使用 。

```
create library example language java from 'oss://oss-cn-hangzhou.
aliyuncs.com filepath=analytics.jar id=xxx key=yyy bucket=zzz';
```

7. 在 AnalyticDB for PostgreSQL 中，执行如下命令，创建和使用相应 UDF。

```
 create table temp (a varchar) distributed randomly;
 insert into temp values ('my string');
 create or replace function java_substring(varchar, int, int)
returns varchar as 'Test.substring' language java;
 select java_substring(a, 1, 5) from temp;
```

8.2 使用 Create Library 命令

为支持用户导入自定义软件包，AnalyticDB for PostgreSQL 引入了 Create Library/Drop Library 命令。使用此命令创建 PL/Java 的 UDF 的示例，请参见 PL/Java UDF的使用。

本文介绍了 Create/Drop Library 命令的使用方法。

语法

```
CREATE LIBRARY library_name LANGUAGE [JAVA] FROM oss_location OWNER
ownername
CREATE LIBRARY library_name LANGUAGE [JAVA] VALUES file_content_hex
OWNER ownername
DROP LIBRARY library_name
```

参数说明:

- `library_name`: 要安装的库的名称。若已安装的库与要安装的库的名称相同，必须先删除现有的库，然后再安装新库。
- `LANGUAGE [JAVA]`: 要使用的语言。目前仅支持 PL/Java。

- `oss_location`：包文件的位置。您可以指定 OSS 存储桶和对象名称，仅可以指定一个文件，且不能为压缩文件。其格式为：

  ```
  oss://oss_endpoint filepath=[folder/[folder/]...]/file_name id=
  userossid key=userosskey bucket=ossbucket
  ```

- `file_content_hex`：文件内容，字节流为 16 进制。例如，`73656c6563742031`表示”select 1”的 16 进制字节流。借助这个语法，可以直接导入包文件，不必通过 OSS。
- `ownername`：指定用户。
- `DROP LIBRARY`：删除一个库。

示例

- 示例 1：安装名为 `analytics.jar` 的 jar 包。

  ```
  create library example language java from 'oss://oss-cn-hangzhou.
  aliyuncs.com filepath=analytics.jar id=xxx key=yyy bucket=zzz';
  ```

- 示例 2：直接导入文件内容，字节流为 16 进制。

  ```
  create library  pglib LANGUAGE java VALUES '73656c6563742031' OWNER
   "myuser";
  ```

- 示例 3：删除一个库。

  ```
  drop library example;
  ```

- 示例 4：查看已经安装的库。

  ```
  select name, lanname from pg_library;
  ```

9 非结构化数据向量分析

9.1 向量分析概述

功能概述

分析型数据库PostgreSQL版的向量分析旨在帮助您实现非结构化数据的近似检索和分析，其实现原理是通过AI算法提取非结构化数据的特征，然后利用特征向量唯一标识非结构化数据，向量间的距离用于衡量非结构化数据之间的相似度。分析型数据库PostgreSQL版向量检索分析基于MPP查询架构构建，帮助用户实现基于SQL接口进行非结构化数据检索，并支持同结构化数据的关联分析。

典型应用场景

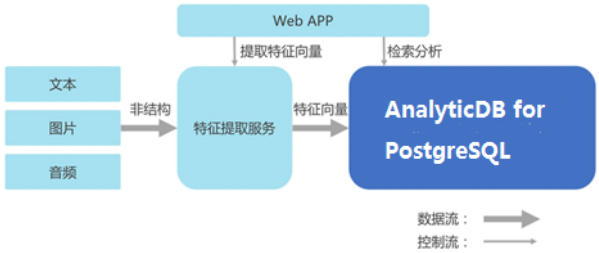
通过分析型数据库PostgreSQL版向量分析，您可以非常容易地搭建各种智能化应用，例如以下几种应用。

- 以图搜图，即通过图片检索图片。
- 声纹匹配，通过音频检索音频。
- 基于语义的文本检索和推荐，通过文本检索近似文本。
- 文件去重，通过文件指纹去除重复文件。
- 商品图片分析，在大量图片中分析哪些图片包含了同一个商品。

向量分析作为分析型数据库PostgreSQL版向量分析的高级特性目前已经服务阿里内外部多项业务，包括阿里巴巴数据中台，阿里电商新零售业务，阿里云城市大脑。

典型架构

图 9-1: 基于AnalyticDB for PG实现非结构化数据向量分析示例



- Web App把文本，图片或者视频等非结构化数据（后续简称非结构化数据）通过特征提取服务提取特征向量，然后再把特征向量写入分析型数据库PostgreSQL版向量分析的向量表。
- Web App检索的时候首先把非结构化数据通过特征提取服务接口提取出向量，然后调用分析型数据库PostgreSQL版向量分析的查询分析接口做查询。

9.2 向量分析的使用

分析型数据库PostgreSQL版的向量分析特性针对非结构化数据检索分析，具备丰富功能和优异性能，与普通的检索系统有较大的差异，主要体现在下面的几点：

- 结构化和非结构化混合分析

  例如，可以检索与输入图片中的连衣裙似度最高、价格在100元到200元之间且上架时间在最近1个月以内的产品。

- 支持数据实时更新

  传统的向量分析系统中数据只能按照T+1更新，不支持数据实时写入。分析型数据库PostgreSQL版向量分析支持数据实时更新和查询。

- 支持向量分析碰撞

  分析型数据库PostgreSQL版向量分析支持KNN-Join，即比较一堆向量与另外一堆向量的相似度，类似于spark中的KNN-Join操作，这种场景计算量巨大，分析型数据库PostgreSQL版针对该场景做了大量优化。

  典型的应用场景有商品去重，计算新加入的商品与历史商品库中有哪些是相似的。人脸聚类，计算一段时间内的人脸库中，哪些人脸是同一个人。

- 易用性

  分析型数据库PostgreSQL版向量分析申请即可使用，支持标准SQL，简化开发流程。同时，分析型数据库PostgreSQL版向量分析内置常用特征提取和属性提取，也支持集成第三方特征提取服务。

结构化数据和非结构化关联分析

以检索与输入图片中的连衣裙相似度最高、价格在100元到200元之间且上架时间在最近1个月以内的产品为例，有下列商品库，其中商品表products的字段设计如下：

字段	类型	说明
Id	Char(64)	商品
Name	Varchar(256)	商品名称
Price	real	价格

字段	类型	说明
InTime	timestamp	入库时间
Url	Varchar(256)	图片链接
Feature	real(512)	图片特征

检索的SQL如下：

```
Select id, price from products where price > 100 and price <=200 and
InTime > ‘2019-03-01 00: 00: 00’ and InTime <= ‘2019-03-31 00: 00: 00’
order by l2_distance(array[10,2.0,…, 512.0], feature) desc limit 100;
```

其中`l2_distance`为向量检索定义的距离函数，它包含下列2个参数：

参数	举例	说明
Feature1	`array[10,2.0,…, 512.0]`	需要查询的特征向量
Feature2	`Feature`	底库中的向量列

分析型数据库PostgreSQL版向量分析支持“欧氏距离“、“点积距离” 及 “汉明距离” 三种距离函数，分别适用于不同的情景。

非结构化数据写入

分析型数据库PostgreSQL版向量分析支持数据实时写入，立等可查。数据的写入方式与传统数据库一样，使用insert语句插入向量数据。

向量检索与分析

以上述的商品库为例，我们实现商品去重的逻辑，计算最近1天加入的商品与上个月的商品库里面有哪些是相似的，它的SQL如下：

```
select A.id as ida, B.id as idb from products A join products B on
dp_distance(A.feature, B.feature) > 0.9 where A.inTime > '2019-03-31
 00: 00: 00' and B.InTime >= '2019-02-01 00: 00: 00' and  B.InTime < '
2019-02-29 00: 00: 00';
```

上述SQL把'2019-03-31 00：00：00'后写入的数据与2月份的数据做笛卡尔积，把向量点积距离大于0.9的商品对应的id提取出来。基于分析型数据库PostgreSQL版强大的优化器，用户无需写子查询，直接使用描述性的SQL即可。

易用性

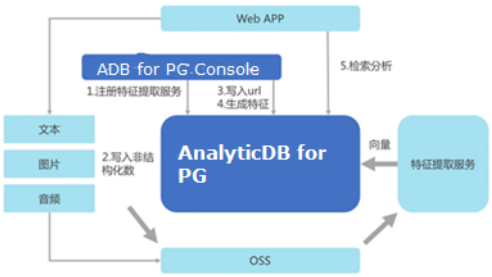
向量分析完整继承了分析型数据库PostgreSQL版向量分析的所有商业工具和生态，并支持常用的特征提取模型，还支持与第三方特征提取服务集成。

· 申请即可使用：开通分析型数据库PostgreSQL版服务即可使用向量分析。

- 分析型数据库PostgreSQL版全面兼容PostgreSQL协议和SQL2003，降低开发成本。
- 内置常用特征提取模块，支持集成第三方特征提取服务。

为了让您对非结构化数据拥有更多的自主控制权，您可以把非结构化数据保存在OSS或者图片服务器上（下图使用OSS），非结构化数据的保存地址即URL存储在分析型数据库PostgreSQL版中，整体架构如下所示。

图 9-2: 特征提取集成



1. 通过分析型数据库PostgreSQL版控制台注册特征提取服务。
2. 非结构化数据保存到OSS，同时返回访问的URL。
3. 非结构化数据的存储地址即URL保存在分析型数据库PostgreSQL版中。
4. 通过Web App调用分析型数据库PostgreSQL版的自定义函数生成向量特征，分析型数据库PostgreSQL版后台通过调用特征提取服务从OSS读取非结构化数据，提取特征，并把特征向量保存在分析型数据库PostgreSQL版中。所有这些操作只需要一条SQL便可轻松完成，SQL语句示例如下。

```
select feature_extractor('clothes','https://xxx/1036684144_687583347
.jpg');

insert into product(id, url, feature) values(0, 'http://xxx/
1036684144_687583347.jpg', feature_extractor('clothes','https://xxx/
1036684144_687583347.jpg'));
```

`feature_extractor`为商品特征提取的自定义函数，传入商品图片URL，提取商品特征向量，该向量可以用来做商品检索和属性提取。

> 说明:
> `feature_extractor`第一个参数区别要提取的属性类型，仅支持FACE、CLOTHES、TEXT。

对于常用的人脸特征提取、文本特征提取BERT模型以及服装特征提取也已经内置于分析型数据库PostgreSQL版服务中，您也可以使用您自己的特征提取服务。

9.3 案例：人脸检索

系统需求

某人脸检索系统，系统有3个方面的要求：

1. 实现黑名单监控告警

黑名单中保存的是在逃人员的人脸特征，摄像头提取的人脸都与黑名单库中的人脸做对比。

2. 同人识别

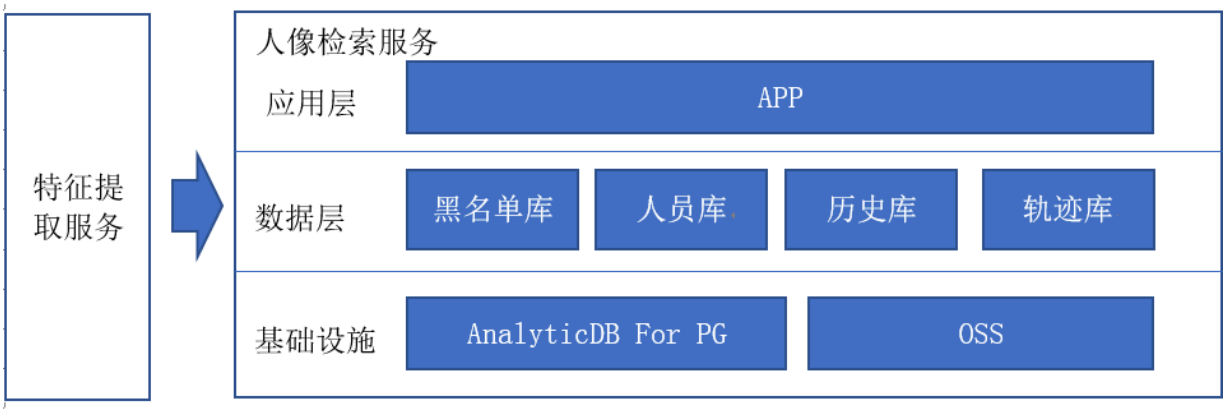
摄像头提取的人脸都会与证照库中的人脸特征做对比，通过人脸找到关联的证件ID，通过这种方式把人脸转出结构化数据。

3. 历史库人像检索

对于在证照库中找不到的人脸，会保存在历史人像库中，并保存1个月，以备后续案件侦破的时候查询轨迹使用。

系统架构

人像管理部分的系统架构如下：



- **应用层**

  实现用户界面和下层调用逻辑

- 数据层

  对数据做了抽象，对人像抽象出四个库。

  1. 黑名单库

     向量库，该库有100w数据。每个摄像头拍到的人像需要首先到这里做对比，1000路摄像头，评价每秒产生1张人脸照片，要求1000 QPS，为了减少误报，系统对准确率要求较高，可以使用无损检索模式。

  2. 人员库

     向量库，保存了2亿人脸特征数据，会有少量更新，摄像头的人脸都会到这个库做检索，对QPS要求极高，每个摄像头每秒检索1次，要求1000 QPS。

  3. 历史库

     向量库，保存1个月的人脸向量数据，每天1000路摄像头会产生4.3亿人脸数据，1个月产生129亿向量，人脸检索的时候，首先通过人员库查找人员ID，如果检索到人员ID则根据人员ID到轨迹库检索轨迹，否则到历史库中通过人脸特征检索人脸轨迹，所以该库QPS要求较低，100 QPS即可。

  4. 轨迹库

     同人识别是为了实现摄像头每张人脸都能映射到人员ID，表中保存，摄像头识别的每张人脸都会去证照表中检索，如果检索到则会把人脸对应的id和摄像头位置等信息写入轨迹库（结构化数据表）。

- 基础设施

  分析型数据库PostgreSQL版实现结构化和向量数据存储和计算，OSS实现非结构化数据存储。

9.4 案例：商品属性提取和多模搜索

系统需求

电商卖家在准备新商品资料时，需要拍摄商品照片、标注商品类别和属性。以女装为例，需要设置子类目是裙子或者衬衣，衬衣需要标注是圆领、V领、长袖、短袖、风格等多种属性。随着商品类别的增长，将积累大量的图片素材。如果这些图片素材没有被很好地管理和利用，则经常会出现找不到之前已经准备好的素材，需要重新拍摄的情况。

分析型数据库PostgreSQL版可以帮助电商卖家管理和检索图片素材，可以根据图片类别、属性或者相似图片做多模检索。例如，检索与输入图片最相似且价格在200元～300元之间的所有商品的图片。

实现架构

分析型数据库PostgreSQL版作为商品属性提取和图片资料管理的核心组件，数据读写流程如下所示。

插入数据

应用端通过以下步骤向商品库中插入数据。

1. 应用端调OSS服务，将图片插入OSS，获得对应的URL。

   > 说明:
   > 当前只支持http和https协议的URL。

2. 应用端调用特征提取服务，获得图片抽象后的特征向量。
3. 应用端调用分析型数据库PostgreSQL版服务将步骤一中的URL和步骤二中的特征向量一起插入商品库。

查询数据

应用端可以采用上述架构图中的任意一种方案从商品库中查询数据，查询步骤如下所示。

1. 应用端调用OSS服务，将要查询的图片插入OSS，获得对应URL。

   > 说明:
   > 当前只支持http和https协议的URL。

2. 应用端调用特征提取服务，获得URL对应图片抽象后的特征向量，同时也支持直接传入图片获取特征向量。
3. 应用端调用分析型数据库PostgreSQL版，获得相似特征向量图片对应的URL列表。
4. 应用端根据URL列表，调OSS服务，获取并返回图片。

示例

创建商品表

```
CREATE TABLE products (
  id bigint,
  image_url varchar,
  properties varchar,
  feature real[],
  PRIMARY KEY (id)
) DISTRIBUTED BY (id);
```

插入数据

先调用用户自定义函数获取特征向量，然后插入数据。

```
Select feature_extractor('clothes','https://xxx.com/img.jpg');
```

 说明:

可以先通过SELECT确认特征向量是否正确。

```
insert into products (id, image_url, feature) values (10, 'https://xxx
/img.jpg', feature_extractor('clothes','https://xxx/img.jpg'));
```

查询数据

支持SELECT中直接带UDF传入图片url查询, 例如检索与图片链接为'https://xxx/img.jpg'最相似的前10条记录的商品id和URL：

```
Select id, image_url from products order by l2_distance(feature_ex
tractor('clothes','https://xxx/img.jpg'), feature) desc limit 10;
```

9.5 案例：个性化推荐系统

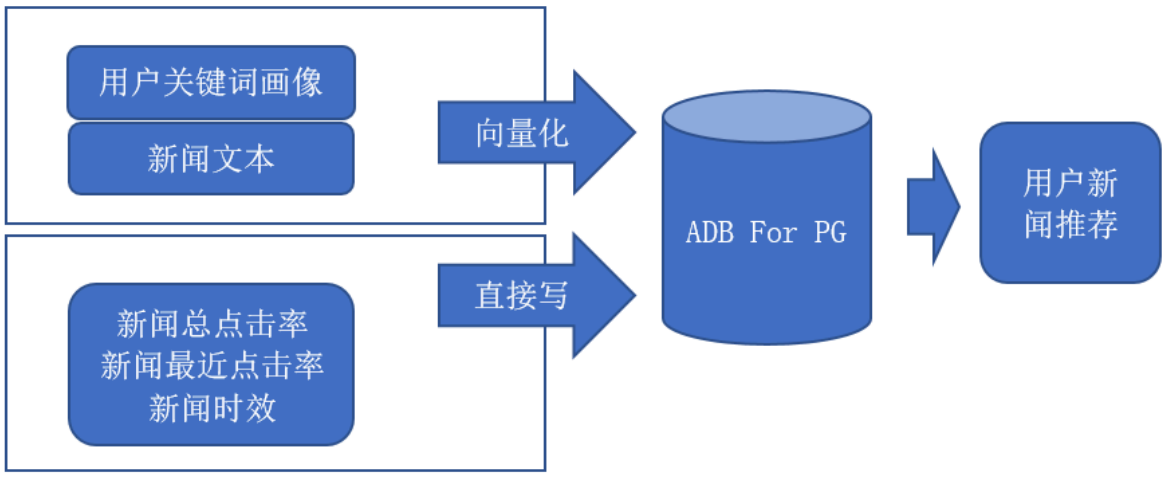
背景信息

互联网时代个性化推荐已经渗透到人们生活的方方面面，例如常见的“猜你喜欢”、“相关商品”等。互联网能够对用户投其所好，向用户推荐他们最感兴趣的内容，实时精准地把握用户兴趣。目前很多成功的手机APP都引入了个性化推荐算法，例如，新闻类的有今日头条新闻客户端、网易新闻客户端、阿里UC新闻客户端等；电商类的有拼多多、淘宝、天猫等。分析型数据库PostgreSQL版推出的向量分析可以帮助您实现上述个性化推荐系统。

个性化推荐系统概述

以个性化新闻推荐系统为例，一篇新闻包含新闻标题、正文等内容，可以先通过NLP（Neuro-Linguistic Programming，自然语言处理）算法，从新闻标题和新闻正文中提取关键词。然后，利用分析型数据库PostgreSQL版向量内置的文本转换为向量函数，将从新闻标题和新闻正文

中提取出的关键词转换为新闻向量导入分析型数据库PostgreSQL版向量数据库中，用于用户新闻推荐，具体实现流程如下图所示。



整个新闻推荐系统由以下步骤实现：

1. 构建分析型数据库PostgreSQL版向量库，得到用户特征向量。

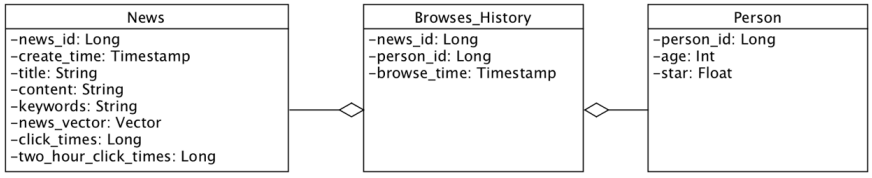
   通过分析用户历史浏览数据，构建相应的用户画像，建立用户偏好模型，得到用户特征向量。新闻推荐系统可以从用户的浏览日志中得到用户历史浏览新闻详情，再从每条历史浏览新闻中提取关键词，建立用户画像。例如，某用户浏览了多条NBA（National Basketball Association，美国职业篮球联赛）季后赛新闻，这些新闻中包含了NBA、篮球、球星、体育等关键词，通过这些关键词可以得出该用户是一个NBA球迷。通过分析型数据库PostgreSQL版向量将这些文本关键词转换为向量并导入到分析型数据库PostgreSQL版向量库中，得到用户特征向量。
2. 根据分析型数据库PostgreSQL版向量数据库和逻辑回归预测模型，将用户感兴趣的新闻推荐给用户。

   通过分析型数据库PostgreSQL版向量数据库，可以从互联网检索出前500条用户没有浏览过的新闻，但是这500条新闻却是该用户最感兴趣的新闻。然后，从这500条新闻中提取每条新闻的创建时间和点击率，根据逻辑回归预测模型（该模型来自于用户以往的浏览的历史记录中），将用户感兴趣的新闻推荐给用户。

分析型数据库PostgreSQL版内置的文本转换为向量函数采用BERT（Bidirectional Encoder Representations from Transformers）模型，同时支持中文和英文两种语言。该模型基于大量的语料进行训练，其中包含了语义信息，而且其查询精度比简单的TF-IDF（term frequency–inverse document frequency）算法高。

个性化推荐系统中数据库表结构设计

图 9-3: 个性化推荐系统分析型数据库PostgreSQL版表结构



上图是个性化新闻推荐系统中分析型数据库PostgreSQL版数据库表结构设计，系统包含了三张表（News, Person，Browses_History）,分别存储新闻信息、用户基本信息、用户浏览记录。

- News表存储新闻信息，包含新闻id（news_id）、新闻创建时间（create_time）、新闻名字（title）、新闻内容（content）、总的用户点击数（click_times）、两个小时内的用户点击次数（two_hour_click_times）。根据新闻的名称和内容得到新闻的关键词keywords，然后将新闻的关键词转化成向量（news_vector）。向news表中插入数据时，系统自动根据关键词转换为向量，将向量和其他新闻信息一起插入news表。

```
CREATE TABLE news (
  news_id bigint,
  create_time timestamp,
  title varchar(100),
  content varchar(200),
  keywords varchar(50),
  click_times bigint,
  two_hour_click_times bigint,
  news_vector real[],
  primary key (news_id)
) distributed by (news_id);
```

- Browses_History表记录用户浏览的新闻的情况，包括新闻id（news_id）、用户id（person_id）、用户浏览新闻的时间（browse_time）。

```
CREATE TABLE browses_history (
  browse_id bigint,
  news_id bigint,
  person_id bigint,
  browse_time timestamp,
  primary key (browse_id)
) distributed by (browse_id);
```

- Person表记录用户信息，包括用户的id（person_id）、用户的年龄（age）、用户的星级（star）。

```
CREATE TABLE person(
  person_id bigint,
  age bigint,
  star float,
  primary key (person_id)
```

```
) distributed by (person_id);
```

个性化推荐系统实施步骤

1. 从新闻中抽取新闻特征向量

   分析型数据库PostgreSQL版通过内置的文本转换为向量函数，抽取新闻特征向量，然后将新闻特征向量存入新闻表news中。例如，执行以下SELECT将返回文本“ADB For PG is very good!”对应的特征向量。

   ```
   select feature_extractor('text', 'ADB For PG is very good!');
   ```

   假设新闻如下图所示，通过以下两个步骤将新闻信息存入新闻表news表中。

   a. 提取新闻关键词。

      由于分析型数据库PostgreSQL版暂时不支持关键词提取函数，您可以调用jieba结巴中文NLP系统）中的关键词抽取函数（jieba.analyse.extract_tags(title + content, 3)）提取关键词。

   b. 执行INSERT将新闻信息（包含关键词和新闻特征向量）存入新闻表news表中。

      ```
      insert into news(news_id, create_time, title, content, keywords,
      click_times,two_hour_click_times)
      values(1, now(),'韩国军方:朝鲜在平安北道一带向东发射不明飞行物', '据韩国联合参谋本部消息，当地时间今天下午16时30分左右，朝鲜在其平安北道一带向东发射不明飞行物。', '韩国 朝鲜 不明飞行物', 123, 3);
      ```

2. 提取用户特征向量

   a. 提取用户浏览关键词。

      根据用户的新闻浏览日志，我们很容易得到用户的浏览关键词。例如，执行以下SELECT得到用户person_id为9527的浏览关键词。

      ```
      select keywords
      from Person p, Browses_History bh, News n
      ```

```
where p.person_id = bh.person_id and bh.news_id = n.news_id and p.
person_id = 9527;
```

将用户浏览关键词全部提取出来之后，就可以得到用户总的浏览关键词 。例如，用户person_id为9527浏览了关键词为“NBA 体育”、“总决赛”、“热火”、“火箭”的新闻。然后通过文本转换为向量函数，将用户person_id为9527浏览的关键词转换成向量。

b. 将用户浏览关键词转换为用户特征向量。

```
select feature_extractor('text', 'NBA 体育 总决赛 热火 火箭'));
```

3. 根据用户特征向量获取新闻推荐结果

通过用户特征向量，到新闻表news中查询相关的新闻信息。例如，执行以下SELECT将返回和用户相关的前500条新闻，同时系统也会过滤掉用户已经阅读过的文章。

```
select news_id, title, content, (extract(epoch from (now()-
create_time)) * w1 + click_times/extract(epoch from (now()-
create_time)) * w2 + two_hour_click_times/extract(epoch from (now()-
create_time)) * w3 + ann_distance * w4) as rank_score
from (select *, l2_distance(news_vector, feature_extractor('textf
', 'NBA 体育 总决赛 热火 火箭')) as ann_distance from news order by
ann_distance desc limit 500) S
order by rank_score desc;
```

参数说明：

- ann_distance：用户与新闻的相关度
- create_time：新闻的创建时间
- click_times/(now()-create_time)：新闻热度点击率
- two_hour_click_times/(now()-create_time)：新闻近期热度点击率
- w1、w2、w3、w4：逻辑回归模型学习中各个属性的权重

获取新闻推荐结果之后，应用就可以将用户感兴趣的新闻推荐给用户了。

10 BI工具

10.1 BI工具兼容概述

AnalyticDB for PostgreSQL 基于开源数据库 Greenplum 构建，兼容 Greenplum 接口及相关工具，兼容业界主流 BI 工具，也兼容阿里云提供的QucikBI 及 DataV 等数据智能和展现工具。针对业界主流工具，用户可以选择以 Greenplum 或 PostgreSQL 作为数据源类型来连接实例。JDBC Driver 可以考虑BI工具自带的 Driver，或采用连接数据库 推荐的 JDBC Driver。

BI工具兼容支持列表

BI 工具	使用指导
阿里云 QuickBI	请参见分析型数据库PostgreSQL版作为数据源对接阿里云 Quick BI，选择AnalyticDB for PostgreSQL（或HybridDB for PostgreSQL）作为数据源。
阿里云 DataV	请参见DataV选择AnalyticDB for PostgreSQL（或HybridDB/HybridDB for PostgreSQL）作为数据源。
Tableau	请参见分析型数据库PostgreSQL版作为数据源对接Tableau BI,选择Greenplum类型数据源连接实例。
帆软	请参见分析型数据库PostgreSQL版作为数据源对接帆软FineBI,选择Greenplum类型数据源连接实例。

10.2 阿里云Quick BI连接分析型数据库PostgreSQL版

本文介绍如何通过阿里云Quick BI连接分析型数据库PostgreSQL版。

操作步骤

1. 登录 Quick BI 控制台。
2. 单击上方菜单栏中的工作空间。
3. 在工作空间页面单击左侧数据源。
4. 单击新建数据源 > AnalyticDB for PostgreSQL。

5. 在添加AnalyticDB for PostgreSQL数据源页面进行参数配置。

> **说明:**
>
> - 请在分析型数据库PostgreSQL版白名单中添加如下IP地址，Quick BI才能访问分析型数据库PostgreSQL版：10.152.69.0/24,10.152.163.0/24,139.224.4.0/24。
> - 关于如何设置白名单，请参见设置白名单。

配置项	说明
显示名称	数据源名称。
数据库地址	分析型数据库PostgreSQL版的连接地址，详情请参见连接地址。
端口	连接地址对应的端口号。
数据库	分析型数据库PostgreSQL版数据库名。
Schema	数据库Schema名。
用户名	分析型数据库PostgreSQL版数据库账号。
密码	分析型数据库PostgreSQL版数据库密码。

6. 完成上述参数配置后，单击连接测试测试连通性，测试通过后，单击添加添加数据源。

> 说明:
> 如果连通失败，请检查各配置项是否填写正确，确认无误后再进行连接测试。

使用 Quick BI

成功连接分析型数据库PostgreSQL版数据源后，您可以参考以下步骤在Quick BI中完成报表分析等操作。

- 创建数据集
- 制作仪表板
- 制作数据门户

10.3 Tableau连接分析型数据库PostgreSQL版

Tableau是一款数据分析与可视化工具，支持连接本地或云端数据，无论是电子表格还是数据库数据，都可以无缝连接。本文介绍使用Tableau连接分析型数据库PostgreSQL版。

连接Tableau

1. 启动Tableau。
2. 在连接页面选择Pivotal Greenplum Database。
3. 在登陆页面填写数据库连接信息后单击登录。

> 说明:

若连接失败请确认数据库连接信息是否正确，检查数据库白名单是否添加Tableau所在服务器IP地址，确认无误后重新登录。如何添加白名单请参见设置白名单。

配置项	描述
服务器	分析型数据库PostgreSQL版外网连接地址。详情请参见申请外网地址。
端口	默认为3432。
数据库	分析型数据库PostgreSQL版数据库名称。
用户名	分析型数据库PostgreSQL版数据库账户。
密码	分析型数据库PostgreSQL版数据库密码。

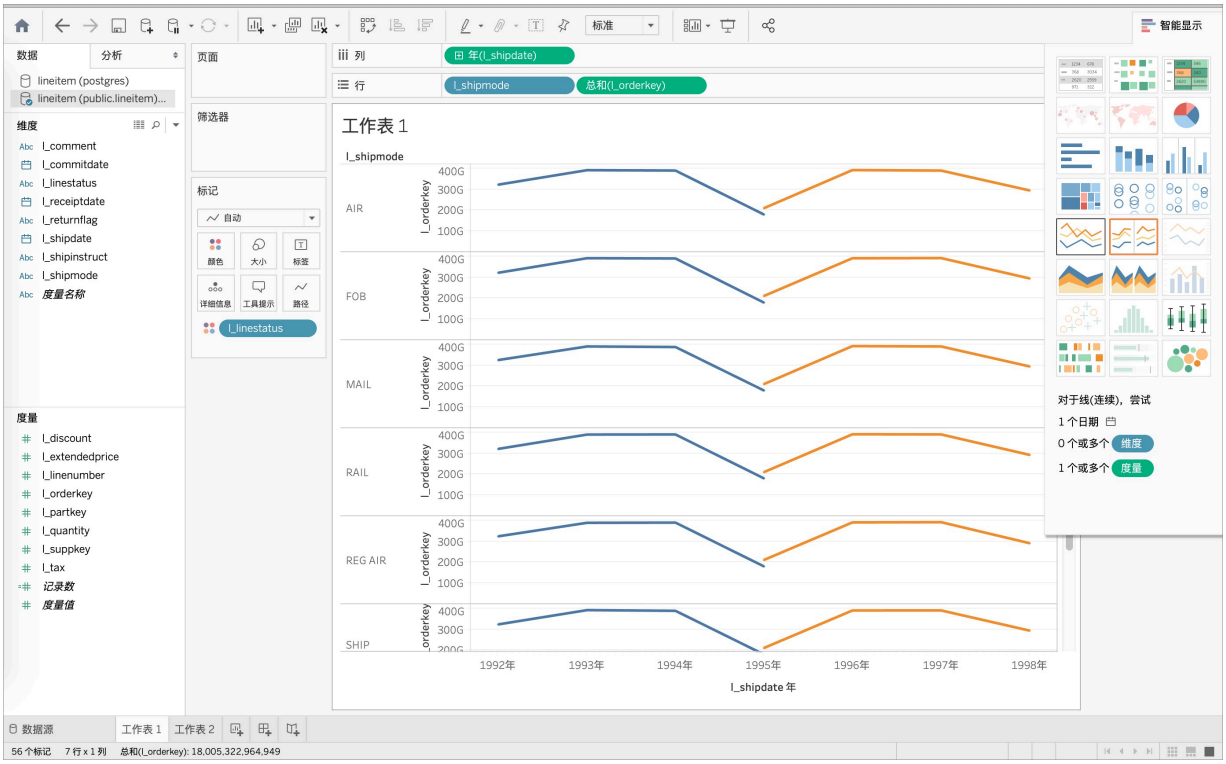
4. 成功登录后页面如下所示。

最佳实践

统计分析

根据指导操作，可以对任意表进行统计分析，并进行报表展示。

例如使用TPCH数据中的lineitem，点开一张工作表可以进行任意维度的数据展示。



当您从度量或者维度中选择字段，放到工作表区时，Tableau都会发送一个query到分析型数据库PostgreSQL版进行数据查询，例如上述图表发送的query如下所示：

```
BEGIN;declare "SQL_CUR0x7fdabf04ca00" cursor with hold for SELECT "
lineitem"."l_linestatus" AS "l_linestatus",
          "lineitem"."l_shipmode" AS "l_shipmode",
          SUM("lineitem"."l_orderkey") AS "sum_l_orderkey_ok",
          ((CAST("lineitem"."l_shipdate" AS DATE) + CAST(TRUNC((-1
 * (EXTRACT(DAY FROM "lineitem"."l_shipdate") - 1))) AS INTEGER) *
INTERVAL '1 DAY') + CAST(TRUNC((-1 * (EXTRACT(MONTH FROM "lineitem"."
l_shipdate") - 1))) AS INTEGER) * INTERVAL '1 MONTH') AS "tyr_l_ship
date_ok"
        FROM "public"."lineitem" "lineitem"
        GROUP BY 1,
          2,
          4;fetch 10000 in "SQL_CUR0x7fdabf04ca00
```

关闭cursor

默认情况下Tableau使用cursor模式从分析型数据库PostgreSQL版拉取数据：

```
FETCH 10000 in “SQL_CUR0x7fe678049e00”
```

如果提取的数据量很大，并且Tableau服务器的内存足够放下所有的查询数据，可以通过关闭cursor模式进行性能调优。

操作步骤如下：

1. 创建关闭cursor模式TDC文件，文件配置信息如下：

```
<?xml version='1.0' encoding='utf-8' ?>
<connection-customization class='greenplum' enabled='true' version='
4.3'>
<vendor name='greenplum'/>
<driver name='greenplum'/>
<customizations>
<customization name='odbc-connect-string-extras' value='UseDeclare
Fetch=0' />
</customizations>
</connection-customization>
```

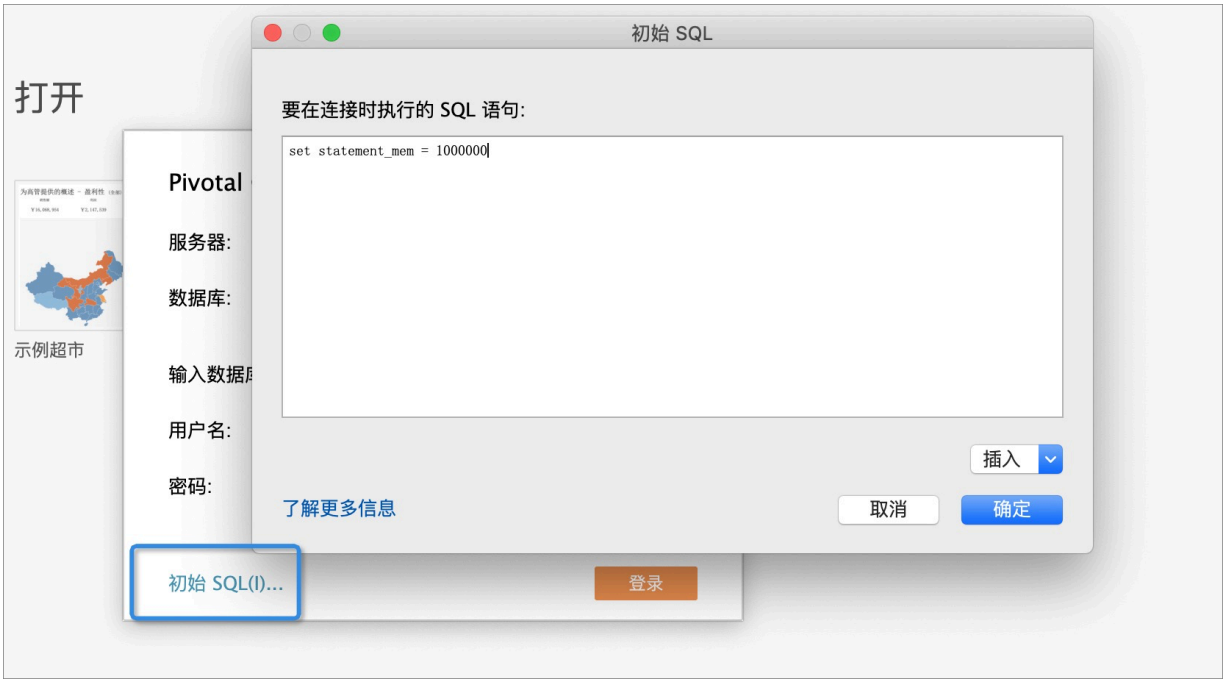
2. 将该文件以tdc为后缀名，Desktop版本的Tableau放到`Documents/My Tableau Repository/Datasources`目录下，其他版本的Tableau同样放置到对应的Datasources目录下。
3. 重启Tableau即可生效。

也可以修改fetch的size值，让其每次fetch更多的数据：

```
<?xml version='1.0' encoding='utf-8' ?>
<connection-customization class='greenplum' enabled='true' version='4.
3'>
<vendor name='greenplum'/>
<driver name='greenplum'/>
<customizations>
<customization name='odbc-connect-string-extras' value='Fetch=100000
' />
</customizations>
</connection-customization>
```

初始化SQL

连接建立时可以通过初始化SQL设置特定参数，如下图所示。



> **说明:**
> SQL语言结尾请不要添加英文分号（;），Tableau会将该SQL封装执行，中间如果有分号会报语法错误。同样在自定义SQL时，SQL结尾也不能加（;）。

10.4 帆软FineBI连接分析型数据库PostgreSQL版

分析型数据库PostgreSQL版基于开源数据库Greenplum构建，兼容Greenplum和PostgreSQL的语法、接口和生态。本文介绍如何通过FineBI连接分析型数据库PostgreSQL版。

前提条件

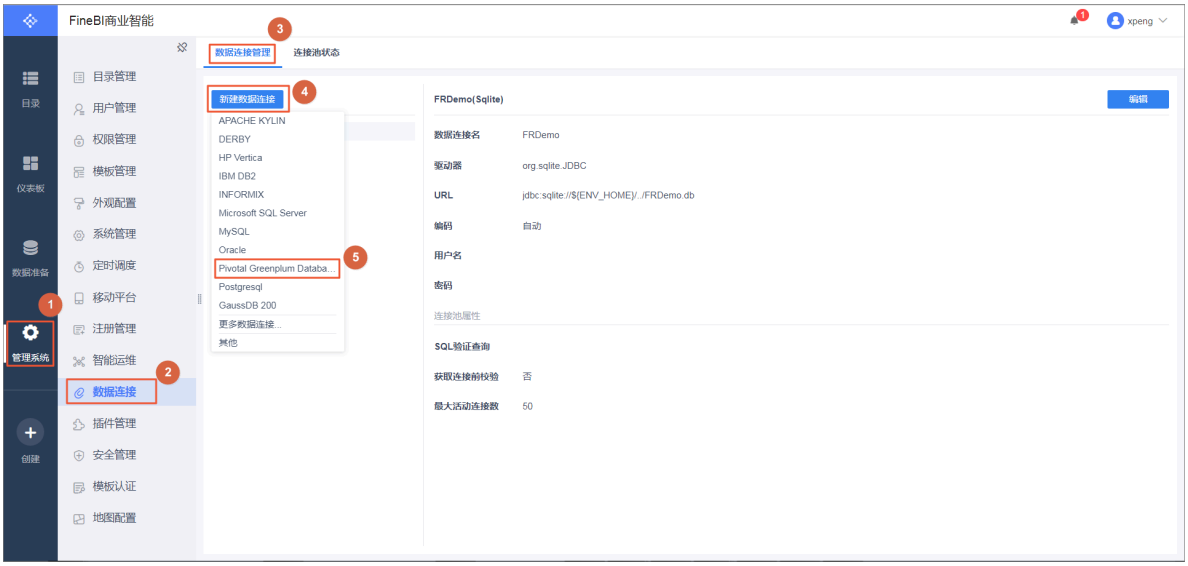
- 下载并安装FineBI。
- 已创建分析型数据库PostgreSQL版，详情请参见创建实例。
- 已在分析型数据库PostgreSQL版白名单中添加FineBI所在服务器IP地址，添加白名单请参见设置白名单。

注意事项

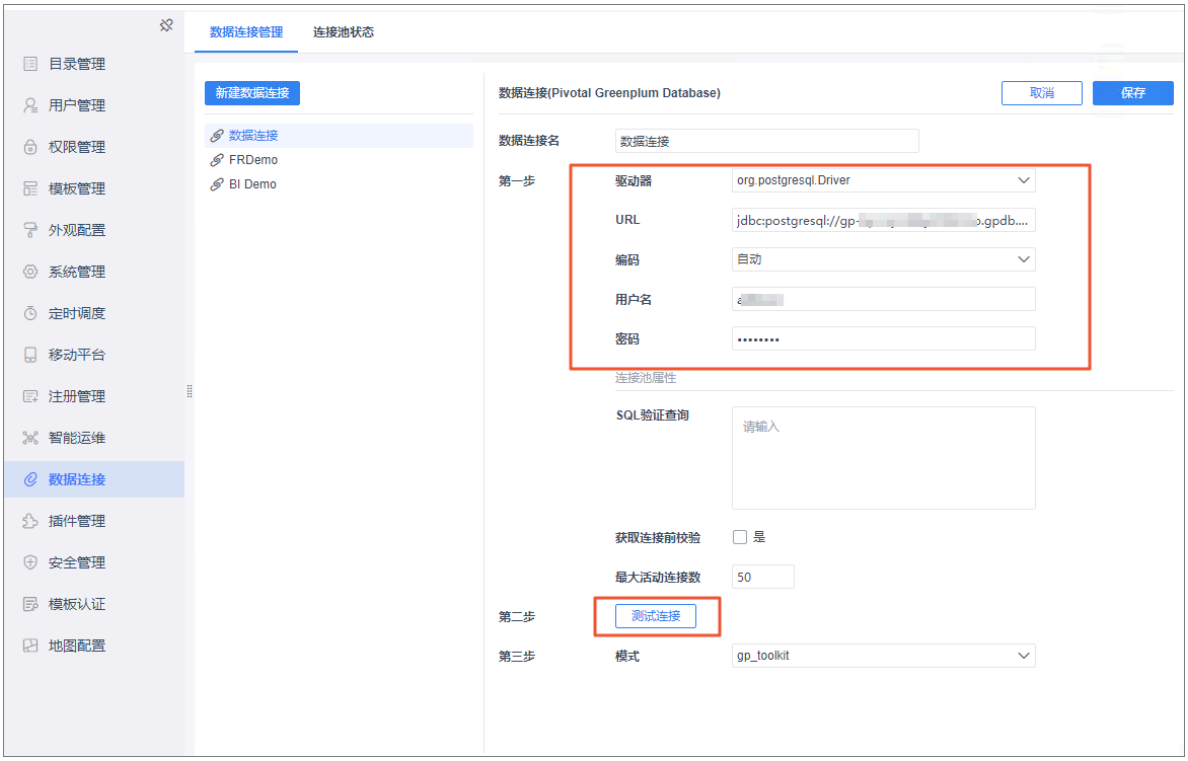
首次连接PostgreSQL数据源，需要下载JDBC Driver，下载地址和安装方式请参见FineBI文档。

操作步骤

1. 启动FineBI客户端。
2. 在左侧导航栏选择系统管理 > 数据连接 > 数据连接管理。
3. 在数据连接管理页面单击新建数据连接，选择Pivotal Greenplum Database。



4. 在数据连接管理页面填写URL、用户名和密码等信息。



配置项	描述	示例
驱动器	驱动器代码。	`org.postgresql.Driver`
URL	数据源连接信息。不同的驱动器，对应的URL格式不同，URL格式请参见FineBI驱动器配置信息。 **说明:** · 连接地址可以在分析型数据库PostgreSQL版控制台基本信息页面进行查看。 · 若FineBI与数据源不在同一可用区，需要通过外网地址访问，分析型数据库PostgreSQL版申请外网地址请参见申请外网地址。 · 分析型数据库PostgreSQL版默认端口为3432。	`jdbc:postgresql://gp-bpxxxxxxxxxxxxxx.gpdb.rds.aliyuncs.com:3432/dbname`
编码	默认为自动。	-
用户名	分析型数据库PostgreSQL版数据库账号。	`adbpgname`

配置项	描述	示例
密码	分析型数据库PostgreSQL版数据库密码。	adbpgpassword

表 10-1: FineBI驱动器配置信息

驱动器代码	URL	支持数据库版本
org.postgresql.Driver	jdbc:postgresql://连接地址:端口/数据库名称	4.3.9；5.0
com.pivotal.jdbc.GreenplumDriver	jdbc:pivotal:greenplum://连接地址:端口;DatabaseName=数据库名称	

5. 连接信息填写完成后单击测试连接。

> 说明:
> 如果测试连接失败，可以根据错误项的提示进行修改。

6. 测试连接通过后，点击右上角保存即可。