文档中心

Greenplum数据仓库 UDW

产品文档

优刻得科技股份有限公司 版权所有

目录

Copyright © 2012-2021 UCloud 优刻得

Copyright © 2012-2021 UCloud 优刻得

Copyright © 2012-2021 UCloud 优刻得

Copyright © 2012-2021 UCloud 优刻得

概览

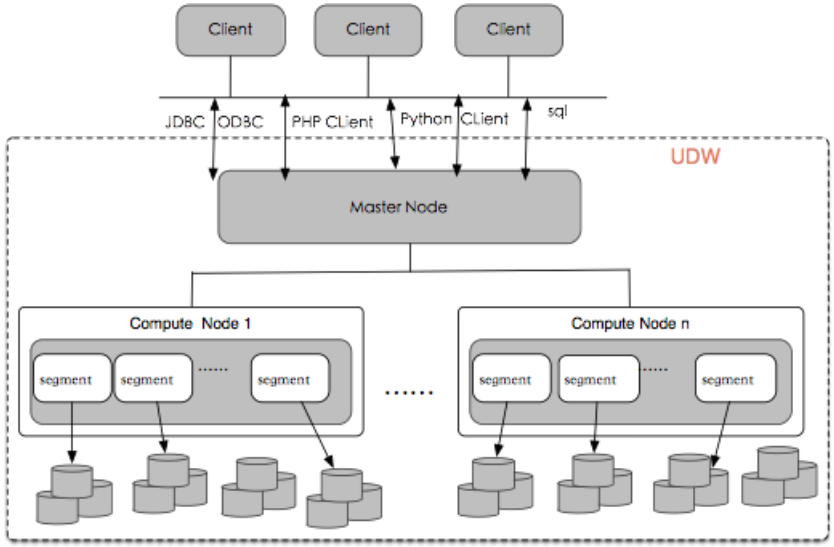
Copyright © 2012-2021 UCloud 优刻得

产品架构

数据仓库(UCloud Data Warehouse)是大规模并行处理数据仓库产品,基于开源的Greenplum开发的大规模并发、完全托管的PB级数据仓库服务。UDW可以通过SQL让数据分析更简单、高效,为互联网、物联网、金融、电信等行业提供丰富的业务分析能力。支持MADlib扩展,客户可以在udw上使用MADlib的扩展功能,从而让机器学习变得简单,支持PostGIS,可以方便的支持空间、地理位置应用。最新支持greeplum6.2.1版本。

云数据仓库产品架构

云数据库仓库 UDW 服务的架构图如下所示:

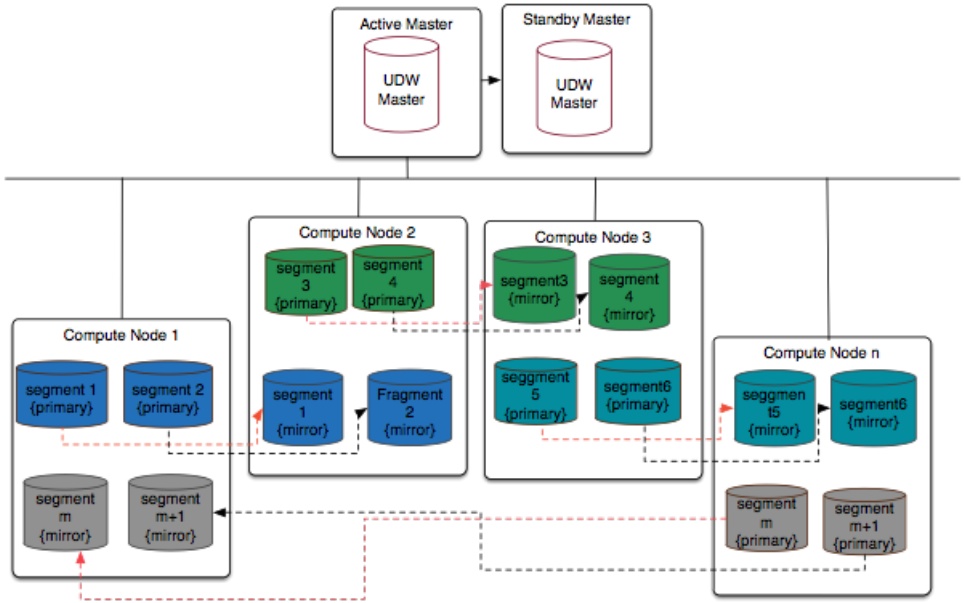


UDW 采用无共享的 MPP 架构,适用于海量数据的存储和计算。UDW 的架构如上图所示,主要有 Client、Master Node 和 Compute Node 组成。基本组成部分的功能如下:

Copyright © 2012-2021 UCloud 优刻得

1. Client：访问 UDW 的客户端
    - 支持通过 JDBC、ODBC、PHP、Python、命令行 Sql 等方式访问 UDW
2. Master Node：访问 UDW 数据仓库的入口
    - 接收客户端的连接请求
    - 负责权限认证
    - 处理 SQL 命令
    - 调度分发执行计划
    - 汇总 Segment 的执行结果并将结果返回给客户端
3. Compute Node：
    - Compute Node 管理节点的计算和存储资源
    - 每个 Compute Node 由多个 Segment 组成
    - Segment 负责业务数据的存储、用户 SQL 的执行

高可用



Copyright © 2012-2021 UCloud 优刻得

如上图所示：

1. Compute Node 中任一 Segment 都会有一个 Mirror Segment 备份到其他的 Compute Node 上，当 Primary Segment 出现不可用的时候会自动切换到 Mirror Segment，当 Primary Segment 恢复之后，Primary Segment 会自动恢复这期间的变更。
2. Master 节点是主从模式，当 Active Master 不可用时会自动切换到 Standby Master。

Copyright © 2012-2021 UCloud 优刻得

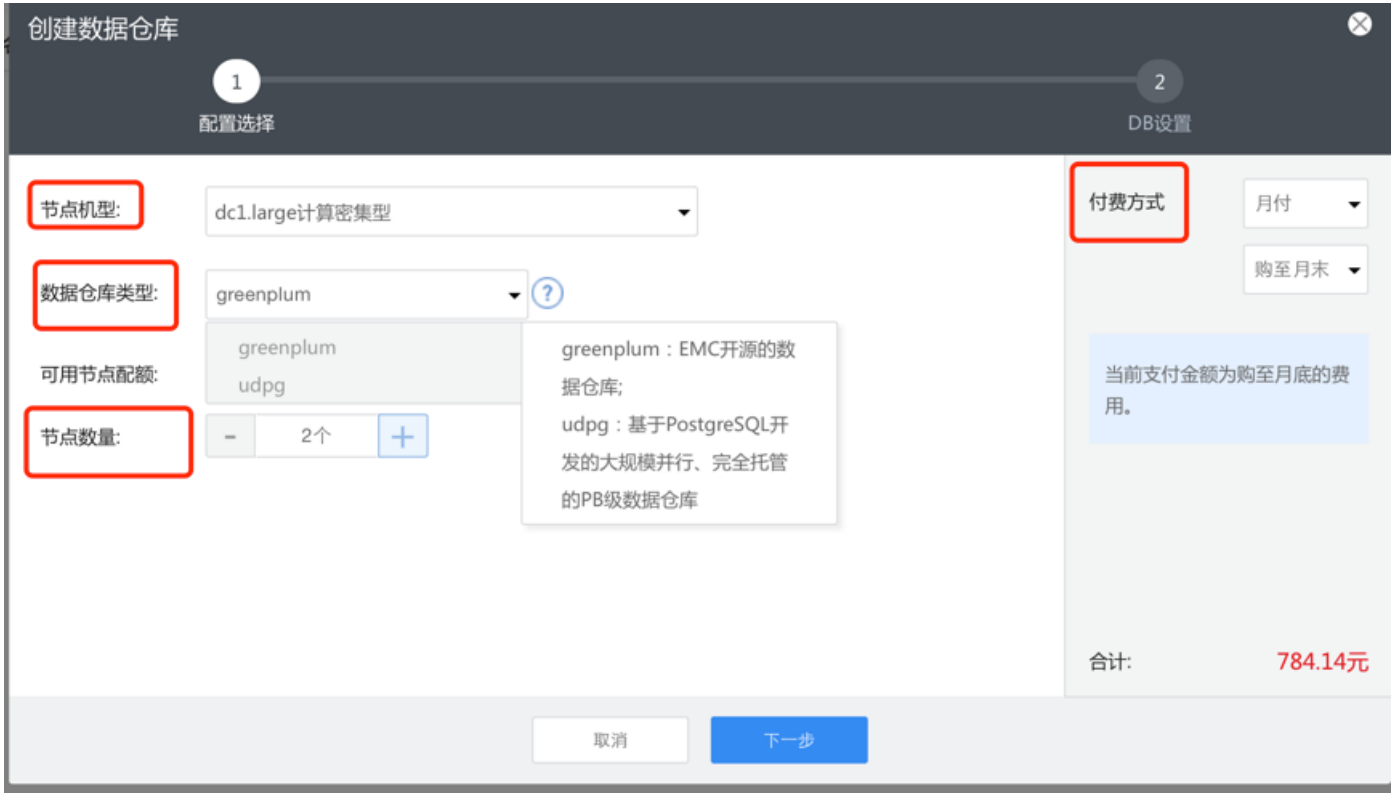
快速上手

一、创建数据仓库

1.选择UDW标签可以跳转到UDW操作界面(如果没有这个标签,请联系客服申请开通),点击欢迎页的“开始探索”,然后点击“创建数据仓库”。

Copyright © 2012-2021 UCloud 优刻得

2.选择计算节点机型、计算节点数量以及付费方式。



其中可选的机型配置有：

机型	名称	配置
存储密集型	ds1.large	4核 24G 2000G(SATA)
存储密集型	ds1.6xlarge	24核 144G 12000G(SATA)
计算密集型	dc1.large	2核 12G 300G(SSD)

Copyright © 2012-2021 UCloud 优刻得

计算密集型	dc1.8xlarge	28核 168G 3800G(SSD)

选择数据仓库类型:Greenplum 是 EMC 开源的数据仓库产品、Udpg 是基于 PostgreSQL 开发的大规模并行、完全托管的 PB 级数据仓库服务。

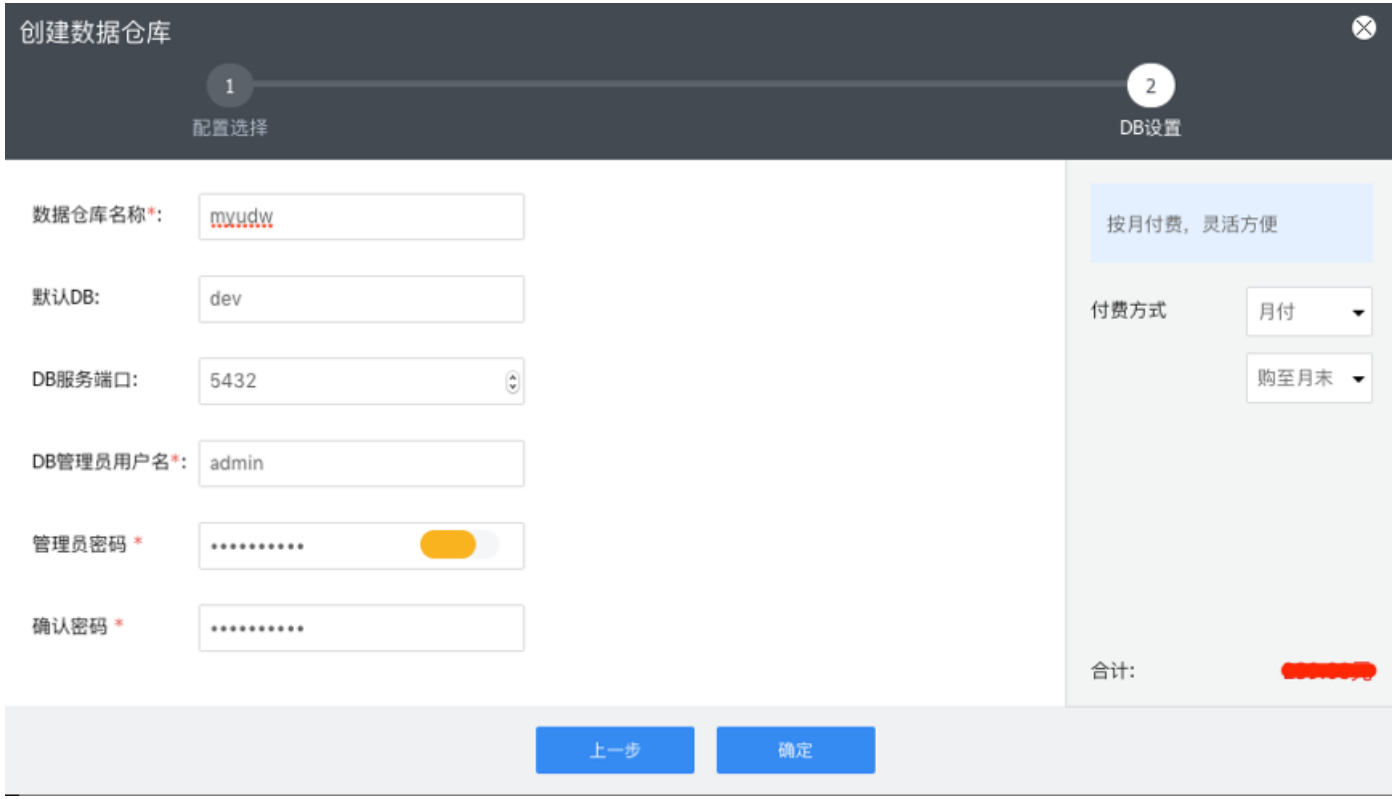
选择节点个数:UDW 是分布式架构、所有节点数据都是双机热备,实际可用总容量略小于节点个数*节点磁盘大小/2,请根据实际数据大小选择合适的节点。

3.设置数据仓库信息

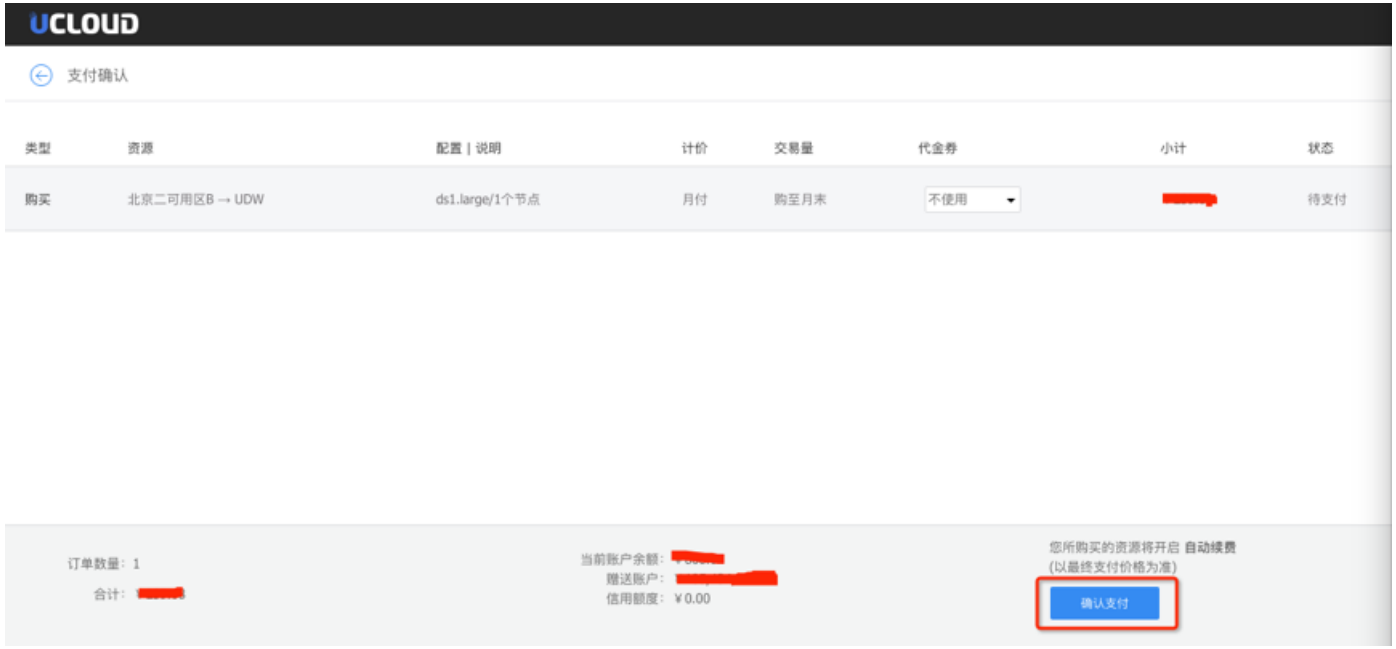
必选项有数据仓库名称、DB管理员用户名、管理员密码。可选项有默认DB,默认DB的名称为dev,你可以选择除了“test”、“postgres”、“template”、“template0”、“template1” 、“default”之外的其他名称。

DB管理员用户名不能为“postgres”。端口固定为 5432,暂不提供修改。

Copyright © 2012-2021 UCloud 优刻得

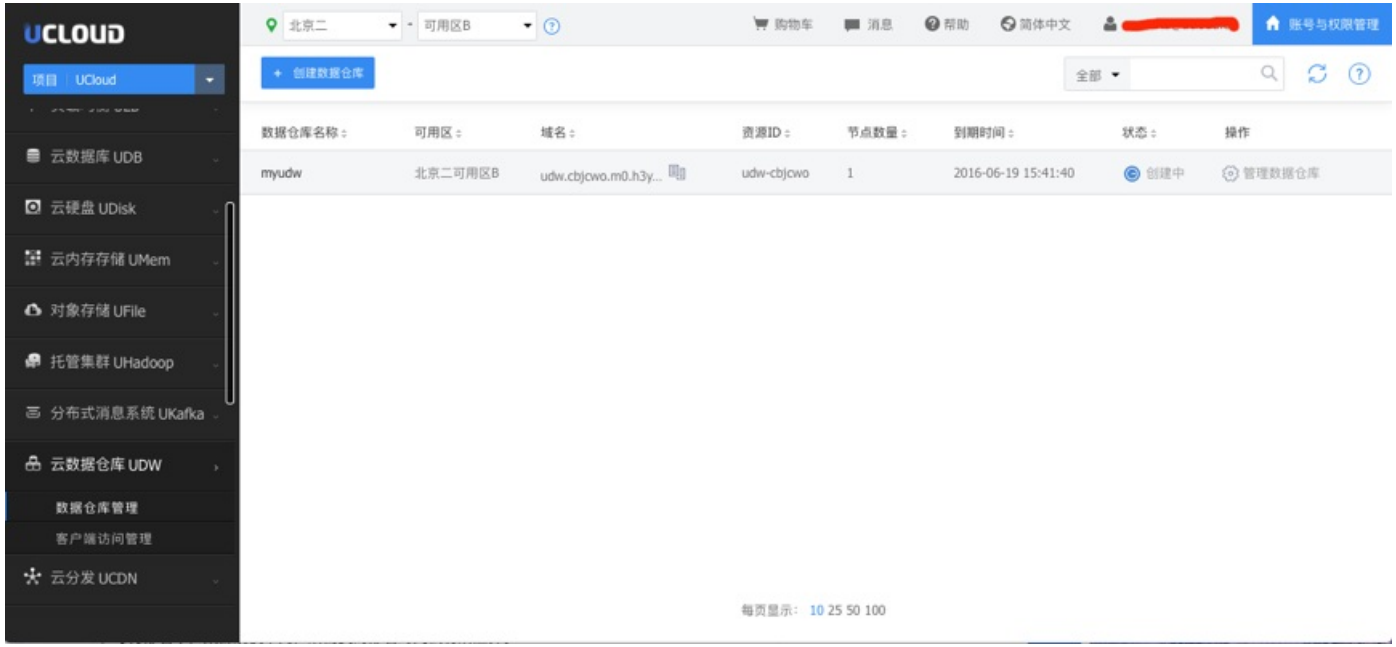

创建数据仓库
1
配置选择
2
DB设置
数据仓库名称*:
myudw
默认DB:
dev
DB服务端口:
5432
DB管理员用户名*:
admin
管理员密码 *
确认密码 *
按月付费，灵活方便
付费方式
月付
购至月末
合计:
上一步
确定

4.确认支付



5.等待部署中 数据仓库规模不同，所需要的部署时间会有所差异。

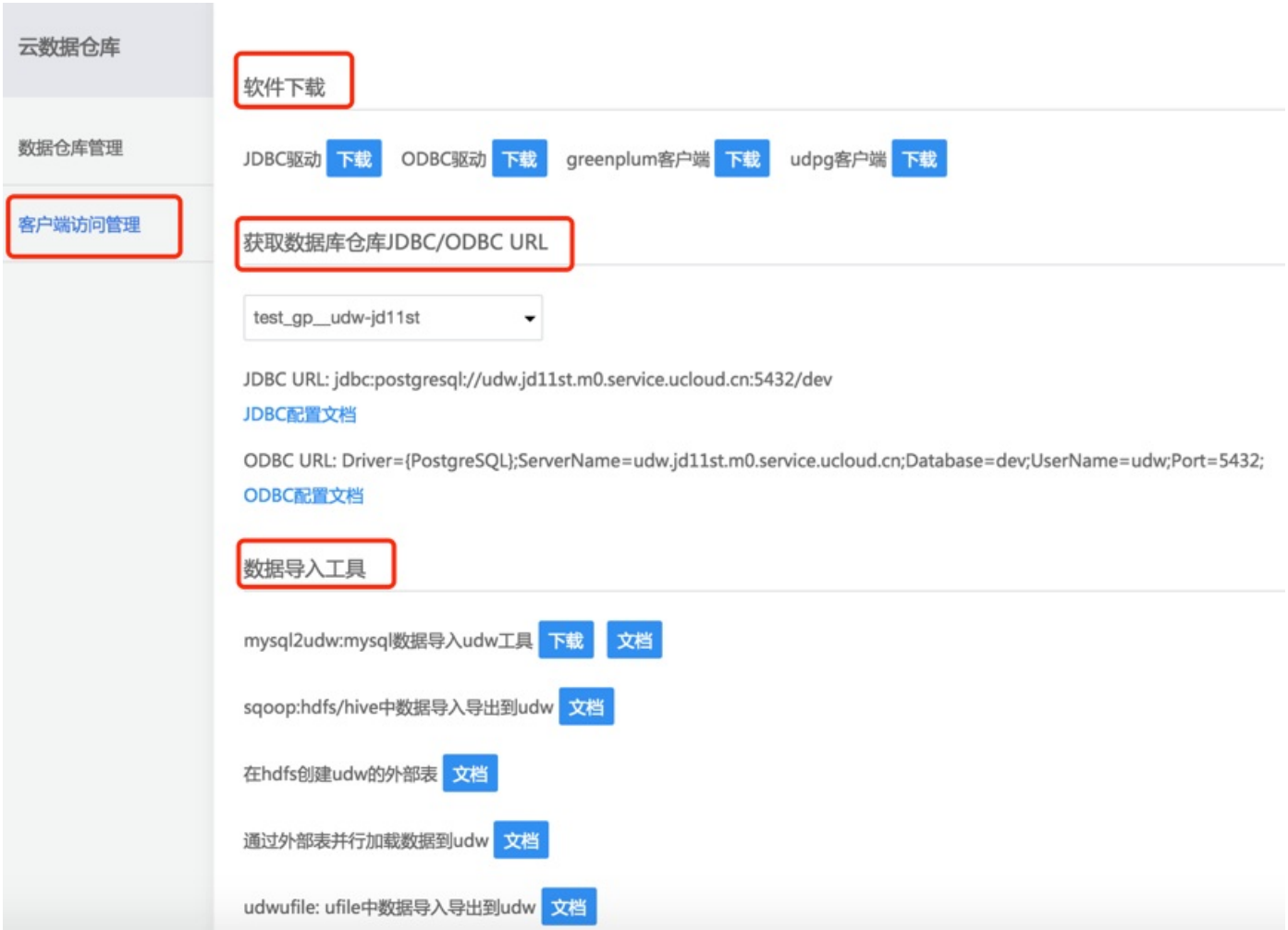
Copyright © 2012-2021 UCloud 优刻得



Copyright © 2012-2021 UCloud 优刻得

二、连接数据仓库

Copyright © 2012-2021 UCloud 优刻得



如上图所示客户端访问管理,提供了客户端下载和数据加载工具和文档的下载。

JDBC连接

Linux操作系统

Copyright © 2012-2021 UCloud 优刻得

```
yum install postgresql-jdbc.noarch –y
```

Windows 环境下 JDBC 驱动，将 jar 添加到工程的 BUILD PATH。

示例程序1，java连接UDW，执行建表，插入操作

PostgreSQLJDBC1.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class PostgreSQLJDBC1 {
public static void main(String args[]) {
Connection c = null;
Statement stmt = null;
try {
Class.forName("org.postgresql.Driver");
c = DriverManager.getConnection("jdbc:postgresql://hostIP:port/dbname",”UserName”,”Password”);
stmt = c.createStatement();
String sql = "CREATE TABLE COMPANY " + "(ID INT PRIMARY KEY NOT NULL," + "NAME TEXT NOT NULL," + "AGE INT NOT NULL," + "ADDRESS CHAR(50),"
+ "SALARY REAL)";
c.setAutoCommit(false);
System.out.println("Opened database successfully");
stmt.executeUpdate(sql);
sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " + "VALUES (1, 'Allen', 25 , 'Texas', 15000.00 );";
stmt.executeUpdate(sql);
```

Copyright © 2012-2021 UCloud 优刻得

```
stmt.close();
c.commit();
c.close();
}
catch (Exception e) {
e.printStackTrace();
System.err.println(e.getClass().getName()+": "+e.getMessage());
System.exit(0);
}
System.out.println("Opened database successfully");
}
}
```

示例程序二:java连接UDW,执行查询操作

PostgreSQLJDBC2.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class PostgreSQLJDBC2 {
public static void main(String[] args) {
Connection c = null;
Statement stmt = null;
```

Copyright © 2012-2021 UCloud 优刻得

```
try{
Class.forName("org.postgresql.Driver");
c = DriverManager.getConnection("jdbc:postgresql://hostIP:port/dbname",”UserName”,”Password”);
stmt = c.createStatement();
String sql = null;
System.out.println("Opened database successfully");
sql = "SELECT * FROM COMPANY;";
ResultSet res=stmt.executeQuery(sql);
while(res.next()) {
System.out.println(res.getInt(1));
System.out.println(res.getString(2));
System.out.println(res.getInt(3));
System.out.println(res.getString(4));
System.out.println(res.getDouble(5));
}
stmt.close();
c.close();
}
catch(Exception e) {
System.err.println( e.getClass().getName()+": "+ e.getMessage() );
System.exit(0);
}
}
}
```

Copyright © 2012-2021 UCloud 优刻得

ODBC方式连接

Linux操作系统:CentOS 6.5 64位

1. 安装 postgresql odbc驱动

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

2. 编辑/etc/odbcinst.ini文件,配置odbc驱动

```
Description = ODBC for PostgreSQL
Driver = /usr/lib/psqlodbc.so
Setup = /usr/lib/libodbcpsqlS.so
Driver64 = /usr/lib64/psqlodbc.so
Setup64 = /usr/lib64/libodbcpsqlS.so
FileUsage = 1
```

3. 测试ODBC驱动是否安装成功

```
# odbcinst -q -d
[PostgreSQL]
```

如果出现以上输出,代表在这台机器上已成功安装了PostgreSQL的ODBC驱动。

4. 编辑/etc/odbc.ini文件配置ODBC连接

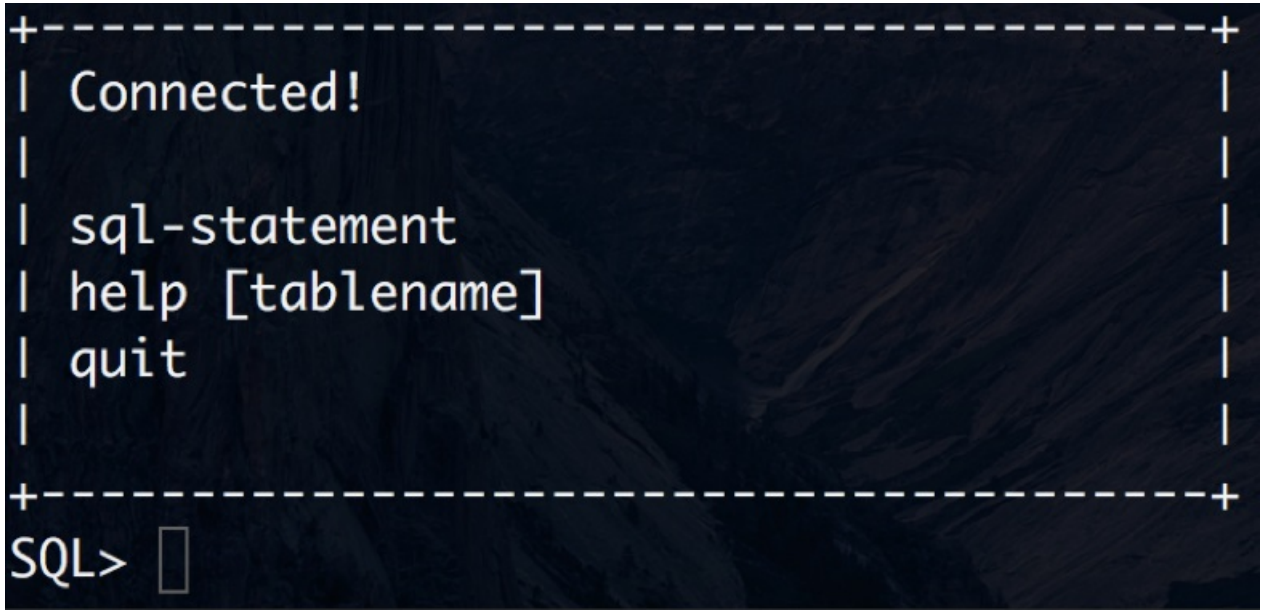
Copyright © 2012-2021 UCloud 优刻得

```
[testdb]Description = PostgreSQL connection to TestDB
Driver = PostgreSQL
Database = Database
Servername = MasterNodeIP
UserName = UserName
Password = Password
Port = Port
Protocol = 8.3
ReadOnly = No
RowVersioning = NoShow
SystemTables = No
ConnSettings =
```

5. 测试连接

```
# isql testdb
```

Copyright © 2012-2021 UCloud 优刻得



如出现以上内容，则表示psqlodbc配置成功。

其他方式

1.udw客户端的方式访问

1.1 udw（greenplum）客户端方式访问（以Centos为例）

如果你选择的数据仓库类型是greenplum、可以采用下面的方式访问

1）下载greenplum客户端解压

Copyright © 2012-2021 UCloud 优刻得

```
wget http://udw.cn-bj.ufileos.com/greenplum-client.tar

tar -zxvf greenplum-client.tar.gz
```

2)配置udw客户端

进入 greenplum-client 安装目录,编辑 greenplum_client_path.sh 修改UDW_HOME(export UDW_HOME= client安装目录)(如/root/greenplum-client)

3) 使配置生效

在~/.bashrc中添加如下配置

```
source /data/greenplum-client/greenplum\_client\_path.sh
```

执行

```
source ~/.bashrc
```

备注:/data/greenplum-client是greenplum-client的安装路径

4) 连接数据库

```
psql -h hostIP(或域名) –U username -d database -p port –W
```

1.2 udw(udpg)客户端方式访问(以Centos为例)

如果你选择的数据仓库类型是udpg、可以采用下面的方式访问

Copyright © 2012-2021 UCloud 优刻得

1）下载udw客户端

```
wget http://udw.cn-bj.ufileos.com/udw-client.tar

tar xvf udw-client.tar
```

2）配置udw客户端

进入udw-client安装目录，编辑 udw_client_path.sh，修改 UDW_CLIENT(export UDW_CLIENT= client安装目录)（如/root/udw-client）

3）使配置生效在~/.bashrc中添加如下配置

```
source /data/udw-client/udw_client_path.sh
```

执行:

```
source ~/.bashrc
```

> 备注：/data/udw-client是udw-client的安装路径

4）连接数据库

```
psql -h hostIP（或域名）–U username -d database -p port –W
```

Copyright © 2012-2021 UCloud 优刻得

2.python客户端访问

```
$yum install python-psycopg2
```

示例1. 连接UDW testconn.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
```

执行 python testconn.py

示例2. 创建一个表 createTable.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute('''CREATE TABLE COMPANY
(ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
```

Copyright © 2012-2021 UCloud 优刻得

```
ADDRESS CHAR(50),
SALARY REAL);''')
print "Table created successfully"
conn.commit()
conn.close()
```

示例3. 插入记录 insert.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1, 'Paul', 32, 'California', 20000.00 )");
cur.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");
conn.commit()
print "Records created successfully";
conn.close()
```

示例4. 查询 select.py

```
#!/usr/bin/python
```

Copyright © 2012-2021 UCloud 优刻得

```
import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("SELECT id, name, address, salary from COMPANY")
rows = cur.fetchall()
for row in rows:
print "ID = ", row[0]
print "NAME = ", row[1]
print "ADDRESS = ", row[2]
print "SALARY = ", row[3], "\n"
print "Operation done successfully";
conn.close()
```

示例5. 更新 update.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("UPDATE COMPANY set SALARY = 25000.00 where ID=1")
conn.commit
```

Copyright © 2012-2021 UCloud 优刻得

```
print "Total number of rows updated :", cur.rowcount
cur.execute("SELECT id, name, address, salary from COMPANY")
rows = cur.fetchall()
for row in rows:
print "ID = ", row[0]
print "NAME = ", row[1]
print "ADDRESS = ", row[2]
print "SALARY = ", row[3], "\n"
print "Operation done successfully";
conn.close()
```

示例6. 删除 delete.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("DELETE from COMPANY where ID=2;")
conn.commit
print "Total number of rows deleted :", cur.rowcount
cur.execute("SELECT id, name, address, salary from COMPANY")
rows = cur.fetchall()
for row in rows:
```

Copyright © 2012-2021 UCloud 优刻得

```
print "ID = ", row[0]
print "NAME = ", row[1]
print "ADDRESS = ", row[2]
print "SALARY = ", row[3], "\n"
print "Operation done successfully";
conn.close()
```

3.php客户端

```
yum install php-pgsql
```

示例1. 连接 conn.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
?>
```

Copyright © 2012-2021 UCloud 优刻得

示例2. 创建表 create.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
CREATE TABLE COMPANY
(ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR(50),
SALARY REAL);
EOF;
$ret = pg_query($db, $sql);
if(!$ret)
{ echo pg_last_error($db);
```

Copyright © 2012-2021 UCloud 优刻得

```
} else {
echo "Table created successfullyn";
}
pg_close($db);
?>
```

示例3. 插入 insert.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00 );
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Allen', 25, 'Texas', 15000.00 );
EOF;
```

Copyright © 2012-2021 UCloud 优刻得

```
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
} else {
echo "Records created successfully\n";
}
pg_close($db);
?>
```

示例4. 查询 select.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
SELECT * from COMPANY;
EOF;
```

Copyright © 2012-2021 UCloud 优刻得

```
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
}
while($row = pg_fetch_row($ret)){
echo "ID = ". $row[0] . "\n";
echo "NAME = ". $row[1] ."\n";
echo "ADDRESS = ". $row[2] ."\n";
echo "SALARY = ".$row[4] ."\n\n";
}
echo "Operation done successfully\n";
pg_close($db);
?>
```

示例5. 更新 update.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
```

Copyright © 2012-2021 UCloud 优刻得

```
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
SELECT * from COMPANY;
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
}
while($row = pg_fetch_row($ret)){
echo "ID = ". $row[0] . "\n";
echo "NAME = ". $row[1] ."\n";
echo "ADDRESS = ". $row[2] ."\n";
echo "SALARY = ".$row[4] ."\n\n";
}
echo "Operation done successfully\n";
pg_close($db);
?>
```

示例6. 删除 delete.php

```
<?php
$host = "host=hostIP";
```

Copyright © 2012-2021 UCloud 优刻得

```
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
DELETE from COMPANY where ID=2;
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
} else {
echo "Record deleted successfully\n";
}
$sql =<<<EOF
SELECT * from COMPANY;
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
```

Copyright © 2012-2021 UCloud 优刻得

```
exit;
}
while($row = pg_fetch_row($ret)){
echo "ID = ". $row[0] . "\n";
echo "NAME = ". $row[1] ."\n";
echo "ADDRESS = ". $row[2] ."\n";
echo "SALARY = ".$row[4] ."\n\n";
}
echo "Operation done successfully\n";
pg_close($db);
?>
```

4.SQL Workbench/J 访问 udw

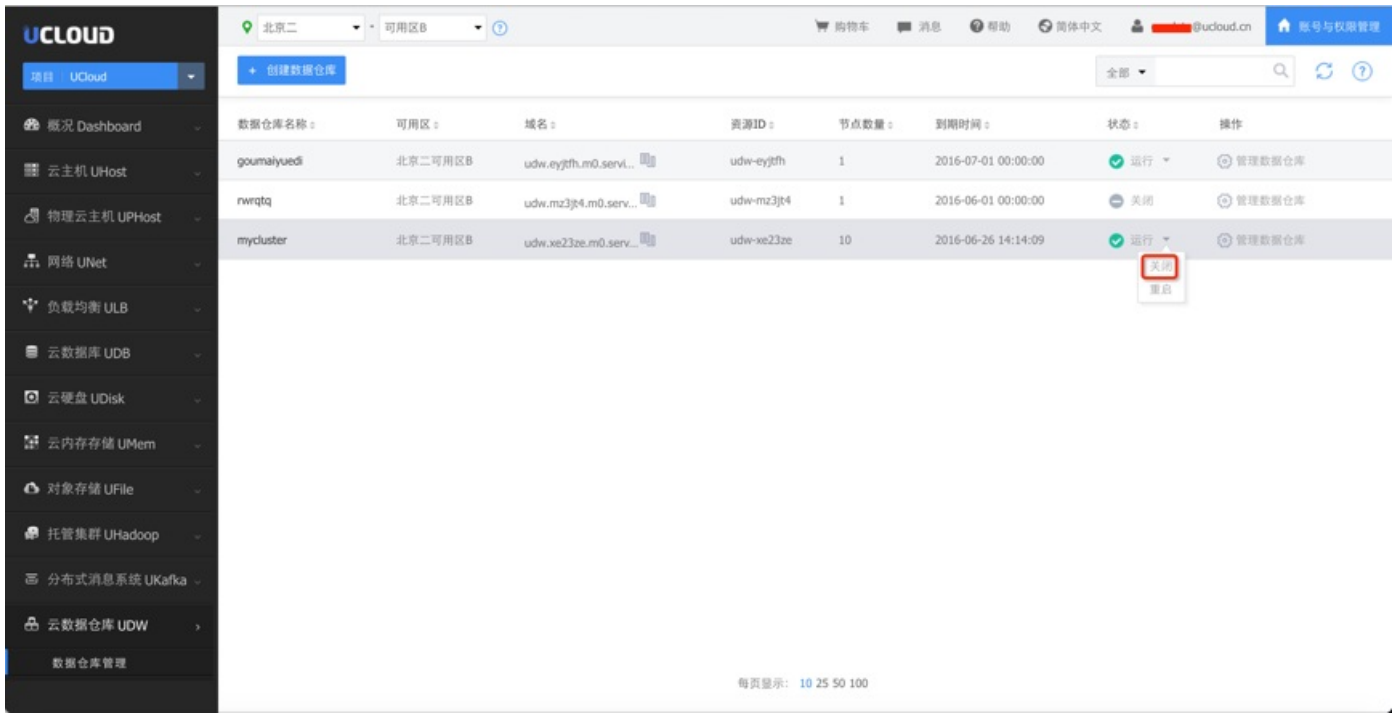
除了以上几种方式，UDW还可以使用SQL Workbench/J来进行访问，详情可见：SQL Workbench/J 访问udw

Copyright © 2012-2021 UCloud 优刻得

操作指南

关闭数据仓库

当数据仓库创建成功之后处于运行状态，可以在数据仓库列表页中关闭数据仓库。



Copyright © 2012-2021 UCloud 优刻得

启动数据仓库

参见关闭数据仓库

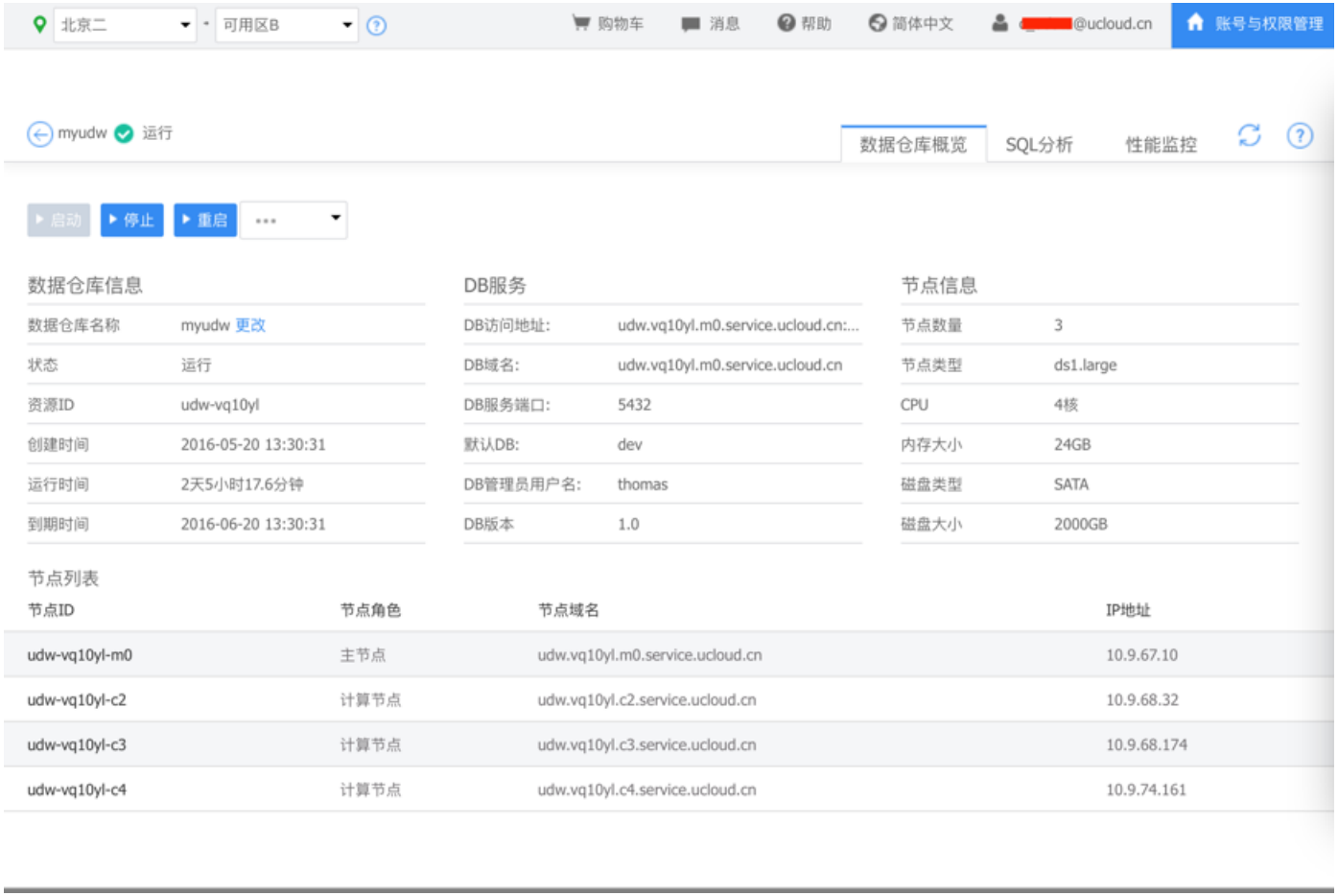
重启数据仓库

参见关闭数据仓库

查看数据仓库详情

点击“管理数据仓库”进入二级页面。

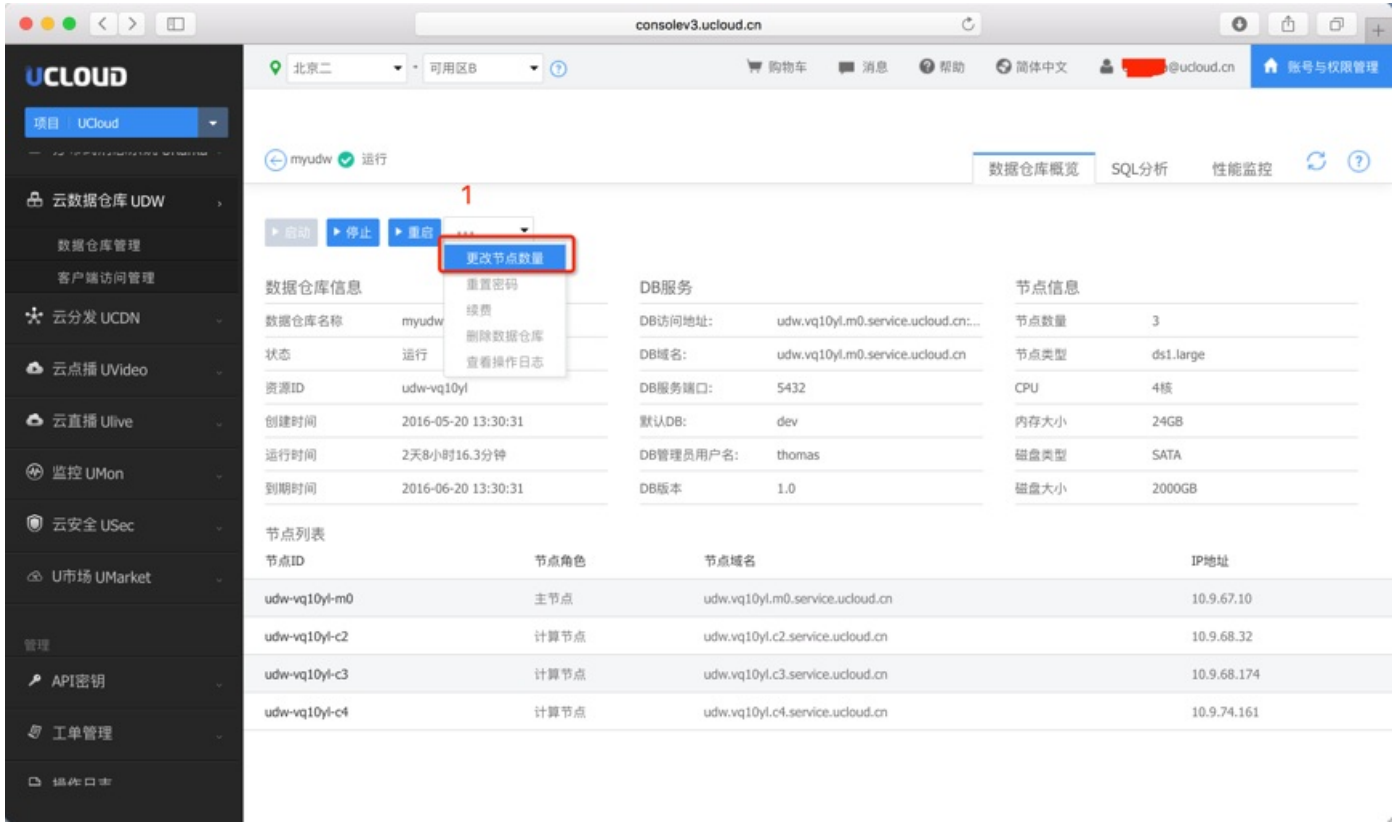
Copyright © 2012-2021 UCloud 优刻得



该页面上可以看到数据仓库的详细信息，包括db信息以及节点信息。在这个页面上也可以启动、停止或重启数据仓库。点击数据仓库名称右边的“更改”，可更改当前数据仓库名称。

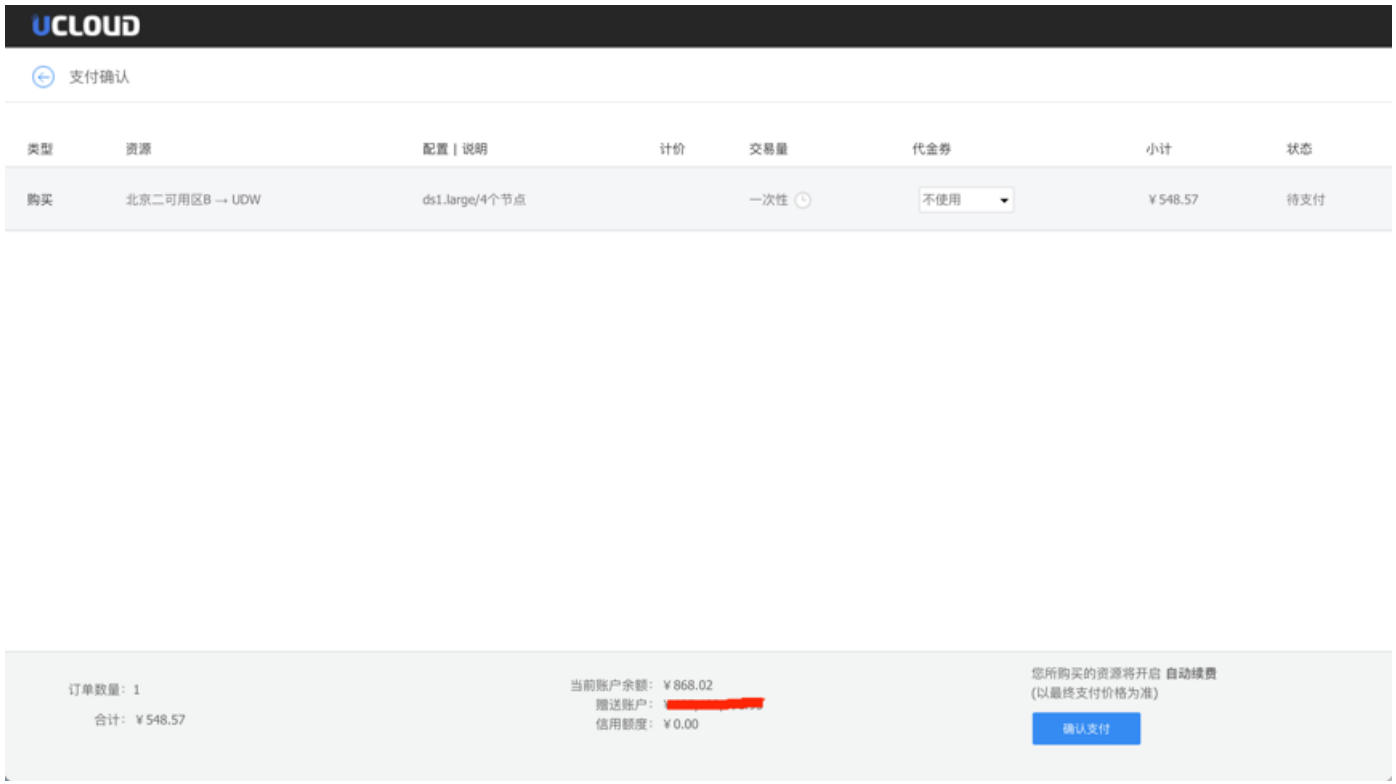
扩容数据仓库

Copyright © 2012-2021 UCloud 优刻得



Copyright © 2012-2021 UCloud 优刻得

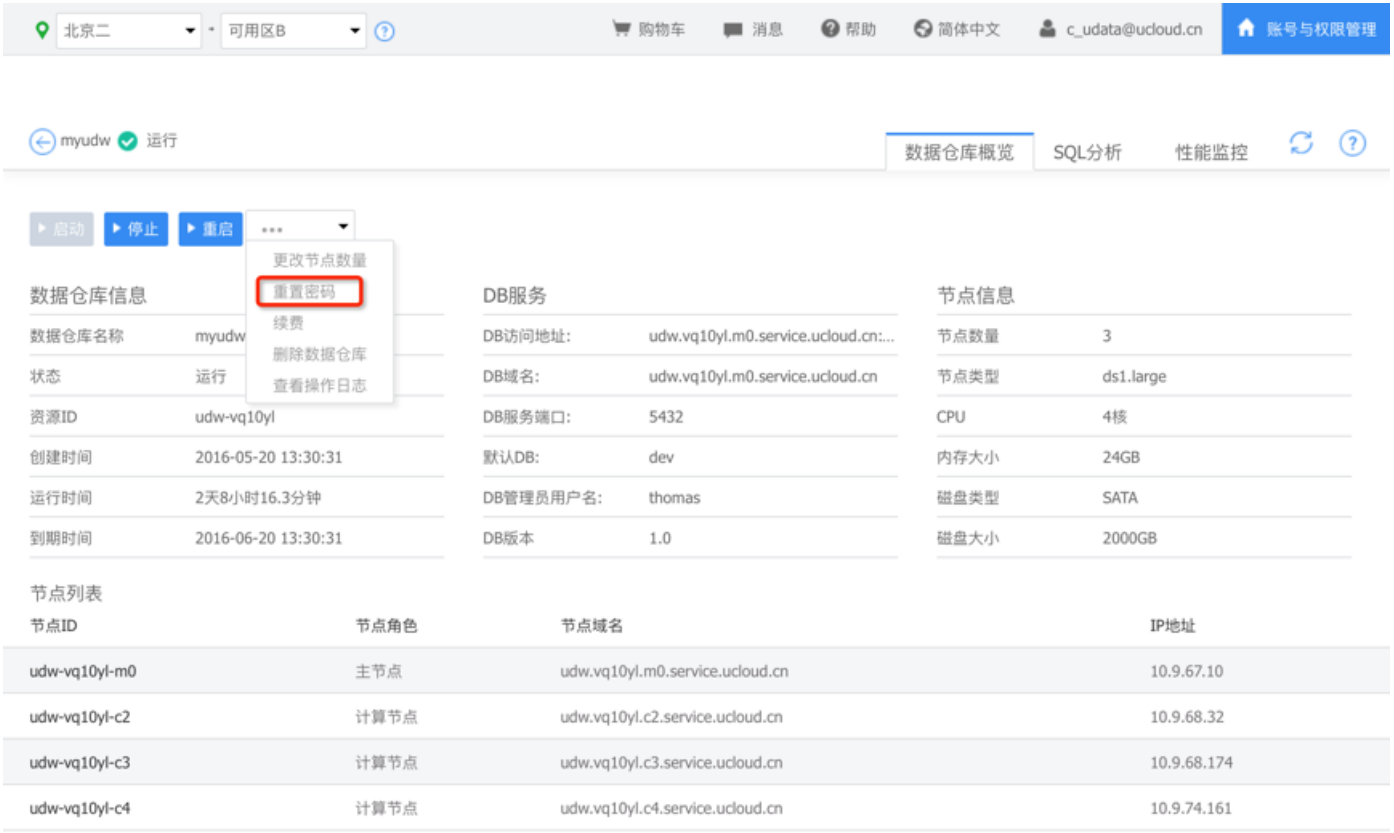
Copyright © 2012-2021 UCloud 优刻得



数据仓库扩容过程中需要对数据进行重分布，因此，扩容完成的时间根据数据量的大小而不同。目前，暂时不支持数据仓库的缩容。

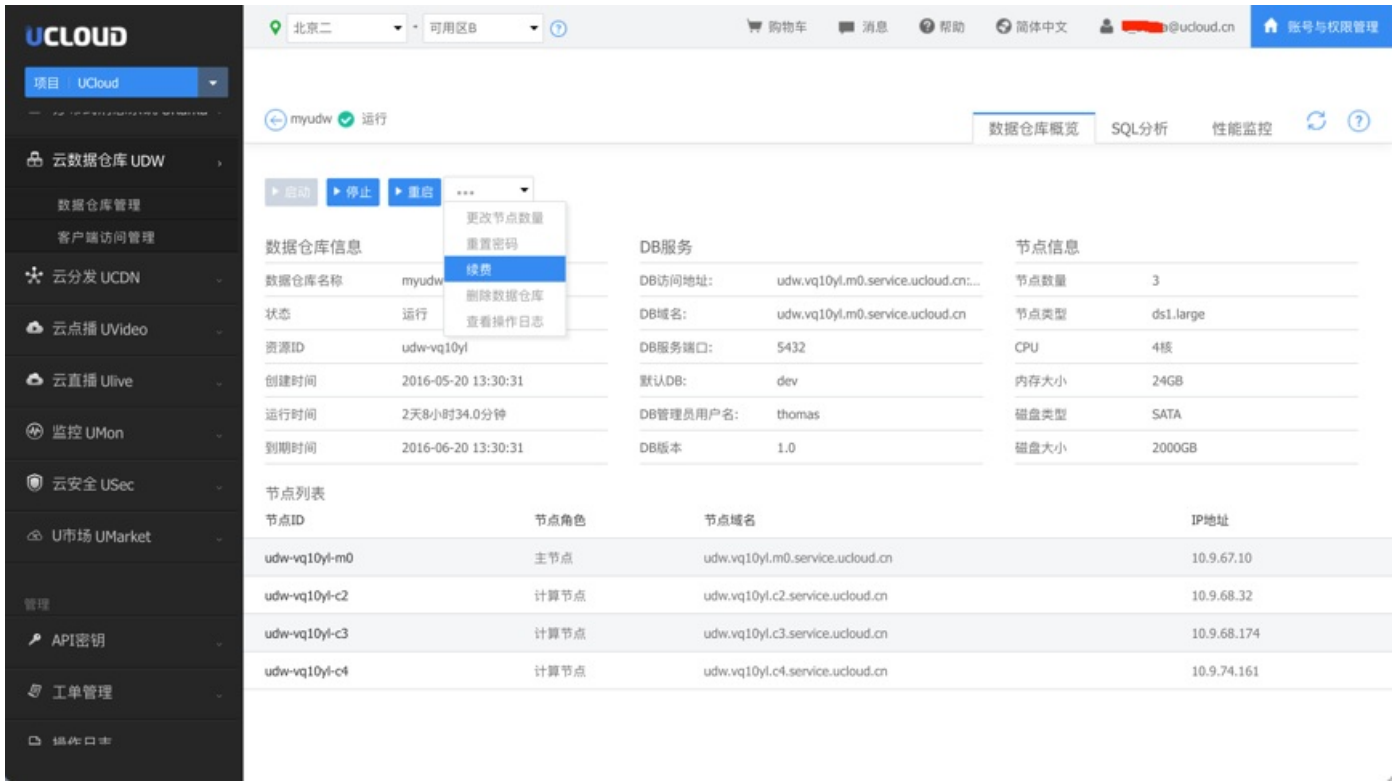
更改数据仓库密码

Copyright © 2012-2021 UCloud 优刻得



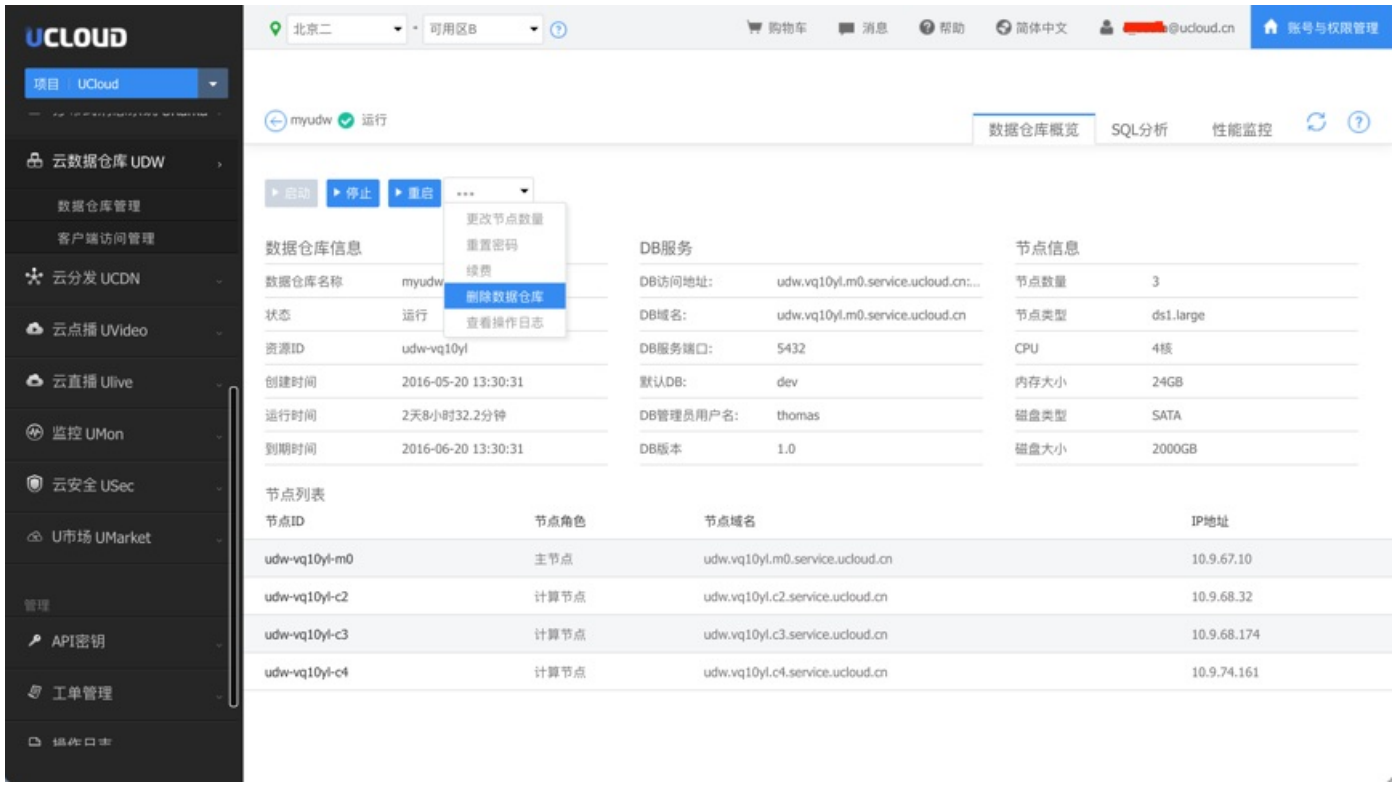
续费

Copyright © 2012-2021 UCloud 优刻得



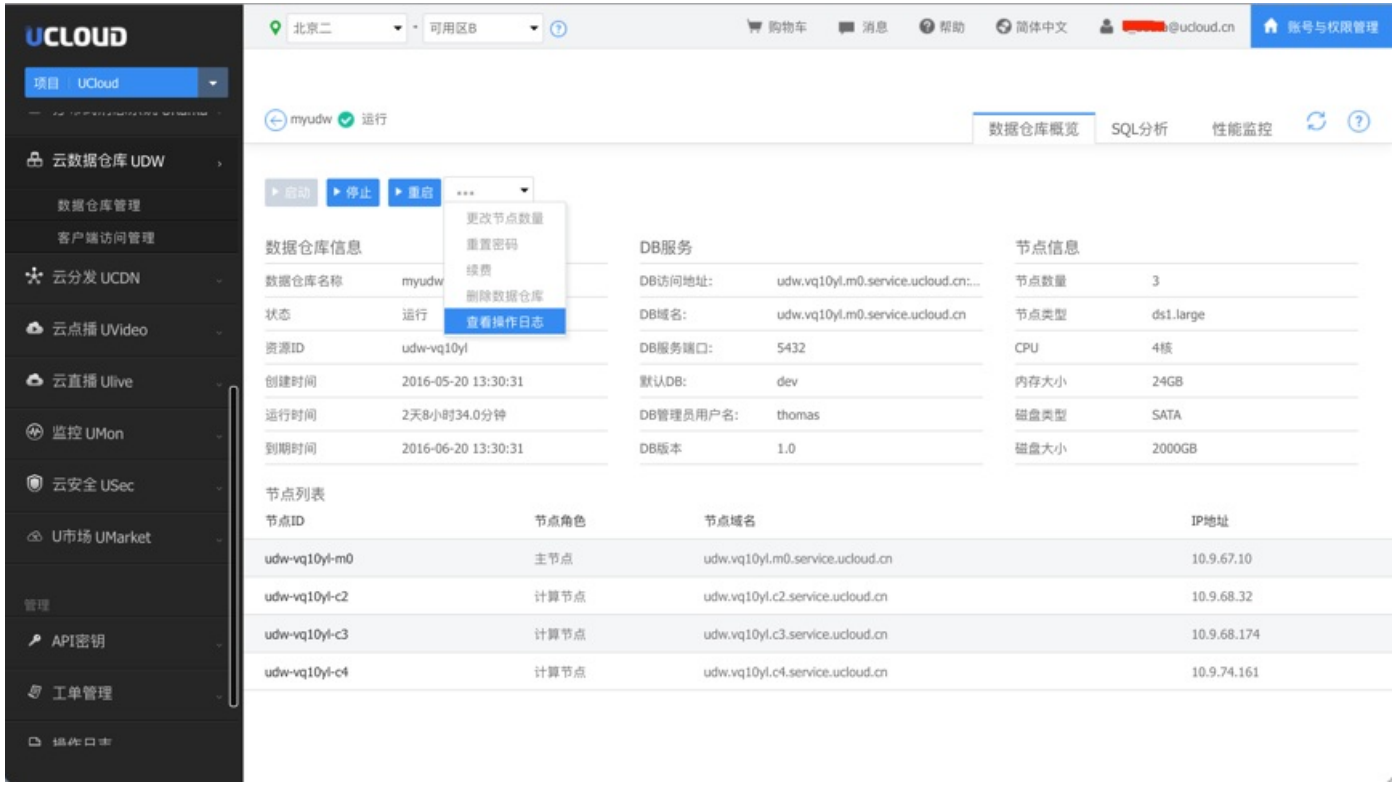
删除数据仓库

Copyright © 2012-2021 UCloud 优刻得



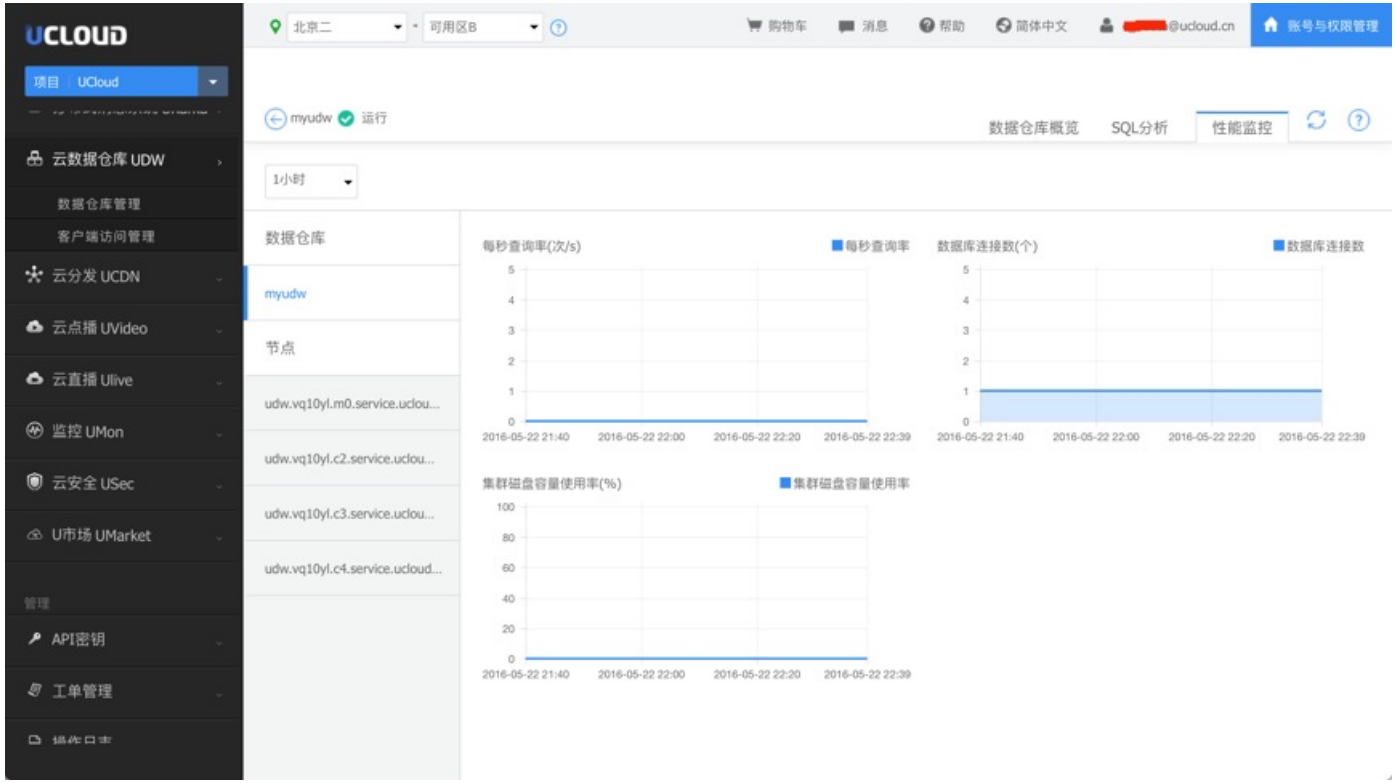
查看操作日志

Copyright © 2012-2021 UCloud 优刻得

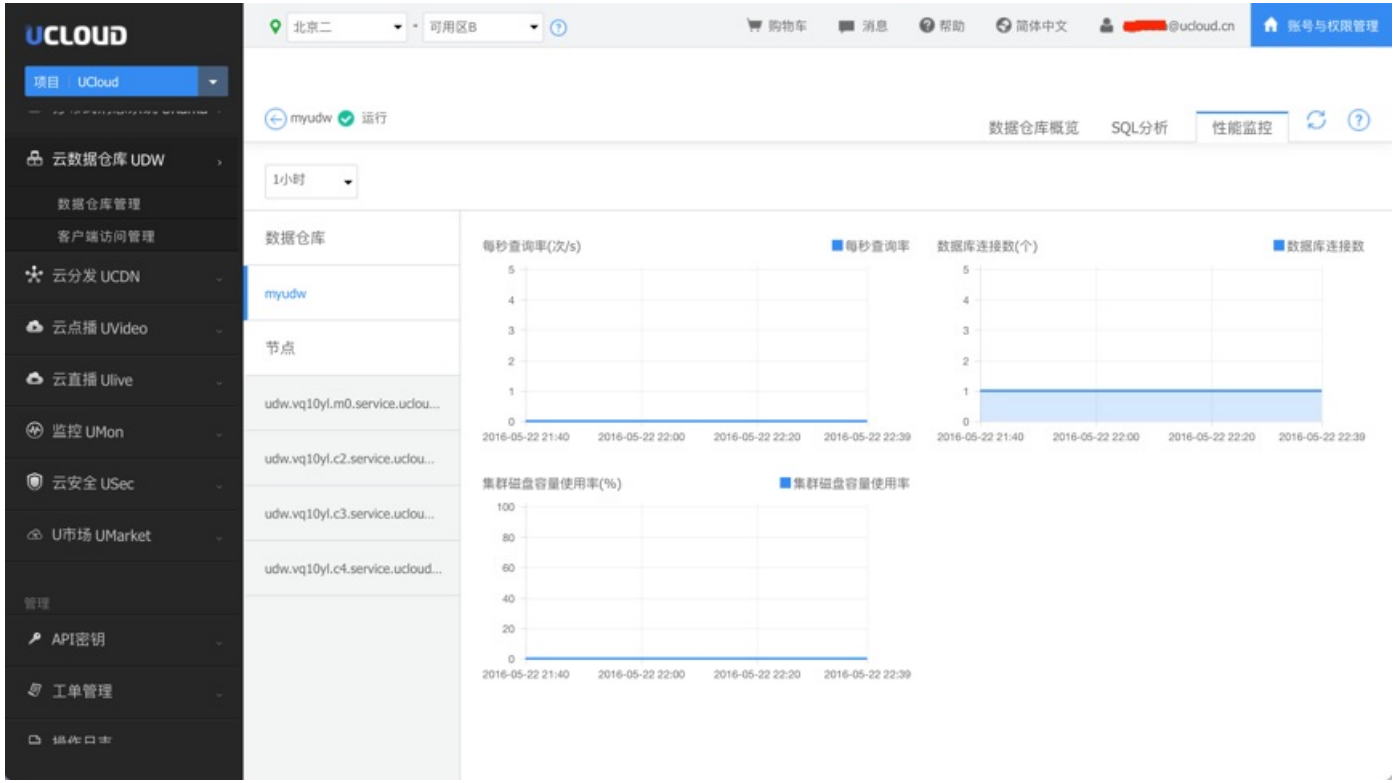


查看监控

Copyright © 2012-2021 UCloud 优刻得



Copyright © 2012-2021 UCloud 优刻得



Copyright © 2012-2021 UCloud 优刻得

访问UDW数据仓库

1 客户端工具访问UDW

udw支持按照postgresql的客户端来访问udw，支持udw客户端访问，还可以支持jdbc、odbc、php、python、psql等方式来访问udw。另外，也可以通过图形化的SQL Workbench/J、Navicat等工具来访问udw。

1.1 psql客户端方式访问

下载psql客户端

```
yum install postgresql.x86_64 (64位系统)

psql -h hostIP（或域名） –U username -d database -p port –W
```

> hostIP：udw master节点的ip或者域名 username: 数据库用户名 database：数据库名称

1.2 udw客户端方式访问

如果你选择是数据仓库类型是greenplum、请用greenplum客户端、如果你选择的数据仓库类型请用udpg客户端。

Copyright © 2012-2021 UCloud 优刻得

1.1 udw(greenplum)客户端方式访问(以Centos为例)

1)下载greenplum客户端解压

```
wget <http://udw.cn-bj.ufileos.com/greenplum-client.tar.gz>

tar -zxvf greenplum-client.tar.gz
```

2)配置udw客户端

进入greenplum-client安装目录,编辑 greenplum_client_path.sh

修改UDW_HOME

```
export UDW_HOME= client安装目录(如/root/greenplum-client)
```

3) 使配置生效

在~/.bashrc中添加如下配置

```
source /data/greenplum-client/greenplum_client_path.sh
```

执行

```
source ~/.bashrc
```

备注:/data/greenplum-client是greenplum-client的安装路径

Copyright © 2012-2021 UCloud 优刻得

4）连接数据库

```
psql -h hostIP（或域名）–U username -d database -p port –W
```

1.2 udw（udpg）客户端方式访问（以Centos为例）

1）下载udw客户端

```
wget http://udw.cn-bj.ufileos.com/udw-client.tar
tar xvf udw-client.tar
```

2）配置udw客户端

进入udw-client安装目录，编辑 `udw_client_path.sh`

修改`UDW_CLIENT`:

```
export UDW_CLIENT=client安装目录（如/root/udw-client）
```

3）使配置生效在`~/.bashrc`中添加如下配置

```
source /data/udw-client/udw_client_path.sh

source ~/.bashrc
```

Copyright © 2012-2021 UCloud 优刻得

备注:/data/udw-client是udw-client的安装路径

4）连接数据库

```
psql -h hostIP(或域名）–U username -d database -p port –W
```

1.3 JDBC方式访问

Linux操作系统

```
yum install postgresql-jdbc.noarch –y
```

Windows环境下JDBC驱动,将jar添加到工程的BUILD PATH。

示例程序1,java连接UDW,执行建表,插入操作

PostgreSQLJDBC1.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class PostgreSQLJDBC1 {
public static void main(String args[]) {
Connection c = null;
Statement stmt = null;
```

Copyright © 2012-2021 UCloud 优刻得

```
try {
Class.forName("org.postgresql.Driver");
c = DriverManager.getConnection("jdbc:postgresql://hostIP:port/dbname",”UserName”,”Password”);
stmt = c.createStatement();
String sql = "CREATE TABLE COMPANY " + "(ID INT PRIMARY KEY NOT NULL," + "NAME TEXT NOT NULL," + "AGE INT NOT NULL," + "ADDRESS CHAR(50),"
+ "SALARY REAL)";
c.setAutoCommit(false);
System.out.println("Opened database successfully");
stmt.executeUpdate(sql);
sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " + "VALUES (1, 'Allen', 25 , 'Texas', 15000.00 );";
stmt.executeUpdate(sql);
stmt.close();
c.commit();
c.close();
}
catch (Exception e) {
e.printStackTrace();
System.err.println(e.getClass().getName()+": "+e.getMessage());
System.exit(0);
}
System.out.println("Opened database successfully");
}
}
```

示例程序二:java连接UDW,执行查询操作

Copyright © 2012-2021 UCloud 优刻得

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class PostgreSQLJDBC2 {
public static void main(String[] args) {
Connection c = null;
Statement stmt = null;
try{
Class.forName("org.postgresql.Driver");
c = DriverManager.getConnection("jdbc:postgresql://hostIP:port/dbname",”UserName”,”Password”);
stmt = c.createStatement();
String sql = null;
System.out.println("Opened database successfully");
sql = "SELECT * FROM COMPANY;";
ResultSet res=stmt.executeQuery(sql);
while(res.next()) {
System.out.println(res.getInt(1));
System.out.println(res.getString(2));
System.out.println(res.getInt(3));
System.out.println(res.getString(4));
System.out.println(res.getDouble(5));
}
stmt.close();
```

Copyright © 2012-2021 UCloud 优刻得

```
c.close();
}
catch(Exception e) {
System.err.println( e.getClass().getName()+": "+ e.getMessage() );
System.exit(0);
}
}
}
```

1.4 ODBC方式访问

Linux操作系统:CentOS 6.5 64位

1. 安装 postgresql odbc驱动

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

2. 编辑/etc/odbcinst.ini文件,配置odbc驱动

```
Description = ODBC for PostgreSQL
Driver = /usr/lib/psqlodbc.so
Setup = /usr/lib/libodbcpsqlS.so
Driver64 = /usr/lib64/psqlodbc.so
Setup64 = /usr/lib64/libodbcpsqlS.so
FileUsage = 1
```

Copyright © 2012-2021 UCloud 优刻得

3. 测试ODBC驱动是否安装成功

```
# odbcinst -q -d
[PostgreSQL]
```

如果出现以上输出，代表在这台机器上已成功安装了PostgreSQL的ODBC驱动。

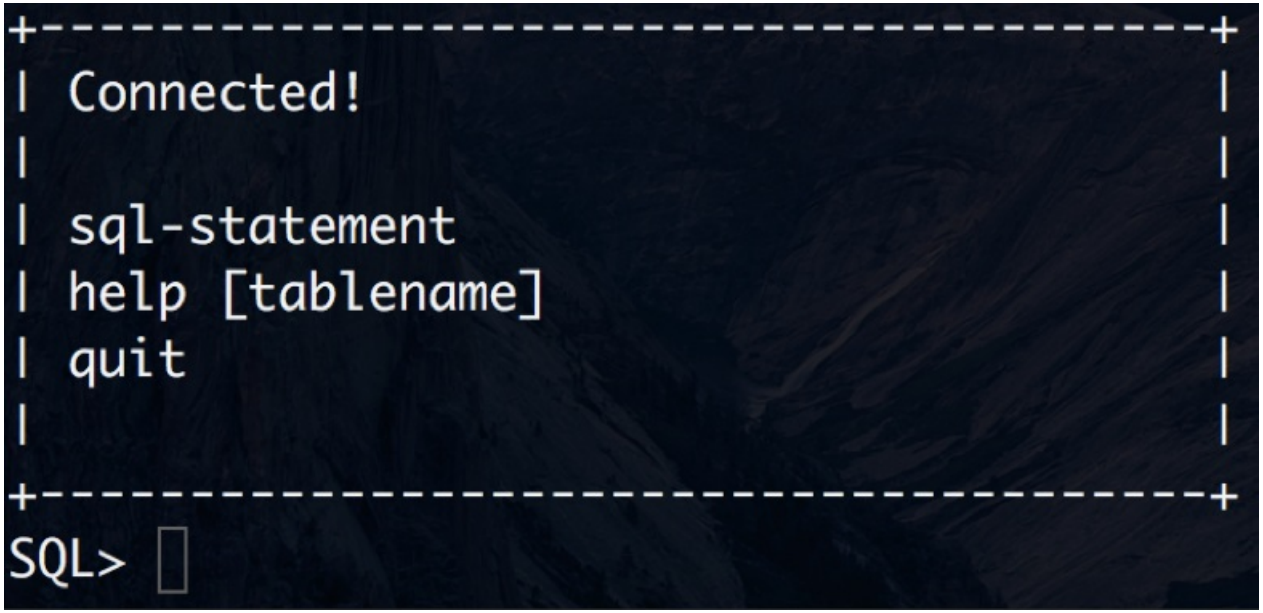
4. 编辑/etc/odbc.ini文件配置ODBC连接

```
[testdb] Description = PostgreSQL connection to TestDB
Driver = PostgreSQL
Database = Database
Servername = MasterNodeIP
UserName = UserName
Password = Password
Port = Port
Protocol = 8.3
ReadOnly = No
RowVersioning = NoShow
SystemTables = No
ConnSettings =
```

5. 测试连接

```
# isql testdb
```

Copyright © 2012-2021 UCloud 优刻得



注解：

如出现以上内容，则表示psqlodbc配置成功。

1.5 python客户端访问

```
$yum install python-psycopg2
```

示例1. 连接UDW testconn.py

```
#!/usr/bin/python
```

Copyright © 2012-2021 UCloud 优刻得

```
import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
```

执行 python testconn.py

示例2. 创建一个表 createTable.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute('''CREATE TABLE COMPANY
(ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR(50),
SALARY REAL);''')
print "Table created successfully"
conn.commit()
conn.close()
```

示例3. 插入记录 insert.py

Copyright © 2012-2021 UCloud 优刻得

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1, 'Paul', 32, 'California', 20000.00 )");
cur.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");
conn.commit()
print "Records created successfully";
conn.close()
```

示例4. 查询 select.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("SELECT id, name, address, salary from COMPANY")
rows = cur.fetchall()
```

Copyright © 2012-2021 UCloud 优刻得

```
for row in rows:
print "ID = ", row[0]
print "NAME = ", row[1]
print "ADDRESS = ", row[2]
print "SALARY = ", row[3], "\n"
print "Operation done successfully";
conn.close()
```

示例5. 更新 update.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("UPDATE COMPANY set SALARY = 25000.00 where ID=1")
conn.commit
print "Total number of rows updated :", cur.rowcount
cur.execute("SELECT id, name, address, salary from COMPANY")
rows = cur.fetchall()
for row in rows:
print "ID = ", row[0]
print "NAME = ", row[1]
print "ADDRESS = ", row[2]
```

Copyright © 2012-2021 UCloud 优刻得

```
print "SALARY = ", row[3], "\n"
print "Operation done successfully";
conn.close()
```

示例6. 删除 delete.py

```
#!/usr/bin/python

import psycopg2
conn = psycopg2.connect(database="dev", user="username", password="password", host="hostIP", port="port")
print "Opened database successfully"
cur = conn.cursor()
cur.execute("DELETE from COMPANY where ID=2;")
conn.commit
print "Total number of rows deleted :", cur.rowcount
cur.execute("SELECT id, name, address, salary from COMPANY")
rows = cur.fetchall()
for row in rows:
print "ID = ", row[0]
print "NAME = ", row[1]
print "ADDRESS = ", row[2]
print "SALARY = ", row[3], "\n"
print "Operation done successfully";
conn.close()
```

Copyright © 2012-2021 UCloud 优刻得

1.6 php客户端访问

```
yum install php-pgsql
```

示例1. 连接 conn.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
?>
```

示例2. 创建表 create.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
```

Copyright © 2012-2021 UCloud 优刻得

```
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
CREATE TABLE COMPANY
(ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR(50),
SALARY REAL);
EOF;
$ret = pg_query($db, $sql);
if(!$ret)
{ echo pg_last_error($db);
} else {
echo "Table created successfullyn";
}
pg_close($db);
?>
```

Copyright © 2012-2021 UCloud 优刻得

示例3. 插入 insert.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00 );
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Allen', 25, 'Texas', 15000.00 );
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
} else {
echo "Records created successfully\n";
}
```

Copyright © 2012-2021 UCloud 优刻得

```
pg_close($db);
?>
```

示例4. 查询 select.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
SELECT * from COMPANY;
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
}
while($row = pg_fetch_row($ret)){
```

Copyright © 2012-2021 UCloud 优刻得

```
echo "ID = ". $row[0] . "\n";
echo "NAME = ". $row[1] ."\n";
echo "ADDRESS = ". $row[2] ."\n";
echo "SALARY = ".$row[4] ."\n\n";
}
echo "Operation done successfully\n";
pg_close($db);
?>
```

示例5. 更新 update.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
SELECT * from COMPANY;
EOF;
```

Copyright © 2012-2021 UCloud 优刻得

```
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
}
while($row = pg_fetch_row($ret)){
echo "ID = ". $row[0] . "\n";
echo "NAME = ". $row[1] ."\n";
echo "ADDRESS = ". $row[2] ."\n";
echo "SALARY = ".$row[4] ."\n\n";
}
echo "Operation done successfully\n";
pg_close($db);
?>
```

示例6. 删除 delete.php

```
<?php
$host = "host=hostIP";
$port = "port=port";
$dbname = "dbname=dbname";
$credentials = "user=user password=password";
$db = pg_connect( "$host $port $dbname $credentials" );
if(!$db){
echo "Error : Unable to open database\n";
```

Copyright © 2012-2021 UCloud 优刻得

```
} else {
echo "Opened database successfully\n";
}
$sql =<<<EOF
DELETE from COMPANY where ID=2;
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
} else {
echo "Record deleted successfully\n";
}
$sql =<<<EOF
SELECT * from COMPANY;
EOF;
$ret = pg_query($db, $sql);
if(!$ret){
echo pg_last_error($db);
exit;
}
while($row = pg_fetch_row($ret)){
echo "ID = ". $row[0] . "\n";
echo "NAME = ". $row[1] ."\n";
echo "ADDRESS = ". $row[2] ."\n";
```

Copyright © 2012-2021 UCloud 优刻得

```
echo "SALARY = ".$row[4] ."\n\n";
}
echo "Operation done successfully\n";
pg_close($db);
?>
```

1.7 node客户端访问

1）安装pg模块

```
npm install -g node_gyp


npm install -g pg
```

2）连接数据库并访问

示例代码如下：

```
var pg = require('pg');
var constring = "tcp://username:password@ip:port/database";
var client = new pg.Client(constring);
client.connect();
client.query("create temp table beatle(name varchar(10),height integer)");
client.query("insert into beatle(name,height) values('john',50)");
client.query("insert into beatle(name,height) values($1,$2)",['brown',68]);
var query = client.query("select * from beatle");
```

Copyright © 2012-2021 UCloud 优刻得

```
query.on('row',function(row){
console.log(row);
});
query.on('end',function(){
client.end();
});
```

如果连接成功，输出：

```
{ name: 'john', height: 50 }

{ name: 'brown', height: 68 }
```

2 图形界面的方式访问UDW

2.1 配置UDW外网访问

udw默认是通过内网访问的，为了数据安全性，尽量不要通过外网访问UDW，如果需要图形界面的方式访问UDW，则需要配置udw的外网访问，请参考：

前提：有一台可以访问 udw 的 uhost，并且这台 uhost 上可以访问外网 ip。

例如：uhost 内网 ip 是 10.10.0.9 外网 IP 是 192.168.120.110，udw 的访问ip 是 10.10.10.1，

我们在 uhost 机器上建立 ssh 隧道即可通过 192.168.120.110访问 udw。在 uhost 机器上执行如下命令：

Copyright © 2012-2021 UCloud 优刻得

```
ssh -C -f -N -g -L 5432:10.10.10.1:5432 root@10.10.0.9
```

备注：请注意开放外网防火墙端口 5432（也可以把 udw 端口映射到 uhost上其他端口上），网络防火墙配置请参考：https://docs.ucloud.cn/unet/firewall/introduction

2.2 SQL Workbench/J

SQL Workbench/J是一个独立于DBMS，跨平台的SQL查询分析工具。具有通用性好、小巧、免安装等优点，

并且功能强大，查询编辑器支持自动补全，Database Explorer可以查看和编辑各种数据库对象（表、视图、存储过程等）。

详情可见：SQL Workbench/J 访问 udw

Copyright © 2012-2021 UCloud 优刻得

数据导入

udw提供了丰富的数据导入方式和工具：

1. 用postgresql的insert和copy方式导入数据到udw
2. 用外部表的方式，把文件并行的导入到udw
3. 创建hdfs的外部表，把hdfs中的数据导入到udw
4. 通过sqoop把hdfs中的数据导入到udw
5. 用mysql2udw把mysql中的数据导入到udw
6. 创建ufile的外部表、把ufile中数据导入到udw

在导入大量的数据的时候我们建议不要使用insert一条条的导入数据、强烈建议使用copy、udwfile导入数据。

insert加载数据

我们可以通过insert插入数据到udw，语法如下所示：

```
INSERT INTO 表名 [ ( 字段 [, ...] ) ]
{ DEFAULT VALUES | VALUES ( { 表达式 | DEFAULT } [, ...] ) | 子查询 }
```

每次插入一条的效率会比较低、我们建议一次插入多条（2000-5000条）数据。如果要加载的数据量比较大的话、强烈建议使用copy方式加载或者我们下面介绍的几种方式加载。如果您的数据已经在udw中，也可以通过`insert into table1 select * from table2`这种方式加载数据。

copy加载数据

我们可以用copy快速加载文件数据到udw。具体语法如下：

Copyright © 2012-2021 UCloud 优刻得

```
cat /data/test.dat | psql -h hostIP -U UserName -d DB -c "COPY employee from STDIN with CSV DELIMITER '|';"
```

hostIP:udw访问id UserName :访问数据的用户名 DB:数据库名称 employee:表名

外部表并行加载数据

外部表并行加载数据是利用http协议实现的一个文件服务器,用于创建udw的外部文件表。使用外部表并行加载数据可以让udw的每个子节点并行的加载数据、大大的加快数据导入udw的速度。在加载数据的时候我们可以先创建一个外部表,然后通过INSERT INTO <table> SELECT * FROM <external_table>。这样就可以并行的加载文件中的数据。

使用方法请参考我们的文档:外部表并行加载数据到udw

从hdfs加载数据

为了方便udw和hdfs之间的数据导入和导出,我们提供个两种方案:

1. 用sqoop实现hdfs和udw直接的数据导入导出,使用方法请参考:hdfs和hive中数据导入导出到udw
2. 创建hdfs外部表,使用方法请参考:创建hdfs外部表

从mysql中导入数据

为了方便mysql数据导入到udw,我们提供了mysql导入数据到udw的工具mysql2udw,使用方法请参考:mysql数据导入到udw

从oracle中导入数据

为了方便oracle数据导入udw,我们提供了oracle导入数据到udw的工具ora2udw,使用方法请参考:oracle数据导入到udw

Copyright © 2012-2021 UCloud 优刻得

从ufile加载数据

为了方便ufile数据导入到udw，我们提供了ufile外部表，导入数据到udw，使用方法请参考：ufile数据导入到udw

Copyright © 2012-2021 UCloud 优刻得

开发指南

1、连接数据库

udw 支持按照 postgresql 方式来访问 udw，可以支持 jdbc、odbc、php、python、psql 等方式来访问 udw。图形化的 pgAdmin、SQL Workbench/J 等工具

1.1 psql 客户端方式访问

下载 psql 客户端（或者通过控制台下载 udw 客户端）

```
yum install postgresql.i686 （32位系统）

yum install postgresql.x86\_64 (64位系统)

psql -h hostIP（或域名） –U username -d database -p port –W
```

- hostIP：udw master节点的ip或者域名
- username：数据库用户名
- database：数据库名称

1.2 udw 客户端方式访问

如果你选择是数据仓库类型是 greenplum、请用 greenplum 客户端、如果你选择的数据仓库类型请用 udpg 客户端。

Copyright © 2012-2021 UCloud 优刻得

1.1 udw(greenplum)客户端方式访问(以Centos为例)

1)下载greenplum客户端解压

```
wget http://udw.cn-bj.ufileos.com/greenplum-client.tar.gz

tar -zxvf greenplum-client.tar.gz
```

2)配置udw客户端

进入 greenplum-client 安装目录,编辑 greenplum-client-path.sh

修改 UDWHOME:

```
export UDWHOME= client安装目录(如/root/greenplum-client)
```

3) 使配置生效

在 ~/.bashrc 中添加如下配置

```
source /data/greenplum-client/greenplum-client-path.sh
```

执行:

```
source ~/.bashrc
```

备注:/data/greenplum-client 是 greenplum-client 的安装路径

Copyright © 2012-2021 UCloud 优刻得

4）连接数据库

```
psql -h hostIP（或域名）–U username -d database -p port –W
```

1.2 udw（udpg）客户端方式访问（以Centos为例）

1）下载 udw 客户端

```
wget http://udw.cn-bj.ufileos.com/udw-client.tar

tar xvf udw-client.tar
```

2）配置 udw 客户端

进入 udw-client 安装目录，编辑 `udw-client-path.sh`

修改 UDWCLIENT:

```
export UDWCLIENT=client安装目录（如/root/udw-client）
```

3）使配置生效在 `~/.bashrc` 中添加如下配置

```
source /data/udw-client/udw-client-path.sh
```

执行：

```
source ~/.bashrc
```

Copyright © 2012-2021 UCloud 优刻得

备注:/data/udw-client 是 udw-client 的安装路径

4）连接数据库

```
psql -h hostIP(或域名) –U username -d database -p port –W
```

1.3 SQL Workbench/J

工具访问udw的文档请参考:http://udw.cn-bj.ufileos.com/SQL%20Workbench%3AJ%20%E8%AE%BF%E9%97%AE%20udw.pdf

2、数据库管理

当你成功连接上数据库后,你可以创建你的第一个数据库(但这不是必须的,你也使用默认创建的数据库来作为你的业务数据库)。下面的操作以 psql 方式连接到 udw 为例。

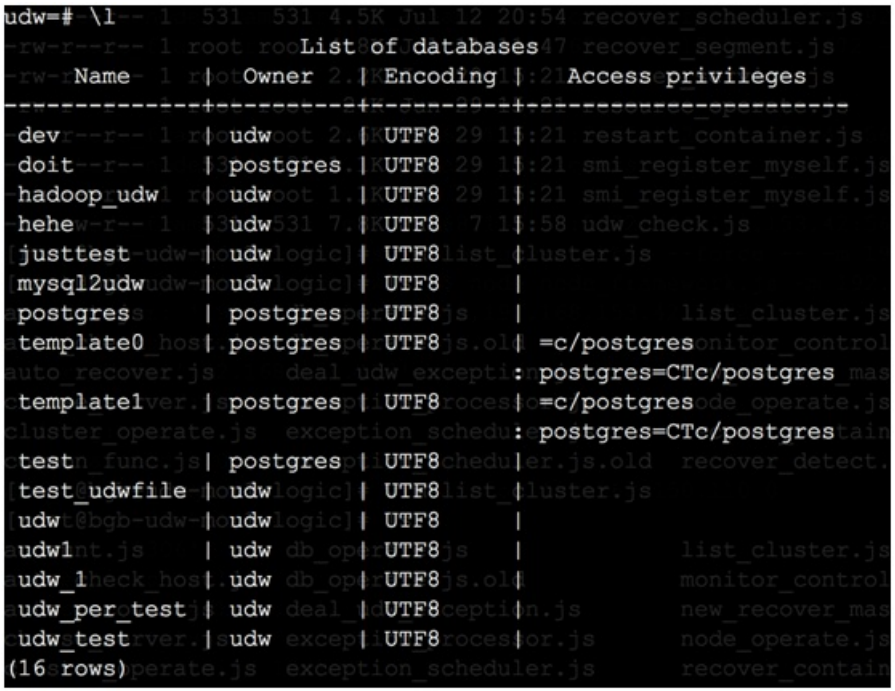
2.1 创建数据库

```
create database product;
```

2.2 查看所有数据库

“l”命令查看

Copyright © 2012-2021 UCloud 优刻得



```
udw=# \l
                              List of databases
    Name    |  Owner  | Encoding |   Access privileges
------------+---------+----------+-----------------------
 dev        | udw     | UTF8     |
 doit       | postgres | UTF8     |
 hadoop_udw | udw     | UTF8     |
 hehe       | udw     | UTF8     |
 justtest   | udw     | UTF8     |
 mysql2udw  | udw     | UTF8     |
 postgres   | postgres | UTF8     |
 template0  | postgres | UTF8     | =c/postgres
                                  : postgres=CTc/postgres
 template1  | postgres | UTF8     | =c/postgres
                                  : postgres=CTc/postgres
 test       | postgres | UTF8     |
 test_udwfile | udw   | UTF8     |
 udw        | udw     | UTF8     |
 udw1       | udw     | UTF8     |
 udw_1      | udw     | UTF8     |
 udw_per_test | udw   | UTF8     |
 udw_test   | udw     | UTF8     |
(16 rows)
```

或者通过下面sql查看

```
select datname from pg\_database; (超级用户)
```

2.3 变更数据库

使用ALTER DATABASE命令，语法如下：

```
ALTER DATABASE name [ [ WITH ] option [ ... ] ]
where option can be:
CONNECTION LIMIT connlimit
```

Copyright © 2012-2021 UCloud 优刻得

```
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
```

2.4 删除数据库

```
c template1 (切换到template1数据库)

DROP DATABASE product;
```

3、模式管理

数据库模式(schema)是包含了一系列数据库对象(表,数据类型,自定义函数)集合的命名容器。一个数据库可以有多个模式。不同模式不共享命名空间。public 模式是在创建数据库之后就会默认创建的,每个用户都有权限在这个 schema 创建对象,如果不指定 schema 那么就会默认创建到这里。

创建一个模式:

```
CREATE SCHEMA testSchema;
```

指定数据库的模式搜索路径:

```
ALTER DATABASE product SET search_path To testSchema,public;
```

Copyright © 2012-2021 UCloud 优刻得

为指定用户指定模式搜索路径：

```
ALTER ROLE roleName SET search_path To testSchema,public;
```

删除空模式：

```
DROP SCHEMA testSchema;
```

删除非空模式：

```
DROP SCHEMA testSchema CASCADE;
```

4、表格设计

udw 的表格创建类似于 postgresql，由于 udw 采用 mpp 数据，创建表格的时候可以选择不同的数据分布策略，不同的存储方式等等。创建表格的时候可以定义下面信息：

- 数据类型
- 表约束
- 数据分布策略
- 表存储模型
- 分区策略
- 外部表：udwfile、udwhdfs

下面分别根据上面的可选信息对表格设计进行分析。

4.1 数据类型

Copyright © 2012-2021 UCloud 优刻得

udw 的数据类型和 postgresql 基本一致，在选择数据类型的时候应该尽可能占用空间小，同时能够保证存储所有可能的数值并且最合理地表达数据。

使用字符型数据类型保存字符串，日期或者日期时间戳类型保存日期类型，数值类型来保存数值。

使用 VARCHAR 或者 TEXT 来保存文本类数据。不推荐使用 CHAR 类型保存文本类型。VARCHAR 或 TEXT 类型对于数据末尾的空白字符将原样保存和处理，但是 CHAR 类型不能满足这个需求。请参考 CREATE TABLE 命令了解更多相关信息。

使用 BIGINT 类型存储 INT 或者 SMALLINT 数值会浪费存储空间。如果数据随时间推移需要扩展，并且数据重新加载比较浪费时间，那么在开始的时候就应该考虑使用更大的数据类型。

4.2 表约束

udw 表格支持 postgresql 的表格约束，拥有 primary、unique 、check、not null、foreign 等约束，主键约束必须使用 hash 策略来分布表数据存储，不能在同一个表同时使用主键和唯一约束，并且指定了primary 和 unique 的列必须全部或者部分包含在分布键中。

创建表检查约束

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

创建非空约束

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

Copyright © 2012-2021 UCloud 优刻得

```
);
```

唯一约束：唯一约束确保存储在一张表中的一列或多列数据数据一定唯一。要使用唯一约束，表必须使用 Hash 分布策略，并且约束列必须和表的分布键对应的列一致（或者是超集）

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

主键约束：主键约束是唯一约束和非空约束的组合。要使用主键约束，表必须使用 Hash 分布策略，并且约束列必须和表的分布键对应的列一致（或者是超集）。如果一张表指定了主键约束，分布键值默认会使用主键约束指定的列。

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

4.3 选择数据分布策略

UDW 表的记录有两种分布策略，分别是哈希分布（DISTRIBUTED BY(key)）和随机分布(DISTRIBUTED RANDOMLY)。如果不指定分布策略则默认按primary key或者第一个column做哈希分布。

为了尽可能的并行处理数据，需要选择能够最大化地将数据均匀分布到所有计算节点的策略，比如选择 primary key；分布式处理中将会存在本地和分布式协作的操作，当不同的表使用相同的分布键的时候，大部分的排序、连接关联操作工作将会在本地完成，本地操作往往比分布式操作快很多，采用随机分布的策略无法享受到这个优势。

Copyright © 2012-2021 UCloud 优刻得

创建一个哈希分布的表：

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

创建一个随机分布的表：

```
CREATE TABLE randomTable (
things text,
content text,
etc text
) DISTRIBUTED RANDOMLY;
```

修改分布策略：

1)分区策略修改为随机分布：

```
alter table test set with (reorganize=true) distributed randomly;
```

2)分区策略修改为按照id的hash分布：

```
alter table test set with (reorganize=true) distributed by (id);
```

Copyright © 2012-2021 UCloud 优刻得

备注:更多关于分区策略的的使用可以通过命令行执行\h create table 或者 \h alter table 查看

4.4 表存储模型（heap表和appendonly表）

UDW 支持两种类型的表:堆表(heap table)和追加表(Appendonly table)。默认创建的是堆表。

堆表(heap table)是最普通的表形式,适合于较小、经常更新的数据存储方式。

追加表(Appendonly table)简称 ao 表,适合大表、updte 比较少的表。

创建一个堆表:

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

创建一个追加表(CREATE TABLE 命令的 WITH 子句来指定表存储模型):

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

4.5 表存储方式（行存储、列存储）

Copyright © 2012-2021 UCloud 优刻得

UDW支持行式存储、列式存储。

行存储的应用场景：

- 表数据在载入后经常 update;
- 表数据经常 insert;
- 查询中选择大部分的列;

列存储的应用场景：

列存储一般适用于宽表（即字段非常多的表）。在使用列存储时，同一个字段的数据连续保存在一个物理文件中，所以列存储的压缩率比普通压缩表的压缩率要高很多，另外在多数字段中筛选其中几个字段中，需要扫描的数据量很小，扫描速度比较快。因此，列存储尤其适合在宽表中对部分字段进行筛选的场景。注意：列存储的表必须是追加表（Appendonly table）。

创建一个行式存储的表

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

创建一个列式存储的表

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

4.6 压缩表

Copyright © 2012-2021 UCloud 优刻得

UDW 压缩表必须是追加表。UDW 支持两种级别的压缩：表级别和字段级别。行式表和列式表对压缩的支持也不一样。

行式表支持表级别的压缩，支持的压缩算法有 ZLIB。

列式表支持表级别和字段级别的压缩，支持的压缩算法有 RLE_TYPE，ZLIB。

RLE_TYPE 的压缩级别 compresslevel 取值从1到4，级别越高压缩比越高。RLE_TYPE适合于有大量重复的数据记录。

ZLIB 的压缩级别 compresslevel 取值从1到9，一般选择5已经足够了。

压缩表的应用场景：业务上对表进行更新和删除操作比较少，用 truncate＋delete 就可以实现业务逻辑。不经常对表进行加字段或修改字段类型，对 ao 表加字段比普通表慢很多。

创建一个使用 ZLIB 压缩的行压缩表：

```
CREATE TABLE rowCompressTable(
a int,
b text
) WITH (appendonly=true,orientation=column,compresstype=ZLIB,compresslevel=5);
```

创建一个使用 RLE_TYPE 压缩的列压缩表

```
CREATE TABLE colCompressTable(
c1 int,
c2 char,
c3 char
) WITH (appendonly=true, orientation=column, compresstype=RLE_TYPE,compresslevel=2);
```

4.7 外部表

Copyright © 2012-2021 UCloud 优刻得

外部表可以方便 udw 加载外部文件或者外部系统文件。详细使用请参考

外部表:外部表并行加载数据到udw

hdfs外部表:创建hdfs外部表

ufile外部表:创建ufile外部表

4.8 变更表

我们可以通过 ALTER TABLE 语句来更改一张表的定义,包括列的定义、数据分布策略、存储模型和分区结构。

给表中的某一列增加非空约束:

```
ALTER TABLE test ALTER COLUMN street SET NOT NOT NULL;
```

改变表的数据分布策略

```
ALTER TABLE test SET DISTRIBUTED BY (id);
```

其他更多可以通过执行\h ALTER TABLE查看帮助。

4.9 删除/清空表

删除表格:

```
DROP TABLE test;
```

Copyright © 2012-2021 UCloud 优刻得

清空表数据：

```
DELETE FROM test1;
TRUNCATE test2;
```

5、加载数据

udw 提供了丰富的数据加载方式和工具：

- 用 postgresql 的 insert 和 copy 方式导入数据到 udw
- 用外部表的方式，把文件并行的导入到 udw
- 创建 hdfs 的外部表，导入导出数据到 hdfs
- 通过 sqoop 把 hdfs 中的数据导入到 udw
- 用 mysql2udw 把 mysql 中的数据导入到 udw
- 创建 ufile 的外部表、导入导出数据到 ufile
- 通过外部表导入 json 格式的数据

在导入大量的数据的时候我们建议不要使用 insert 一条条的导入数据、强烈建议使用 copy、udwfile 导入数据。

5.1 insert加载数据

我们可以通过insert插入数据到udw，语法如下所示：

```
INSERT INTO 表名 [ ( 字段 [, ...] ) ] { DEFAULT VALUES | VALUES ( { 表达式 | DEFAULT } [, ...] ) | 子查询 }
```

每次插入一条的效率会比较低、我们建议一次插入多条（500-5000条）数据。如果要加载的数据量比较大的话、强烈建议使用 copy 方式加载或者我们下面介绍的几种方式加载。如果您的数据已经在 udw 中，也可以通过 `insert into table1 select * from table2` 这种方式加载数据。

Copyright © 2012-2021 UCloud 优刻得

5.2 copy加载数据

我们可以用copy快速加载文件数据到udw。具体语法如下：

```
cat /data/test.dat | psql -h hostIP -U UserName -d DB -c "COPY employee from STDIN with CSV DELIMITER '|';"
```

- hostIP：udw访问id
- UserName ：访问数据的用户名
- DB：数据库名称
- employee：表名

5.3 外部表并行加载数据

外部表并行加载数据是利用 http 协议实现的一个文件服务器，用于创建 udw 的外部文件表。使用外部表并行加载数据可以让 udw 的每个子节点并行的加载数据、大大的加快数据导入 udw 的速度。在加载数据的时候我们可以先创建一个外部表，然后通过 INSERT INTO <table> SELECT * FROM <external_table>。这样就可以并行的加载文件中的数据。

使用方法请参考我们的文档：外部表并行加载数据到udw

5.4 从hdfs加载数据

为了方便 udw 和 hdfs 之间的数据导入和导出，我们提供个两种方案；

1. 用 sqoop 实现 hdfs 和 udw 直接的数据导入导出，使用方法请参考：hdfs和hive中数据导入导出到udw
2. 创建 hdfs 外部表，使用方法请参考：创建hdfs外部表

5.5 从mysql加载数据

为了方便 mysql 数据导入到 udw，我们提供了 mysql 导入数据到 udw 的工具 mysql2udw，使用方法请参考：mysql数据导入到udw

Copyright © 2012-2021 UCloud 优刻得

5.6 从oracle中导入数据

为了方便 oracle 数据导入 udw，我们提供了 oracle 导入数据到 udw 的工具 ora2udw,使用方法请参考：oracle数据导入到udw

5.7 从ufile加载数据

为了方便 ufile 数据导入到 udw，我们提供了 ufile 外部表，导入数据到 udw，使用方法请参考：ufile数据导入到udw

6、分区表

分区表在逻辑上把一个大表切割成小表，分区表可以优化查询性能、在查询的时候只查询部分分区的内容。另外分区表可以很方便的让数据仓库把一些比较老的数据移出数据仓库。

目前udw支持的分区表类型有：

- range分区：把数据根据指定的范围进行分区，例如：时间范围、数值范围
- list分区：把数据按照一个list的值进行分区，例如：产品的种类、地区

使用分区表的场景：

- 数据表足够大：大表格是比较适合做分区的、如果你的表格有上亿行或者更多的的数据，可以通过分区把数据通过分区分为很多小的部分、从而提高性能。如果一个表只有几千行和几万行就不需要再做分区。
- 查询模式固定：例如你经常按照日期去查找表格数据、我们可以按照每月或者每天做分区；如果你需要按照地区去访问数据，我们可以按照地区去做分区。
- 数据仓库保留一个时间窗口的数据：例如您数据仓库需要保留一年的数据、如果按月做分区、可以通过分区很方便的删除最早的月份分区、把数据加载到最新的月份分区。
- 把数据分为几个均等的部分：通过一个分区标准把一个大表的数据划分为均等的分区，这样可以等倍的提高查询性能。

使用分区的时候请避免建立过多的分区，创建过多的分区可能会影响管理和维护作业，例如： 清理工作，节点恢复，集群扩展，查看磁盘使用情况等。

Copyright © 2012-2021 UCloud 优刻得

6.1 创建分区表

创建分区变需要注意以下问题：

- 确定分区策略：按照日期分区、按照数值分区、按照一个列表值分区
- 选择需要做分区的列
- 选择创建表格方式（heap表 ，append表、按列存储的表）

6.1.1 按照日期创建分区表

日期划分的分区表使用一个日期或时间戳列做为分区键值列。可以按天或者按月进行分析。

您可以通过指定起始值（START），终止值（END）和增量子句（EVERY）指出分区的增量值，让 UDW 数据仓库来自动地生成分区。默认情况下，起始值总是包含的（闭区间），而终止值是排除的（开区间）。例如：

场景一：默认创建的分区表是heap表

```
CREATE TABLE p_store_sales(
id int,
date date,
prices decimal(7,2)
) DISTRIBUTED BY (id)
PARTITION BY RANGE (date) (
START (date '2016-01-01') INCLUSIVE
END (date '2017-01-01') EXCLUSIVE
EVERY (INTERVAL '1 day')
);
```

Copyright © 2012-2021 UCloud 优刻得

场景二:创建appendonly的分区表

```
CREATE TABLE p_store_sales(
id int,
date date,
prices decimal(7,2)
) with (APPENDONLY=true ,compresslevel=5) DISTRIBUTED BY (id)
PARTITION BY RANGE (date) (
START (date '2016-01-01') INCLUSIVE
END (date '2017-01-01') EXCLUSIVE
EVERY (INTERVAL '1 day')
);
```

场景三:创建列存储的分区表

```
CREATE TABLE p_store_sales(
id int,
date date,
prices decimal(7,2)
) with (APPENDONLY=true,ORIENTATION=column,compresslevel=5) DISTRIBUTED BY (id)
PARTITION BY RANGE (date) (
START (date '2016-01-01') INCLUSIVE
END (date '2017-01-01') EXCLUSIVE
EVERY (INTERVAL '1 day')
);
```

Copyright © 2012-2021 UCloud 优刻得

场景四:创建带默认分区的分区表(一般情况不建议创建默认分区)

```
CREATE TABLE p_store_sales(
id int,
date date,
prices decimal(7,2)
) DISTRIBUTED BY (id)
PARTITION BY RANGE (date) (
START (date '2016-01-01') INCLUSIVE
END (date '2017-01-01') EXCLUSIVE
EVERY (INTERVAL '1 day'),
DEFAULT PARTITION error
);
```

如果输入的数据不满足分区的 CHECK 约束条件,并且没有创建默认分区,数据将被拒绝插入。默认分区能够保证在输入数据不满足分区时,能够将数据插入到默认分区。

场景五:为每个分区指定独立的名:

```
CREATE TABLE p_store_sales(
id int,
date date,
prices decimal(7,2)
) DISTRIBUTED BY (id)
PARTITION BY RANGE (date) (
PARTITION Sep08 START (date '2016-09-01') INCLUSIVE ,
```

Copyright © 2012-2021 UCloud 优刻得

```
PARTITION Oct08 START (date '2016-10-01') INCLUSIVE ,
PARTITION Nov08 START (date '2016-11-01') INCLUSIVE ,
PARTITION Dec08 START (date '2016-12-01') INCLUSIVE
END (date '2017-01-01') EXCLUSIVE
);
```

除了最后一个分区外，其他分区的终止值不用指出。

6.1.2 按照数值范围创建分区表

使用数值范围的分区表，利用单独的数值类型列做为分区键值列。例如下面根据年龄范围做分区：

```
CREATE TABLE p_members(
id int,
date date,
age int,
region char(1)
) DISTRIBUTED BY (id)
PARTITION BY RANGE (age) (
START (1) END (100) EVERY (1),
DEFAULT PARTITION extra
);
```

6.1.3 按照列表值创建分区表

Copyright © 2012-2021 UCloud 优刻得

使用列表值进行分区的表可以选用任何支持等值比较的数据类型列做为分区键值列。对于列表值分区来说，您必须为每一个要创建的分区指定对应的列表值。例如下面根据地区进行分区：

```
CREATE TABLE p_members(
id int,
date date,
age int,
region char(2)
) DISTRIBUTED BY (id)
PARTITION BY LIST (region) (
PARTITION shanghai VALUES ('SH'),
PARTITION beijing VALUES ('BG'),
DEFAULT PARTITION other
);
```

6.2查看分区表信息

通过 pg_partitions 视图，您可以查看分区表设计信息。下面示例可以查看 p_store_sales 表的分区设计信息：

```
SELECT partitionboundary, partitiontablename, partitionname, partitionlevel, partitionrank
FROM pg_partitions WHERE tablename='p_store_sales';
```

备注：

- pg_partition：跟踪分区表及其继承关系信息。
- pg_partition_templates：创建子分区使用的子分区模版信息。
- pg_partition_columns：分区表分区键值信息。

Copyright © 2012-2021 UCloud 优刻得

- partitionrank：分区表的rank值，删除分区表的时候需要用这个值，详情请参考删除分区表

6.3 加载数据分区表

在创建了分区表结构后，父表里面是没有数据的。数据自动地存储到最底层的子分区中。

如果记录不满足任何子分区表的要求，插入将会被拒绝，数据加载都会失败。要避免不合要求的记录在加载时被拒绝导致的失败，可以在定义分区结构时，创建一个默认分区（DEFAULT）。任何不满足分区 CHECK 约束记录都会被加载到默认分区。

6.4 修改分区表

1. 通过修改父表来改变子表名称

```
ALTER TABLE p_store_sales RENAME TO c_store_sales;
```

2. 通过修改指定分区名称，来更加便捷的识别子分区

```
ALTER TABLE p_store_sales RENAME PARTITION FOR ('2016-01-01') TO part1;
```

6.5 增加分区/增加默认分区

增加分区：

您可以通过 ALTER TABLE 命令向已有的分区表中添加新的分区，例如：

```
ALTER TABLE p_store_sales
ADD PARTITION
```

Copyright © 2012-2021 UCloud 优刻得

```
START (date '2017-02-01') INCLUSIVE
END (date '2017-03-01') EXCLUSIVE
with(APPENDONLY=true,ORIENTATION=column,compresslevel=5);
```

增加默认分区(一般不建议使用默认分区):

```
ALTER TABLE p_store_sales ADD DEFAULT PARTITION other;
```

如果输入的数据不满足分区的 CHECK 约束条件,并且没有创建默认分区,数据将被拒绝插入。默认分区能够保证在输入数据不满足分区时,能够将数据插入到默认分区。

如果分区表中包含默认分区,您必须通过分裂默认分区的方式来增加新的分区。在使用 INTO 子句时,需要将默认分区做为第二个分区名称。例如:

```
ALTER TABLE p_store_sales
SPLIT DEFAULT PARTITION
START ('2016-01-01') INCLUSIVE
END ('20016-02-01') EXCLUSIVE
INTO (PARTITION new_part, default partition);
```

如果是按照列值创建的分区表,需要通过split default partition实现

```
ALTER TABLE p_members
SPLIT DEFAULT PARTITION at ('GZ') into (PARTITION guangzhou, DEFAULT PARTITION);
```

6.6 删除清空分区

删除分区

Copyright © 2012-2021 UCloud 优刻得

```
ALTER TABLE p_store_sales DROP PARTITION FOR (RANK(1));
```

清空分区：

```
ALTER TABLE p_store_sales TRUNCATE PARTITION FOR (RANK(1));
```

备注：RANK括号里面的1是分区的rank值、可以通过上述查看分区信息查看，增加或者减少分区都可能影响rank值。

6.7 把一个分区分为两个分区

使用 ALTER TABLE 命令来把一个分区分为两个分区

```
ALTER TABLE p_store_sales SPLIT PARTITION FOR ('2016-01-01')
AT ('2016-01-16')
INTO (PARTITION part_001, PARTITION part_002);
```

如果您的分区表中包含默认分区，您必须通过分裂默认分区的方式来增加新的分区。在使用 INTO 子句时，需要将默认分区做为第二个分区名称。例如：

```
ALTER TABLE p_store_sales SPLIT DEFAULT PARTITION
START ('2016-01-01') INCLUSIVE
END ('20016-02-01') EXCLUSIVE
INTO (PARTITION new_part, default partition);
```

6.8 分区表索引管理

Copyright © 2012-2021 UCloud 优刻得

分区表对父表创建索引、分区表会自动增加索引，新增的分区也会自动添加索引。

例如：

```
create index date_index on p_store_sales (date);
```

7、序列

通过使用序列，系统可以在新的纪录插入表中时，自动地按照自增方式分配一个唯一ID。使用序列一般就是为插入表中的纪录自动分配一个唯一标识符。您可以通过声明一个 SERIAL 类型的标识符列，该类型将会自动创建一个序列来分配 ID。

创建序列

```
CREATE SEQUENCE myid START 0;
```

使用序列

```
INSERT INTO test VALUES (nextval('myid'), 'test');
```

修改序列

```
ALTER SEQUENCE myid RESTART WITH 88;
```

删除序列

Copyright © 2012-2021 UCloud 优刻得

```
DROP SEQUENCE myid;
```

8、索引

udw 支持B-tree、位图索引(bitmap)

默认创建的是 B-tree 索引;唯一索引必须包含在分布键中(可以是全部或者部分列,在第1个索引中可以部分列,之后必须全部列),唯一索引不支持 ao 表,唯一索引不会跨越分区起作用,只是针对独立的分区;位图索引比普通索引占用更小的空间,位图索引不应在经常更新的表中使用。

btree索引

```
CREATE UNIQUE INDEX test_index ON t1 (id);
```

位图索引

```
CREATE INDEX test_bmp_index ON t2 USING bitmap(id);
```

更改索引:

```
Alter INDEX name RENAME TO new_name;
```

查看表的索引信息

```
select * from pg_indexes where tablename='t1';
```

Copyright © 2012-2021 UCloud 优刻得

检查索引使用

查看 EXPLAIN 输出中出现的提示：

```
EXPLAIN select id, name from t1 where id = 100;
```

如有 Index Scan 或者 Bitmap Heap Scan 或者 Bitmap Index Scan, 则使用了索引。

删除索引：

```
drop index test_index;
```

关于建索引的一些建议：

- 不要在频繁更新的字段上建索引
- 索引列通常用来做join
- 批量导入数据需先删除索引，等数据导完后再重建，这样会更快
- 索引列经常被频繁使用在where语句中

9、 ANALYZE/VACUUM

ANALYZE：收集数据库的统计信息。

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

VACUUM：数据库磁盘垃圾回收并收集统计信息。

Copyright © 2012-2021 UCloud 优刻得

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

10、常用SQL大全

1. psql客户端常用

```
\h 获取SQL命令的帮助
\? 获取psql命令的帮助
\q 退出
```

2. 一般选项

```
\c [数据库名]－[用户名] 连接到新数据库
\cd [目录名] 改变当前的工作目录
\encoding [编码] 显示或设置客户端编码
\h [名字] SQL命令的语法帮助
\set [名字 [值]] 设置内部变量
\timing 查询计时开关切换(默认关闭)
\unset 名字 取消(删除)内部变量
```

3. 查询缓冲区选项

Copyright © 2012-2021 UCloud 优刻得

```
\e [文件名] 用一个外部编辑器编辑当前查询缓冲区或文件
\g [文件名] 向服务器发送SQL命令
\p 显示当前查询缓冲区的内容
\r 重置 (清理) 查询缓冲区
\s [文件名] 打印历史或者将其保存到文件
\w [文件名] 将查询缓冲区写出到文件
```

4. 输入／输出选项

```
\echo [字串] 向标准输出写出文本 /i 文件名 执行来自文件的命令
\o [文件名] 向文件或者 |管道 发送所有查询结果
```

5. 信息选项

```
\d [名字] 描述表, 索引, 序列, 或者视图
\d{t|i|s|v|S} [模式] (加 "+" 获取更多信息)  列出表/索引/序列/视图/系统表
\da [模式] 列出聚集函数
\db [模式] 列出表空间 (加 "+" 获取更多的信息)
\dc [模式] 列出编码转换
\dC 列出类型转换
\dd [模式] 显示目标的注释
\df [模式] 列出函数 (加 "+" 获取更多的信息)
\dg [模式] 列出组
\dn [模式] 列出模式 (加 "+" 获取更多的信息)
```

Copyright © 2012-2021 UCloud 优刻得

\do [名字] 列出操作符

\dl 列出大对象, 和 lo_list 一样

\dp [模式] 列出表, 视图, 序列的访问权限

\dT [模式] 列出数据类型 (加 "+" 获取更多的信息)

\du [模式] 列出用户

\l 列出所有数据库 (加 "+" 获取更多的信息)

\z [模式] 列出表, 视图, 序列的访问权限 (和 dp 一样)

6. 格式选项

\a 在非对齐和对齐的输出模式之间切换

\C [字串] 设置表标题, 如果参数空则取消标题

\f [字串] 为非对齐查询输出显示或设置域分隔符

\H 在 HTML 输出模式之间切换 (当前是 关闭)

\pset 变量 [值] 设置表的输出选项

\t 只显示行 (当前是 关闭)

\T [字串] 设置 HTML <表> 标记属性, 如果没有参数就取消设置

\x 在扩展输出之间切换 (目前是 关闭)

12、常用SQL命令

命令:ABORT

描述: 终止当前事务

Copyright © 2012-2021 UCloud 优刻得

语法:

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

命令: ALTER AGGREGATE

描述: 改变一个聚集函数的定义

语法:

```
ALTER AGGREGATE 名字 ( 类型 ) RENAME TO 新名字
```

命令: ALTER DATABASE

描述: 改变一个数据库

语法:

```
ALTER DATABASE 名字 SET 参数 { TO | = } { 值 | DEFAULT }
ALTER DATABASE 名字 RESET 参数
ALTER DATABASE 名字 RENAME TO 新名字
ALTER DATABASE 名字 OWNER TO 新属主
```

命令: ALTER FUNCTION

描述: 改变一个函数的定义

语法:

Copyright © 2012-2021 UCloud 优刻得

```
ALTER FUNCTION 名字 ( [ 类型 [, ...] ] ) RENAME TO 新名字
ALTER FUNCTION 名字 ( [ 类型 [, ...] ] ) OWNER TO 新属主
```

命令: ALTER GROUP

描述: 改变一个用户组

语法:

```
ALTER GROUP 组名称 ADD USER 用户名称 [, ... ]
ALTER GROUP 组名称 DROP USER 用户名称 [, ... ]
ALTER GROUP 组名称 RENAME TO 新名称
```

命令: ALTER INDEX

描述: 改变一个索引的定义

语法:

```
ALTER INDEX name RENAME TO new_name
ALTER INDEX name SET TABLESPACE tablespace_name
ALTER INDEX name SET (storage_parameter = value )
ALTER INDEX name RESET (storage_parameter [, ... ])
```

命令: ALTER SCHEMA

描述: 改变一个模式的定义

语法:

Copyright © 2012-2021 UCloud 优刻得

```
ALTER SCHEMA 名字 RENAME TO 新名字
ALTER SCHEMA 名字 OWNER TO 新属主
```

命令: ALTER SEQUENCE

描述: 改变一个序列生成器的定义

语法:

```
ALTER SEQUENCE 名字 [ INCREMENT [ BY ] 递增 ] [ MINVALUE 最小值 | NO MINVALUE ] [ MAXVALUE 最大值 | NO MAXVALUE ] [ RESTART [ WITH ] 开始 ] [ CACHE 缓存 ] [ [ NO ] CYCLE ] [OWNED BY {表名.列名 | NONE}] ALTER SEQUENCE name SET SCHEMA new_schema
```

命令: ALTER TYPE

描述: 改变一个类型的定义语法:

```
ALTER TYPE 名字 OWNER TO 新属主
```

命令: ALTER USER

描述: 改变一个数据库用户

语法:

```
ALTER USER name RENAME TO newname
ALTER USER name SET parameter { TO | = } { value | DEFAULT }
ALTER USER name RESET parameter
ALTER USER name [ [ WITH ] option [ ... ] ]  where option can be:   CREATEDB | NOCREATEDB    | CREATEUSER | NOCREATEUSER    | [ ENCRYPTED |
```

Copyright © 2012-2021 UCloud 优刻得

```
UNENCRYPTED ] PASSWORD 'password'    | VALID UNTIL 'abstime'
```

命令: ANALYZE

描述: 收集关于数据库的统计信息

语法:

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

命令: BEGIN

描述: 开始一个事务块

语法:

```
BEGIN [ WORK | TRANSACTION ] [ 事务模式] [ READ WRITE | READ ONLY  ]
事务模式为下面之一:
ISOLATION LEVEL { SERIALIZABLE | READ COMMITTED | READ UNCOMMITTED }
```

命令: CLOSE

描述: 关闭一个游标

语法: CLOSE 名字

命令: COMMIT

描述: 提交当前事务

Copyright © 2012-2021 UCloud 优刻得

语法: COMMIT [WORK | TRANSACTION]

命令: COPY

描述: 在一个文件和一个表之间拷贝数据

语法:

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

命令: CREATE AGGREGATE

描述: 定义一个新的聚集函数

语法:

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

命令: CREATE CAST

描述: 定义一个新的类型转换

语法:

Copyright © 2012-2021 UCloud 优刻得

```
CREATE CAST (源类型 AS 目标类型)  WITH FUNCTION 函数名 (参数类型)  [ AS ASSIGNMENT | AS IMPLICIT ] CREATE CAST (源类型 AS 目标类型)  WITHOUT FUNCTION  [ AS ASSIGNMENT | AS IMPLICIT ]
```

命令: CREATE DATABASE

描述: 创建一个新的数据库

语法:

```
CREATE DATABASE 数据库名称  [ [ WITH ] [ OWNER [=] 数据库属主 ]  [ TEMPLATE [=] 模板 ]  [ ENCODING [=] 编码 ]  [ TABLESPACE [=] 表空间 ] ]
```

命令: CREATE FUNCTION

描述: 定义一个新的函数

语法:

```
CREATE [ OR REPLACE ] FUNCTION 名字 ( [ [ 参数名字 ] 参数类型 [, ...] ] )  RETURNS 返回类型  { LANGUAGE 语言名称  | IMMUTABLE | STABLE | VOLATILE  | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT  | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER  | AS 'definition'  | AS 'obj_file', 'link_symbol'  } ...  [ WITH ( attribute [, ...] ) ]
```

命令: CREATE GROUP

描述: 定义一个新的用户组

语法:

```
CREATE GROUP 组名 [ [ WITH ] option [ ... ] ]  option 可以为:  SUPERUSER | NOSUPERUSER | CREATEDB | NOCREATEDB | CREATEROLE | NOCREATEROLE |
```

Copyright © 2012-2021 UCloud 优刻得

```
CREATEUSER | NOCREATEUSER | INHERIT | NOINHERIT | LOGIN | NOLOGIN | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password' | VALID UNTIL 'timestamp' | IN ROLE rolename [, ...] | IN GROUP rolename [, ...] | ROLE rolename [, ...] | ADMIN rolename [, ...] | USER rolename [, ...] | SYSID uid
```

命令: CREATE INDEX

描述: 定义一个新的索引

语法:

```
CREATE [ UNIQUE ] INDEX 索引名称 ON 表名 [ USING method ]  ( { column | ( expression ) } [ opclass ] [, ...] )  [ TABLESPACE tablespace ]  [ WHERE predicate ]
```

命令: CREATE RULE

描述: 定义一个新的重写规则

语法:

```
CREATE [ OR REPLACE ] RULE 名字 AS ON 事件  TO 表 [ WHERE 条件 ]  DO [ ALSO | INSTEAD ] { NOTHING | 命令 | ( 命令 ; 命令 ... ) }
```

命令: CREATE SCHEMA

描述: 定义一个新的模式

语法:

```
CREATE SCHEMA 模式名称 [ AUTHORIZATION 用户名称 ] [ 模式元素 [ ... ] ]
CREATE SCHEMA AUTHORIZATION 用户名称 [ 模式元素 [ ... ] ]
```

Copyright © 2012-2021 UCloud 优刻得

命令: CREATE SEQUENCE

描述: 定义一个新的序列生成器

语法:

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE name [ INCREMENT [ BY ] increment ]  [ MINVALUE minvalue | NO MINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE ]  [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]  [OWNED BY { table.column | NONE }]
```

命令: CREATE TABLE

描述: 定义一个新的表

语法:

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name (  { column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ] | table_constraint  | LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ] } [, ... ] )  [ INHERITS ( parent_table [, ... ] ) ]  [ WITH OIDS | WITHOUT OIDS ]  [ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]  [ TABLESPACE tablespace ]  where column_constraint is:  [ CONSTRAINT constraint_name ]  { NOT NULL | NULL | UNIQUE [ USING INDEX TABLESPACE tablespace ] |  PRIMARY KEY [ USING INDEX TABLESPACE tablespace ] | CHECK (expression) | REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]  [ ON DELETE action ] [ ON UPDATE action ] }  [ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY  IMMEDIATE ]  and table_constraint is:  [ CONSTRAINT constraint_name ]  { UNIQUE ( column_name [, ... ] ) [ USING INDEX TABLESPACE tablespace ] |  PRIMARY KEY ( column_name [, ... ] ) [ USING INDEX TABLESPACE tablespace ] |  CHECK ( expression ) | FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]  [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] } [ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY  IMMEDIATE ]
```

命令: CREATE TABLE AS

描述: 以一个查询的结果定义一个新的表

Copyright © 2012-2021 UCloud 优刻得

语法:

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE 表名字 [ (字段名字 [, ...] ) ] [ [ WITH | WITHOUT ] OIDS ]  AS query
```

命令: CREATE TYPE

描述: 定义一个新的数据类型

语法:

```
CREATE TYPE name AS ( attribute_name data_type [, ... ] ) CREATE TYPE name ( INPUT = input_function, OUTPUT = output_function [, RECEIVE = receive_function] [, SEND = send_function] [, INTERNALLENGTH = {internallength | VARIABLE}] [, PASSEDBYVALUE] [, ALIGNMENT = alignment] [, STORAGE = storage] [, DEFAULT = default] [, ELEMENT = element] [, DELIMITER = delimiter] ) CREATE TYPE name
```

命令: CREATE USER

描述: 定义一个新的数据库用户帐户

语法:

```
CREATE USER name [ [ WITH ] option [ ... ] ]  where option can be:  SUPERUSER | NOSUPERUSER | CREATEDB | NOCREATEDB | CREATEROLE | NOCREATEROLE | CREATEUSER | NOCREATEUSER | INHERIT | NOINHERIT | LOGIN | NOLOGIN | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password' | VALID UNTIL 'timestamp' | IN ROLE rolename [, ...] | IN GROUP rolename [, ...] | ROLE rolename [, ...] | ADMIN rolename [, ...] | USER rolename [, ...] | SYSID uid | RESOURCE QUEUE queue_name
```

命令: CREATE VIEW 描述: 定义一个新的视图 语法:

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

Copyright © 2012-2021 UCloud 优刻得

命令: DECLARE

描述: 定义一个游标

语法:

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

命令: DELETE

描述: 删除一个表的记录

语法:

```
DELETE FROM [ ONLY ] 表 [ WHERE 条件]
```

命令: DROP AGGREGATE

描述: 删除一个聚集函数

语法:

```
DROP AGGREGATE 名字 ( 类型 ) [ CASCADE | RESTRICT ]
```

命令: DROP CAST

描述: 删除一个类型转换

Copyright © 2012-2021 UCloud 优刻得

语法:

```
DROP CAST (源类型 AS 目标类型) [ CASCADE | RESTRICT ]
```

命令: DROP DATABASE

描述: 删除一个数据库

语法:

```
DROP DATABASE 名字
```

命令: DROP FUNCTION

描述: 删除一个函数

语法:

```
DROP FUNCTION 名字 ( [ 类型 [, ...] ] ) [ CASCADE | RESTRICT ]
```

命令: DROP GROUP

描述: 删除一个用户组

语法:

```
DROP GROUP 名字
```

命令: DROP INDEX

Copyright © 2012-2021 UCloud 优刻得

描述: 删除一个索引

语法:

```
DROP INDEX 名字 [, ...] [ CASCADE | RESTRICT ]
```

命令: DROP RULE

描述: 删除一个重写规则

语法:

```
DROP RULE 名字 ON 关系 [ CASCADE | RESTRICT ]
```

命令: DROP SCHEMA

描述: 删除一个模式

语法:

```
DROP SCHEMA 名字 [, ...] [ CASCADE | RESTRICT ]
```

命令: DROP SEQUENCE

描述: 删除一个序列

语法:

```
DROP SEQUENCE 名字 [, ...] [ CASCADE | RESTRICT ]
```

Copyright © 2012-2021 UCloud 优刻得

命令: DROP TABLE

描述: 删除一个表

语法:

```
DROP TABLE 名字 [, ...] [ CASCADE | RESTRICT ]
```

命令: DROP TYPE

描述: 删除一个数据类型

语法:

```
DROP TYPE 名字 [, ...] [ CASCADE | RESTRICT ]
```

命令: DROP USER

描述: 删除一个数据库用户帐户

语法:

```
DROP USER 名字
```

命令: DROP VIEW

描述: 删除一个视图

语法:

Copyright © 2012-2021 UCloud 优刻得

```
DROP VIEW 名字 [, ...] [ CASCADE | RESTRICT ]
```

命令: END

描述: 提交当前事务

语法:

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

命令: EXECUTE

描述: 执行一个准备好的语句

语法:

```
EXECUTE 规划名称 [ (参数 [, ...] ) ]
```

命令: EXPLAIN

描述: 显示语句的执行规划

语法:

```
EXPLAIN [ ANALYZE ] [ VERBOSE ] 语句
```

命令: FETCH

描述: 恢复来自一个使用游标查询的行

Copyright © 2012-2021 UCloud 优刻得

语法:

```
FETCH [ direction { FROM | IN } ] cursorname  direction 可以为空或下面的一种:  NEXT | FIRST | LAST | ABSOLUTE count | RELATIVE count | count | ALL | FORWARD | FORWARD count | FORWARD ALL
```

命令: GRANT

描述: 定义访问权限

语法:

```
IGRAN { {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
```

命令: INSERT

描述: 在一个表中插入记录

语法:

```
INSERT INTO 表名 [ ( 字段 [, ...] ) ] { DEFAULT VALUES | VALUES ( { 表达式 | DEFAULT } [, ...] ) | 子查询 }
```

Copyright © 2012-2021 UCloud 优刻得

命令: LOCK

描述: 锁定一个表

语法:

```
LOCK [ TABLE ] 名字 [, ...] [ IN lockmode MODE ] [ NOWAIT ]
lockmode 可以是下面的一种:
ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE
```

命令: PREPARE

描述: 为执行准备一条语句

语法:

```
PREPARE 规划名称 [ (数据类型 [, ...] ) ] AS 语句
```

命令: REINDEX

描述: 重建索引

语法:

```
REINDEX { DATABASE | TABLE | INDEX } 名字 [ FORCE ]
```

命令: RELEASE SAVEPOINT

描述: 删除一个以前定义的 savepoint

Copyright © 2012-2021 UCloud 优刻得

语法:

```
RELEASE [ SAVEPOINT ] savepoint_name
```

命令: RESET

描述: 恢复运行时参数值为默认值

语法:

```
RESET 名字
RESET ALL
```

命令: REVOKE 描述: 删除访问权限 语法:

```
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]
```

命令: ROLLBACK

Copyright © 2012-2021 UCloud 优刻得

描述: 终止当前事务

语法:

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

命令: ROLLBACK TO SAVEPOINT

描述: 回滚到一个 savepoint

语法:

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

命令: SAVEPOINT

描述: 在当前事物中定义一个新的 savepoint

语法:

```
SAVEPOINT savepoint_name
```

命令: SELECT

描述: 从表或视图中查询记录

语法:

```
SELECT [ALL | DISTINCT [ON (expression [, ...])]] * | expression [[AS] output_name] [, ...] [FROM from_item [, ...]] [WHERE condition] [GROUP BY
```

Copyright © 2012-2021 UCloud 优刻得

```
grouping_element [, ...]] [HAVING condition [, ...]] [WINDOW window_name AS (window_specification)] [{UNION | INTERSECT | EXCEPT} [ALL] select] [ORDER BY expression [ASC | DESC | USING operator] [, ...]] [LIMIT {count | ALL}] [OFFSET start] [FOR {UPDATE | SHARE} [OF table_name [, ...]] [NOWAIT] [...]]
where grouping_element can be one of: () expression ROLLUP (expression [,...]) CUBE (expression [,...]) GROUPING SETS ((grouping_element [, ...])) where window_specification can be: [window_name] [PARTITION BY expression [, ...]] [ORDER BY expression [ASC | DESC | USING operator] [, ...] [{RANGE | ROWS} { UNBOUNDED PRECEDING | expression PRECEDING | CURRENT ROW | BETWEEN window_frame_bound AND window_frame_bound }]] where window_frame_bound can be one of: UNBOUNDED PRECEDING expression PRECEDING CURRENT ROW expression FOLLOWING UNBOUNDED FOLLOWING
where from_item can be one of: [ONLY] table_name [[AS] alias [( column_alias [, ...] )]] (select) [AS] alias [( column_alias [, ...] )] function_name ( [argument [, ...]] ) [AS] alias [( column_alias [, ...] | column_definition [, ...] )] function_name ( [argument [, ...]] ) AS ( column_definition [, ...] ) from_item [NATURAL] join_type from_item [ON join_condition | USING ( join_column [, ...] )]
```

命令: SELECT INTO

描述: 以一个查询的结果定义一个新的表

语法:

```
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] [, ...]] [LIMIT {count | ALL}] [OFFSET start] [FOR {UPDATE | SHARE} [OF table_name [, ...]] [NOWAIT] [...]]
```

命令: SET

描述: 改变一个运行时参数

语法:

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

Copyright © 2012-2021 UCloud 优刻得

命令: SET TRANSACTION

描述: 设置当前事务的属性

语法:

```
SET TRANSACTION [transaction_mode] [READ ONLY | READ WRITE] SET SESSION CHARACTERISTICS AS TRANSACTION transaction_mode [READ ONLY | READ WRITE]
事务模式为其中之一:
ISOLATION LEVEL {SERIALIZABLE | READ COMMITTED | READ UNCOMMITTED}
```

命令: SHOW

描述: 显示运行时参数值

语法:

```
SHOW 参数名
SHOW ALL
```

命令: START TRANSACTION

描述: 开始一个事务块

语法:

```
START TRANSACTION [ 事务模式 [, ...] ]
事务模式为下面之一:
```

Copyright © 2012-2021 UCloud 优刻得

```
ISOLATION LEVEL { SERIALIZABLE | READ COMMITTED | READ UNCOMMITTED } READ WRITE | READ ONLY
```

命令: TRUNCATE

描述: 清空一个表

语法:

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

命令: UPDATE

描述: 更新表的记录

语法:

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

命令: VACUUM

描述: 垃圾回收和可选择分析一个数据库

语法:

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

命令: VALUES

Copyright © 2012-2021 UCloud 优刻得

描述: 计算值表达式的值

语法:

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

13、用户自定义函数

udw 支持用户自定义函数,关于用户自定义函数请参考:官方文档

udw优化指南

1 数据类型

通过选择最合适的数据类型可以节省磁盘空间,提高查询性能。字符类型的数据建议使用 TEXT 或者 VARCHAR 类型,不要使用 CHAR 类型。TEXT 或者 VARCHAR 类型能够减少存储空间。选取合适的数字类型,如 INT 或者 SMALLINT 能满足业务需求时,使用 BIGINT 会带来磁盘空间的浪费。

2 存储方式

udw 在创建表格的时候提供了堆表和追加表两种存储模式,提供了行存储、列存储、外部表的存储方式。

2.1 堆表和追加表

Copyright © 2012-2021 UCloud 优刻得

堆表适合于较小、经常更新的数据存储方式。

追加表适合不经常更新，批量 INSERT 的大表。不要在追加表上执行单个的 INSERT、UPDATE 或者 DELETE 操作。并发的 INSERT 操作是可以，但是不要执行并发的 UPDATE 或者 DELETE 操作。

2.2 行存储和列存储

行存储的应用场景：

1. 表数据在载入后经常update;
2. 表数据经常insert；
3. 查询中选择大部分的列；

列存储的应用场景：

列存储一般适用于宽表（即字段非常多的表）。在使用列存储时，同一个字段的数据连续保存在一个物理文件中，所以列存储的压缩率比普通压缩表的压缩率要高很多，另外在多数字段中筛选其中几个字段中，需要扫描的数据量很小，扫描速度比较快。因此，列存储尤其适合在宽表中对部分字段进行筛选的场景。

列存储的表必须是追加表（Appendonly table）。

3 压缩

压缩能够节约存储空间并减少从存储读取的数据大小，这种方法还可以减少磁盘 I/O 量，因此可提高查询性能。

4 数据分布

UDW表的记录有两种分布策略，分别是哈希分布（DISTRIBUTED BY(key)）和随机分布(DISTRIBUTED RANDOMLY)。如果不指定分布策略则默认按 primary key 或者第一个 column

Copyright © 2012-2021 UCloud 优刻得

做哈希分布。

在MPP环境下，查询的执行时间是由所有节点决定的。当数据出现倾斜时，会导致较低的性能以及内存溢出的情况。

当选择分布键时，考虑以下几个方面：

1. 为所有的表显示地指定哈希或随机分布，不要使用默认的。
2. 理想的情况下，使用一个可以将数据均匀分布在各个节点上的字段作为分布键。
3. 不要选择可能出现在 WHERE 子句的字段作为分布键。
4. 不要选择时间或时间戳类型的字段作为分布键。
5. 作为分布键的字段的值应具有唯一性或非常高的散列程度。
6. 如果单个字段无法达到平均分布，使用多个字段作为分布键，但不要超过两个。
7. 如果两个字段作为分布键仍无法达到平均分布，考虑使用随机分布。
8. 常做join操作的大表之间，选择相同的字段作为分布键（字段的顺序也要一样）。
9. 作为分布键的字段数据类型也要一致（虽然不同数据类型的某些值是一样的，但是它们的存储方式不同，哈希之后的值会不一样，因而会分布到不同的节点上）。

5 导入数据

可以参考上面的加载数据、我们提供丰富的数据导入方法，我们不建议使用 insert 一条条的导入数据、这个效率是比较低的。强烈建议用 copy 或者其他更高效的数据导入方式。

6 分区

分区表主要用来存放大表，把大数据切片，便于查询，也便于数据库维护。分区创建时，每个分区会自带一个Check约束，来限定数据的范围。

关于分区的一些建议：

1. 只对大表进行进行分区。
2. 不要使用默认分区。
3. 对行存储的表不要使用过多的分区。

7 索引

Copyright © 2012-2021 UCloud 优刻得

在绝大部分传统数据中，索引都能够极大地提高数据访问速速。然而，在像 UDW 数据仓库这样的分布式数据库系统中，索引的使用需要更加谨慎。

索引会增加数据库系统的运行开销，它们占用存储空间并且在数据更新时，需要额外的维护工作。请确保查询集合在使用您创建的索引后，性能得到了改善（和全表顺序扫描相比）。可以使用 EXPLAIN 命令来确认索引是否被使用。

创建索引时，需要注意下面的问题点：

1. 查询特点：索引对于查询只返回单条记录或者较少的数据集时，性能提升明显。
2. 压缩表：对于压缩的 append 表来说，对于返回一部分数据的查询来说性能也能得到提高。对于压缩的数据，使用索引访问方法时，只有需要的数据才会被解压缩。
3. 避免在经常改变的列上创建索引：在经常更新的列上创建索引会导致每次更新数据时写操作大量增加。
4. 创建选择率高的 B-树索引，对于选择率较低的列，使用 Bitmap 索引。
5. 对参与连接操作的列创建索引：对经常用于连接的列（例如：外键列）创建索引，可以让查询优化器使用更多的连接算法，进而提高连接效率。
6. 对经常出现在 WHERE 条件中的列创建索引。
7. 避免创建冗余的索引：如果索引开头几列重复出现在多个索引中，这些索引就是冗余的。
8. 在大量数据加载时，删除索引：如果要向表中加载大量数据，考虑加载数据前删除索引，加载后重新建立索引的方法。这样的操作通常比带着索引加载要快。
9. 考虑聚簇索引：聚簇索引是指数据在物理上，按照索引顺序存储。如果您访问的数据在磁盘是随机存储，那么数据库就需要在磁盘上不断变更位置读取您需要的数据。如果数据更加紧密的存储起来，读取数据的操作效率就会更高。例如：在日期列上创建聚簇索引，数据也是按照日期列顺序存储。一个查询如果读取一个日期范围的数据，那么就可以利用磁盘顺序扫描的快速特性。

8 ANALYZE/VACUUM

定期的执行 ANALYZE 或者在表格数据变化比较大的时候执行、可以提高查询效率，定期的执行 VACUUM 可以清理一些磁盘空间。但是执行 ANALYZE 和 VACUUM 都会占用一些系统资源。

Copyright © 2012-2021 UCloud 优刻得

表膨胀

表膨胀的原因

udw的存储实现(MVCC-多版本并发控制)来自于Postgres。根据MVCC的原理，没有办法直接更新数据(更新操作(update)是通过先删除(delete)再插入(insert)实现的)，被更新之前的行数据仍然在数据文件中。

如何避免表膨胀

方法一：**vacuum full table**

> vacuum full不能回收索引的膨胀空间。vacuum full 加载的锁与 DDL 锁类似，是排它锁。建议在没有业务的时候执行，不要堵塞业务。

使用 vacuum full 回收垃圾的建议操作流程：

1. 记录下表的索引
2. 删除索引
3. vacuum full 表
4. 重建索引

示例：

Copyright © 2012-2021 UCloud 优刻得

创建测试表：

```
dev=# create table test(id int, name text);
NOTICE: Table doesn't have 'DISTRIBUTED BY' clause -- Using column named 'id' as the Greenplum Database data distribution key for this table.
HINT: The 'DISTRIBUTED BY' clause determines the distribution of data. Make sure column(s) chosen are the optimal data distribution key to minimize skew.
CREATE TABLE

dev=# insert into test select generate_series(1,100000000), 'test';
INSERT 0 100000000

dev=# create index idx_test on test(id);
CREATE INDEX

dev=# update test set name='nice';
UPDATE 100000000
```

查看表格数据：

```
dev=# select pg_size_pretty(pg_relation_size('test'));
pg_size_pretty
----------------
8401 MB
(1 row)
```

查看索引数据：

Copyright © 2012-2021 UCloud 优刻得

```
dev=# select pg_size_pretty(pg_relation_size('idx_test'));
pg_size_pretty
----------------
6377 MB
(1 row)
```

先回收表数据(此方法不能回收索引数据):

```
dev=# vacuum full test;
VACUUM

dev=# select pg_size_pretty(pg_relation_size('test'));
pg_size_pretty
----------------
4200 MB
(1 row)

Time: 4.278 ms
dev=# select pg_size_pretty(pg_relation_size('idx_test'));
pg_size_pretty
----------------
6377 MB
(1 row)
```

Copyright © 2012-2021 UCloud 优刻得

回收索引和表的数据：

```
dev=# drop index idx_test;
DROP INDEX

dev=# vacuum full test;
VACUUM

dev=# create index idx_test on test(id);
CREATE INDEX

dev=# select pg_size_pretty(pg_relation_size('test'));
pg_size_pretty
----------------
4200 MB
(1 row)

dev=# select pg_size_pretty(pg_relation_size('idx_test'));
pg_size_pretty
----------------
2126 MB
(1 row)
```

方法二：通过修改分布键释放空间

Copyright © 2012-2021 UCloud 优刻得

修改分布键可以回收索引的膨胀空间。修改分布键加载的锁与 DDL 锁类似，是排它锁。建议在没有业务的时候执行，不要影响业务。

```
alter table test set with (reorganize=true) distributed randomly;
alter table test set with (reorganize=true) distributed by (id);
```

实例：

```
dev=# update test set name='back';
UPDATE 100000000
```

查看数据：

```
dev=# select pg_size_pretty(pg_relation_size('test'));
pg_size_pretty
----------------
8401 MB
(1 row)

dev=# select pg_size_pretty(pg_relation_size('idx_test'));
pg_size_pretty
----------------
4251 MB
(1 row)
```

查看表格的分布键，如下所示Distributed by: (id)，分布键为id

Copyright © 2012-2021 UCloud 优刻得

```
dev=# \d+ test
Table "public.test"
Column | Type | Modifiers | Storage | Description
--------+---------+-----------+----------+-------------
id | integer | | plain |
name | text | | extended |
Indexes:
"idx_test" btree (id)
Has OIDs: no
Distributed by: (id)
```

按照原有的分布键重新分布

```
dev=# alter table test set with (reorganize=true) distributed by (id);
ALTER TABLE
```

查看数据

```
dev=# select pg_size_pretty(pg_relation_size('test'));
pg_size_pretty
----------------
4200 MB
(1 row)
```

Copyright © 2012-2021 UCloud 优刻得

```
dev=# select pg_size_pretty(pg_relation_size('idx_test'));
pg_size_pretty
----------------
2126 MB
```

方法三：创建新表，导入数据

CREATE TABLE...AS SELECT 命令把该表拷贝为一个新表，新建的表将不会出现膨胀现象。然后删除原始表并且重命名拷贝的表。

参考：

- https://gp-docs-cn.github.io/docs/best_practices/bloat.html
- https://docs.ucloud.cn/udw/developer?id=_43-%e9%80%89%e6%8b%a9%e6%95%b0%e6%8d%ae%e5%88%86%e5%b8%83%e7%ad%96%e7%95%a5

Copyright © 2012-2021 UCloud 优刻得

UDW中Json类型

Json相关操作

操作符	参数类型	作用	例子	执行结果
->	int	获取JSON数组元素，索引以0为开始	select '[{"a":"foo"},{"b":"bar"},{"c":"baz"}]'::json->2;	{"c":"baz"}
->	text	通过键来获取 JSON 对象的域（field）	select '{"a": {"b":"foo"}}'::json->'a';	{"b":"foo"}
->>	int	获取 JSON 数组元素，然后以 text 形式返回它	select '[1,2,3]'::json->>2;	3
->>	text	获取 JSON 对象的域，然后以 text 形式返回它	select '{"a":1,"b":2}'::json->>'b';	2
#>	text[]	获取指定路径上的 JSON 对象	select '{"a": {"b":{"c": "foo"}}}'::json#>'{a,b}';	{"c": "foo"}
#>>	text[]	获取指定路径上的 JSON 对象，并以 text 形式返回它	select '{"a":[1,2,3],"b":[4,5,6]}'::json#>>'{a,2}';	3

Json操作举例

创建json类型的表格，插入数据

```
create table test_json(id int , name json)
```

Copyright © 2012-2021 UCloud 优刻得

```
with (APPENDONLY=true,ORIENTATION=column,compresslevel=5) DISTRIBUTED BY (id);
INSERT INTO test_json VALUES
(1,'{ "id": 1, "sub": { "subid": 10,"subsub": {"subsubid": 100}}}'),
(2,'{ "id": 20, "sub": { "subid": 200,"subsub": {"subsubid": 2000}}}'),
(1,'{ "id": 1, "sub": { "subid": "test","subsub": {"subsubid": 100}}}'),
(3,'{ "id": 1,"sub":"test","name":"me","ip":"10.10.10.10" }');
```

json操作类型操作举例：

```
select * from test_json where name->>'id'=1;
id | name
----+--------------------------------------------------------------------
1 | { "id": 1, "sub": { "subid": 10,"subsub": {"subsubid": 100}}}
1 | { "id": 1, "sub": { "subid": "test","subsub": {"subsubid": 100}}}
3 | { "id": 1,"sub":"test","name":"me","ip":"10.10.10.10" }


select * from test_json where name->'sub'->>'subid'=10;
id | name
----+--------------------------------------------------------------
1 | { "id": 1, "sub": { "subid": 10,"subsub": {"subsubid": 100}}}

select * from test_json where name->>'name'='me';
id | name
```

Copyright © 2012-2021 UCloud 优刻得

```
----+-----------------------------------------------------------
3 | { "id": 1,"sub":"test","name":"me","ip":"10.10.10.10" }
```

Json相关函数

Json创建函数

```
to_json(anyelement)
array_to_json(anyarray [, pretty_bool])
row_to_json(record [,pretty_bool])
json_build_array(VARIADIC "any")
json_build_object(VARIADIC "any")
json_object(text[])
json_object(keys text[], values text[])
```

Json处理函数

```
json_array_length(json)
json_extract_path(from_json json, VARIADIC path_elems text[])
json_extract_path_text(from_json json, VARIADICpath_elems text[])
json_object_keys(json)
json_populate_record( base anyelement, from_json json)
```

Copyright © 2012-2021 UCloud 优刻得

```
json_typeof(json)
json_to_record(json)
json_to_recordset(json)
```

json函数的详细操作请参考文档下面的部分。

Json创建函数

```
to_json(anyelement)
```

以 JSON 格式返回输入的值。数组和复合数据会被（递归地）转换为数组和对象；如果有转换函数可以将输入的数据转换为 json 的话，那么使用转换函数；或者产生一个 JSON 标量（scalar）值。数字、布尔值和空值（null）之外的其他标量会被表示为文本格式，并通过正确的引用和转义来保证它是一个合法的 JSON 字符串。如下所示：

```
test=# select to_json('Fred said "Hi."'::text);
       to_json
---------------------
 "Fred said \"Hi.\""
(1 row)
```

```
array_to_json(anyarray [, pretty_bool])
```

Copyright © 2012-2021 UCloud 优刻得

以 JSON 数组格式返回输入的数组。一个UDW多维数组将被转换成一个由多个数组组成的 JSON 数组。如果 pretty_bool的值为 true ，那么则在维度-1元素之间添加换行符。如下所示：

```
test=# SELECT array_to_json(array(select 1 as a));
 array_to_json
---------------
 [1]
(1 row)

test=# SELECT array_to_json(array_agg(q),false) from (select x as b, x * 2 as c from generate_series(1,3) x) q;
                  array_to_json
-------------------------------------------------
 [{"f1":1,"f2":2},{"f1":2,"f2":4},{"f1":3,"f2":6}]
(1 row)

test=# SELECT array_to_json(array_agg(q),true) from (select x as b, x * 2 as c from generate_series(1,3) x) q;
   array_to_json
--------------------
 [{"f1":1,"f2":2},
  {"f1":2,"f2":4},
  {"f1":3,"f2":6}]
(1 row)
```

row_to_json(record [,pretty_bool])

以 JSON 对象格式返回行。如果pretty_bool为 true，将在级别-1元素之间添加换行符。

Copyright © 2012-2021 UCloud 优刻得

```
test=# SELECT row_to_json(row(1,'foo'));
     row_to_json
---------------------
 {"f1":1,"f2":"foo"}
(1 row)

test=# SELECT row_to_json(q)
test-# FROM (SELECT $$a$$ || x AS b,
test(#           y AS c,
test(#           ARRAY[ROW(x.*,ARRAY[1,2,3]),
test(#                 ROW(y.*,ARRAY[4,5,6])] AS z
test(#       FROM generate_series(1,2) x,
test(#            generate_series(4,5) y) q;
                              row_to_json
-----------------------------------------------------------------------
 {"f1":"a1","f2":4,"f3":[{"f1":1,"f2":[1,2,3]},{"f1":4,"f2":[4,5,6]}]}
 {"f1":"a1","f2":5,"f3":[{"f1":1,"f2":[1,2,3]},{"f1":5,"f2":[4,5,6]}]}
 {"f1":"a2","f2":4,"f3":[{"f1":2,"f2":[1,2,3]},{"f1":4,"f2":[4,5,6]}]}
 {"f1":"a2","f2":5,"f3":[{"f1":2,"f2":[1,2,3]},{"f1":5,"f2":[4,5,6]}]}
(4 rows)
```

Copyright © 2012-2021 UCloud 优刻得

json_build_array(VARIADIC "any")

建立一个可能不同类型的JSON数组，由可变参数列表组成。例如：

```
test=# SELECT json_build_array('a',1,'b',1.2,'c',true,'d',null,'e',json '{"x": 3, "y": [1,2,3]}');
                          json_build_array
---------------------------------------------------------------------
 ["a", 1, "b", 1.2, "c", true, "d", null, "e", {"x": 3, "y": [1,2,3]}]
(1 row)
```

json_build_object(VARIADIC "any")

建立一个JSON对象的可变参数列表。根据习惯，该参数列表由交替的键和值组成。例如：

```
test=# SELECT json_build_object('a',1,'b',1.2,'c',true,'d',null,'e',json '{"x": 3, "y": [1,2,3]}');
                          json_build_object
---------------------------------------------------------------------
 {"a" : 1, "b" : 1.2, "c" : true, "d" : null, "e" : {"x": 3, "y": [1,2,3]}}
(1 row)

test=#
test=# SELECT json_build_object(
test(#         'a', json_build_object('b',false,'c',99),
test(#         'd', json_build_object('e',array[9,8,7]::int[],
test(#             'f', (select row_to_json(r) from ( select relkind, oid::regclass as name from pg_class where relname = 'pg_class') r)));
                                 json_build_object
-----------------------------------------------------------------------------------
 {"a" : {"b" : false, "c" : 99}, "d" : {"e" : [9,8,7], "f" : {"f1":"r","f2":"pg_class"}}}
(1 row)
```

json_object(text[])

输入的文本数组构建一个 JSON 对象。输入的数组要么就是由偶数个成员组成的一维数组，数组中的每两个成员组成一个键值对；要么就是一个二维数组，并且每个内部数组都正好包含两个元素，这两个元素组成一个键值对。例如：

Copyright © 2012-2021 UCloud 优刻得

```
test=# SELECT json_object('{a,1,b,2,3,NULL,"d e f","a b c"}');
                    json_object
-----------------------------------------------------
 {"a" : "1", "b" : "2", "3" : null, "d e f" : "a b c"}
(1 row)

test=# SELECT json_object('{{a,1},{b,2},{3,NULL},{"d e f","a b c"}}');
                    json_object
-----------------------------------------------------
 {"a" : "1", "b" : "2", "3" : null, "d e f" : "a b c"}
(1 row)
```

```
json_object(keys text[], values text[])
```

这个格式的 json_object 函数接受两个数组作为输入，第一个数组的元素会被用作键值对的键，而第二个数组的元素则会被用作键值对的值。

```
test=# select json_object('{a,b,c,"d e f"}','{1,2,3,"a b c"}');
                      json_object
-------------------------------------------------------
 {"a" : "1", "b" : "2", "c" : "3", "d e f" : "a b c"}
(1 row)
```

Json处理函数

```
json_array_length(json)
```

返回最外层的 JSON 数组的元素数量。例如：

Copyright © 2012-2021 UCloud 优刻得

```
test=# SELECT json_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4]');
 json_array_length
-------------------
                 5
(1 row)

test=# SELECT json_array_length('[]');
 json_array_length
-------------------
                 0
(1 row)
```

json_extract_path(from_json json, VARIADIC path_elems text[])

返回 path_elems 所指向的 JSON 值。等同于 #> 操作符。例如：

```
test=# select json_extract_path('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}','f4','f6');
 json_extract_path
-------------------
 "stringy"
(1 row)

test=# select json_extract_path('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}','f2');
 json_extract_path
-------------------
 {"f3":1}
(1 row)

test=# select json_extract_path('{"f2":["f3",1],"f4":{"f5":99,"f6":"stringy"}}','f2',0::text);
 json_extract_path
-------------------
 "f3"
(1 row)

test=# select json_extract_path('{"f2":["f3",1],"f4":{"f5":99,"f6":"stringy"}}','f2',1::text);
 json_extract_path
-------------------
 1
(1 row)
```

Copyright © 2012-2021 UCloud 优刻得

json_extract_path_text(from_json json, VARIADICpath_elems text[])

以 text 格式，返回 path_elems 所指向的 JSON 值。效果等同于 #>> 操作符。例如：

```
test=# select json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}','f4','f6');
 json_extract_path_text
------------------------
 stringy
(1 row)

test=# select json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}','f2');
 json_extract_path_text
------------------------
 {"f3":1}
(1 row)
```

json_object_keys(json)

返回最外层的 JSON 对象所包含的键。例如：

```
test=# select json_object_keys ('{"f1":"abc","f2":{"f3":"a", "f4":"b"}}');
 json_object_keys
------------------
 f1
 f2
(2 rows)
```

json_populate_record(base anyelement, from_json json)

将 from_json 中的对象展开到一个行里面，这个行的各个列与 base 中定义的 record 类型一致。例如：

Copyright © 2012-2021 UCloud 优刻得

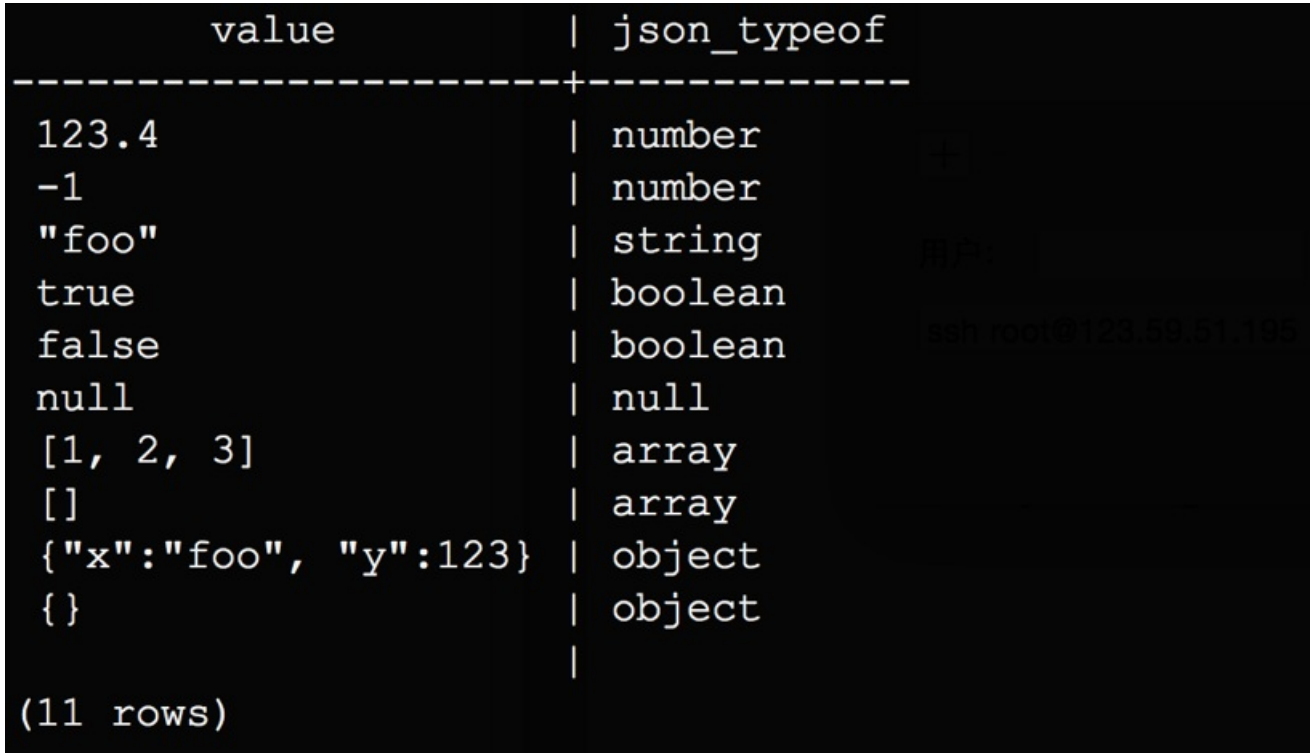
```
test=# create type jpop as (a text, b int, c timestamp);
CREATE TYPE
test=# select * from json_populate_record(null::jpop,'{"a":"blurfl","x":43.2}') q;
   a    | b | c
--------+---+---
 blurfl |   |
(1 row)

test=# select * from json_populate_record(row('x',3,'2012-12-31 15:30:56')::jpop,'{"a":"blurfl","x":43.2}') q;
   a    | b |          c
--------+---+---------------------
 blurfl | 3 | 2012-12-31 15:30:56
(1 row)
```

json_typeof(json)

以字符串形式返回最外层 JSON 值的类型。可能出现的类型有 object 、array、string、number 、boolean 和 null 。例如：

```
test=# select value, json_typeof(value)
test-#   from (values (json '123.4'),
test(#                (json '-1'),
test(#                (json '"foo"'),
test(#                (json 'true'),
test(#                (json 'false'),
test(#                (json 'null'),
test(#                (json '[1, 2, 3]'),
test(#                (json '[]'),
test(#                (json '{"x":"foo", "y":123}'),
test(#                (json '{}'),
test(#                (NULL::json))
test-#         as data(value);
```

Copyright © 2012-2021 UCloud 优刻得



```
        value          | json_typeof
-----------------------+-------------
 123.4                 | number
 -1                    | number
 "foo"                 | string
 true                  | boolean
 false                 | boolean
 null                  | null
 [1, 2, 3]             | array
 []                    | array
 {"x":"foo", "y":123}  | object
 {}                    | object
                       |
(11 rows)
```

json_to_record(json)

根据一个 JSON 对象来构建一个任意的 record 。和所有返回 record 的函数一样，调用者必须通过 as 语句来明确地定义 record 的结构。

Copyright © 2012-2021 UCloud 优刻得

```
test=# select * from json_to_record('{"a":1,"b":"foo","c":"bar"}')
test-#     as x(a int, b text, d text);
 a |  b  | d
---+-----+---
 1 | foo |
(1 row)

test=# select *, c is null as c_is_null
test-# from json_to_record('{"a":1, "b":{"c":16, "d":2}, "x":8}'::json)
test-#     as t(a int, b json, c text, x int);
 a |        b        | c | x | c_is_null
---+-----------------+---+---+-----------
 1 | {"c":16, "d":2} |   | 8 | t
(1 row)
```

json_to_recordset(json)

根据一个由 JSON 对象组成的数组，构建一个任意的 record 集合。和所有返回 record 的函数一样，调用者必须通过 as 语句来明确地定义 record 的结构。例如：

Copyright © 2012-2021 UCloud 优刻得

接入第三方 BI 工具

UDW可以接入第三方商业智能(BI)工具来快速实现数据的可视化。第三方商业智能(BI)工具使用标准数据库接口连接 UDW 数据仓库,例如:JDBC 和 ODBC。

目前经过测试的有:Zeppelin 和 SuperSet。

一、 UDW 接入 Zeppelin

Zeppelin 简介

Zeppelin 是一个开源的 Apache 的孵化项目. 它是一款基本 web 的 notebook 工具,支持交互式数据分析。通过插件的方式接入各种解释器(interpreter),使得用户能够以特定的语言或数据处理后端来完成交互式查询,并快速实现数据可视化。

部署 Zeppelin

1) 安装 Java

Zeppelin 支持的操作系统如下图所示。在安装 Zeppelin 之前,你需要在部署的服务器上安装 Oracle JDK 1.7 或以上版本, 并配置好相应的 JAVA_HOME 环境变量。

Copyright © 2012-2021 UCloud 优刻得

Name	Value
Oracle JDK	1.7 (set `JAVA_HOME`)
OS	Mac OSX Ubuntu 14.X CentOS 6.X Windows 7 Pro SP1

以CentOS为例，具体操作过程如下：

a) 下载JDK安装包(jdk-7u79-linux-x64.tar.gz),下载地址为：

http://www.oracle.com/technetwork/cn/java/javase/downloads/jdk7-downloads-1880260.html。

创建JDK安装目录，并将安装包解压至该目录：

```
mkdir /usr/java

tar zxvf jdk-7u79-linux-x64.tar.gz
```

a) 建立软链接

```
ln -s /usr/java/jdk1.7.0_79 /usr/java/java
```

Copyright © 2012-2021 UCloud 优刻得

b) 配置环境变量。在 /etc/profile 文件结尾添加：

```
export JAVA_HOME=/usr/java/java
export JRE_HOME=${JAVA\_HOME}/jre
export PATH=${JAVA_HOME}/bin:$PATH
```

d) 使环境变量生效

```
source /etc/profile
```

2) 获取Zeppelin

下载地址：http://zeppelin.apache.org/download.html

选择二进制安装包，这里以zeppelin-0.6.2-bin-all.tgz为例。

3) 安装Zeppelin

安装Zeppelin只需如下命令解压二进制安装包即可：

```
tar zxvf zeppelin-0.6.2-bin-all.tgz
```

启动Zeppelin:

```
cd /data/zeppelin-0.6.2-bin-all (Zeppelin的安装目录)
```

Copyright © 2012-2021 UCloud 优刻得

```
bin/zeppelin-daemon.sh start
```

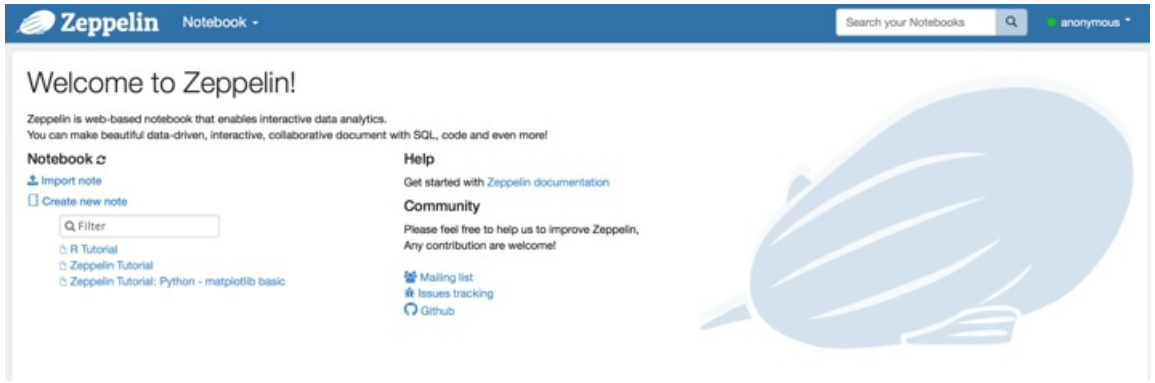
第一次启动Zeppelin，输出如下：

```
Log dir doesn't exist, create /data/zeppelin-0.6.2-bin-all/logs
Pid dir doesn't exist, create /data/zeppelin-0.6.2-bin-all/run
Zeppelin start                                             [  OK  ]
```

这说明Zeppelin已经部署成功。

4）验证

Zeppelin 默认启动在 8080 端口，在浏览器中访问 Zeppelin 主页，访问地址是: http://your_host_ip:8080/，你将看到类似如下的页面。

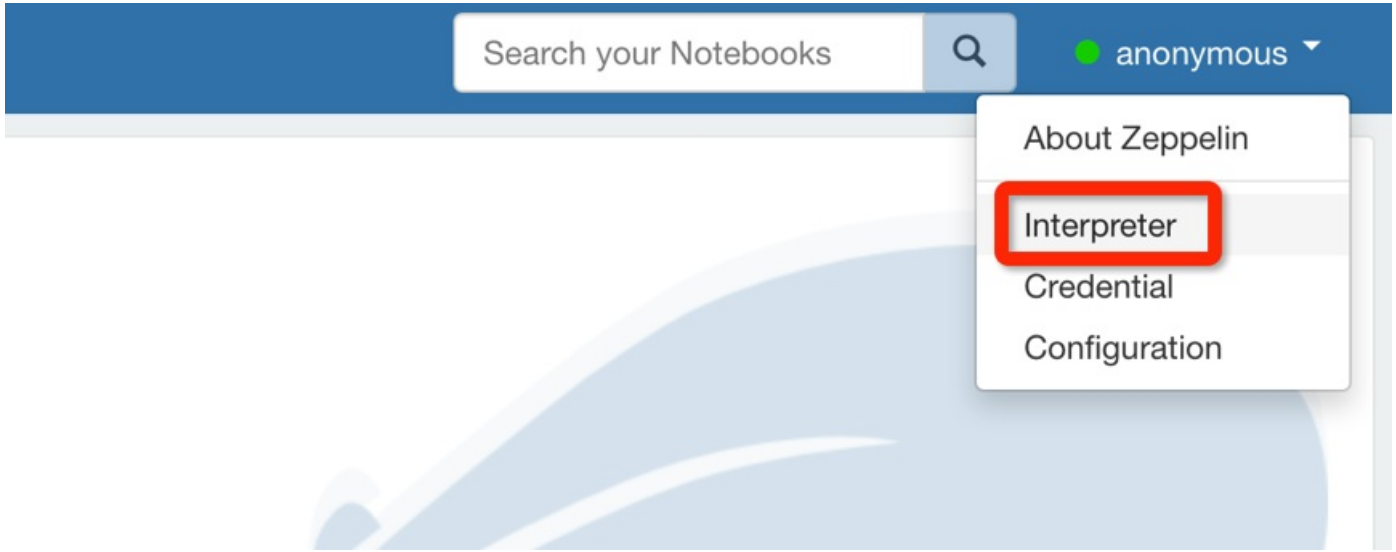


Zeppelin 接入 UDW 数据仓库

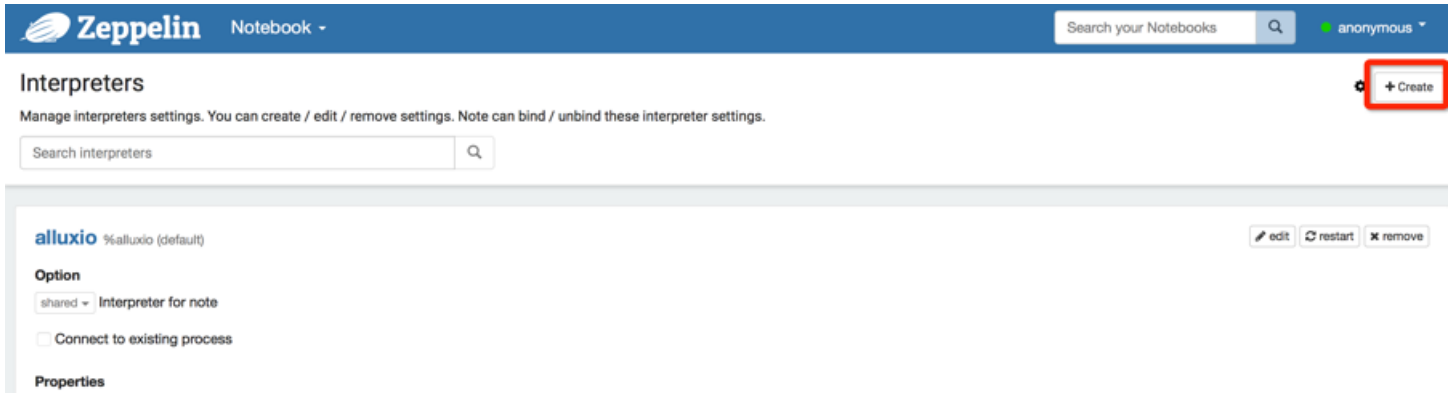
1）为 UDW 创建一个 interpreter。

鼠标点击右上角的“anonymous”，在弹出的下拉列表中选择“Interpreter”。

Copyright © 2012-2021 UCloud 优刻得

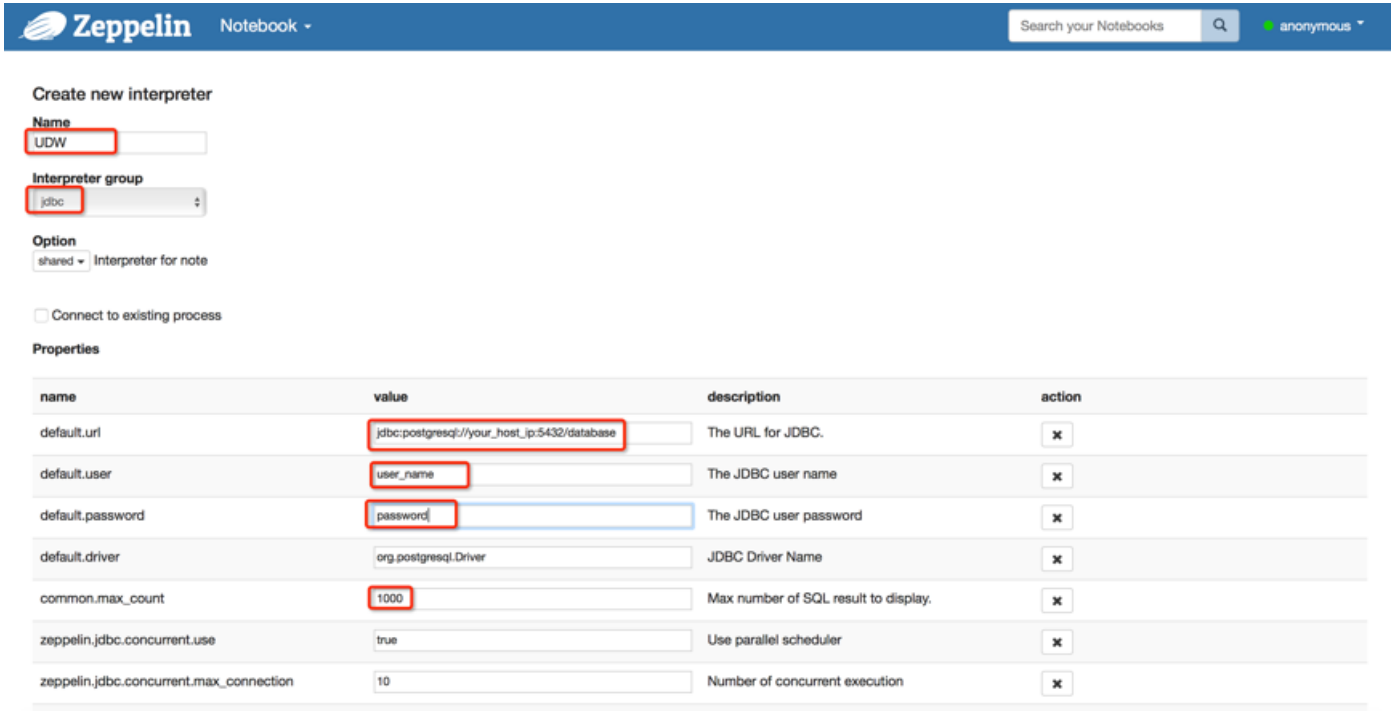


你将进入如下页面，然后点击右上角的“+Create”按钮。



接着，便进入了解释器的新建页面，如下图：

Copyright © 2012-2021 UCloud 优刻得



输入解释器的名字(可任意),解释器分组选择jdbc。

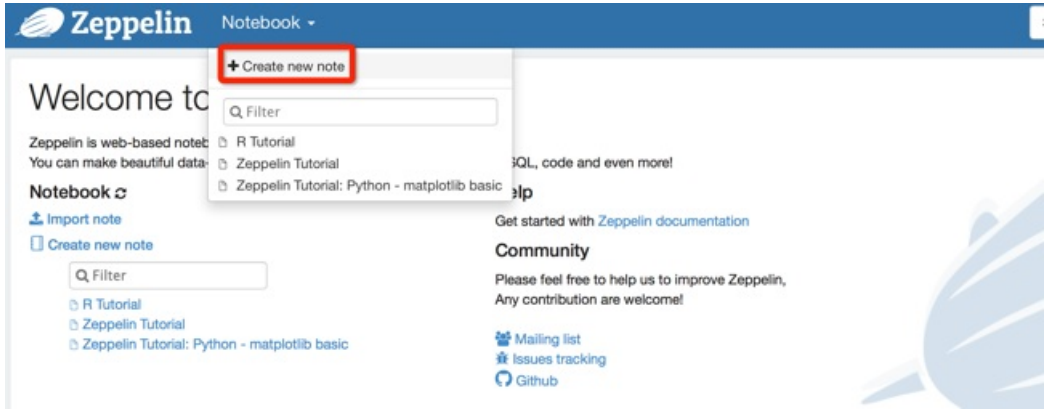
保留并修改上图中连接数据仓库的配置参数,点选action下面的“x”删除其他无关参数。然后点击“Save”按钮,保存设置。

2)创建笔记

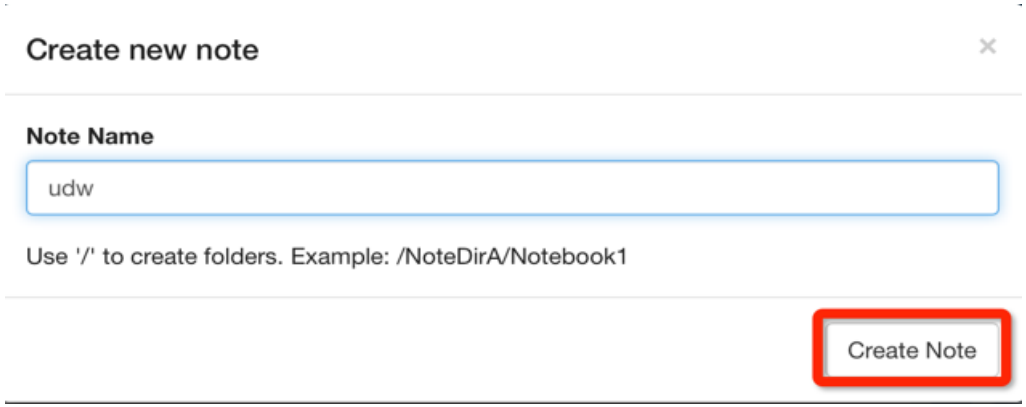
现在,你可以新建笔记来测试该Interpreter了。

鼠标点击“Notebook”下方的“Create new note”创建一个笔记。

Copyright © 2012-2021 UCloud 优刻得

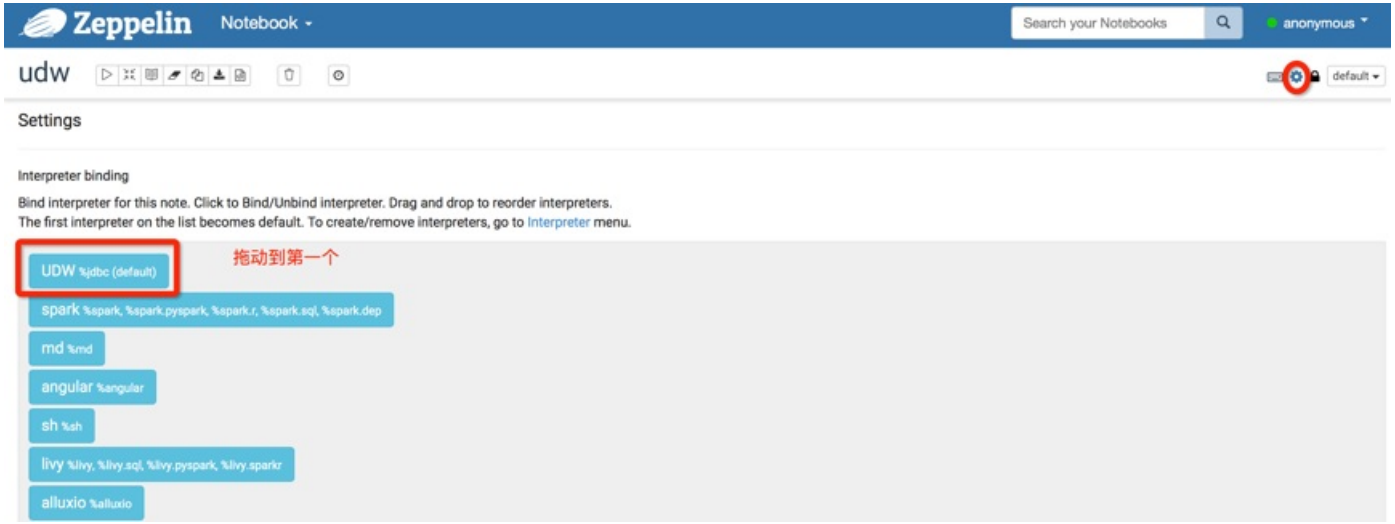


输入笔记名称(任意),点击“Create Note”,如下图:



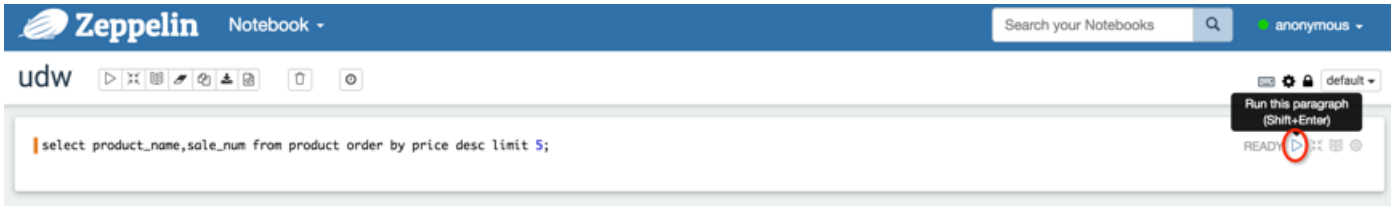
笔记创建完后,点击“齿轮”图标进行设置,将之前创建的解释器UDW移到最上方,作为默认的Interpreter使用。点击“Save”保存设置。

Copyright © 2012-2021 UCloud 优刻得



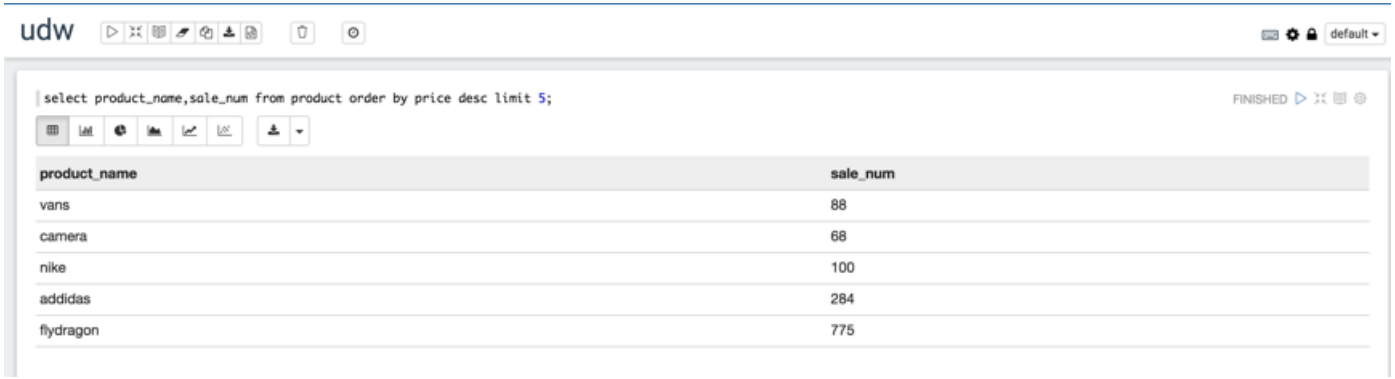
3）操作数据仓库

在方框内输入SQL语句，点击下图中的“▶”按钮，执行SQL。



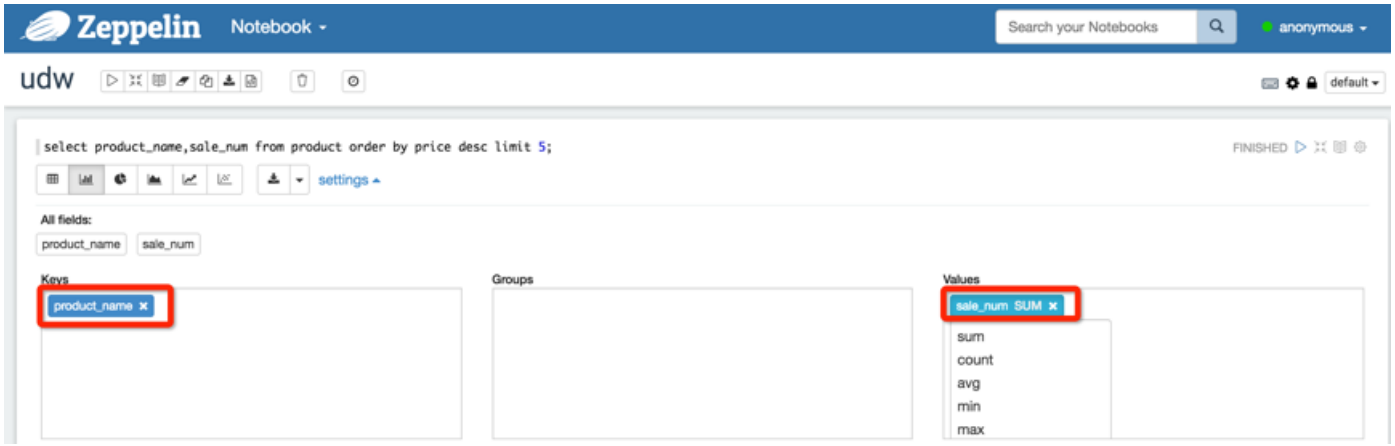
执行结束，输出如下。

Copyright © 2012-2021 UCloud 优刻得



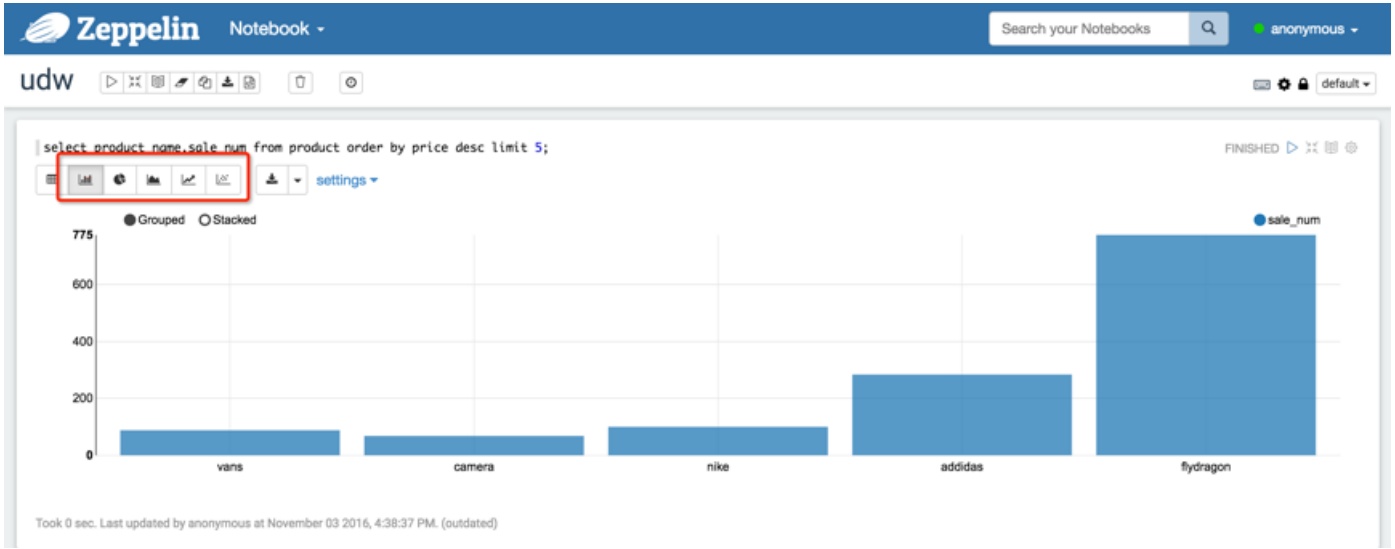
数据可视化

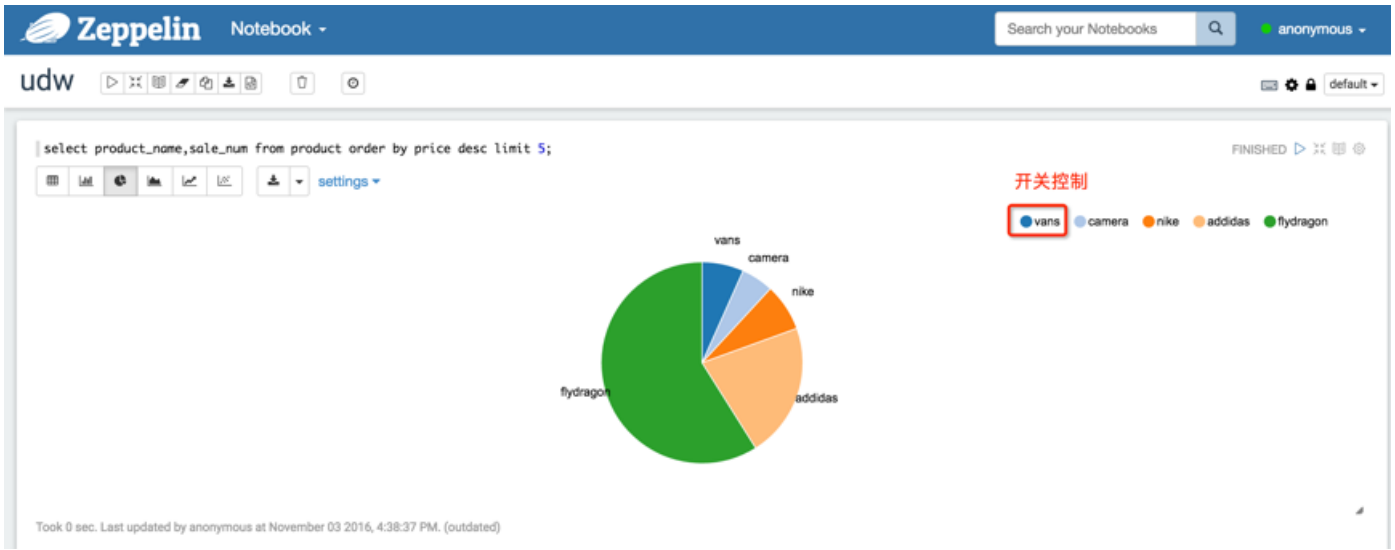
Zeppelin提供了非常丰富且简单的可视化功能，点击如下图中的可视化选项，拖动fields完成简单的setting设置，即可看到不同种类的可视化图表了。



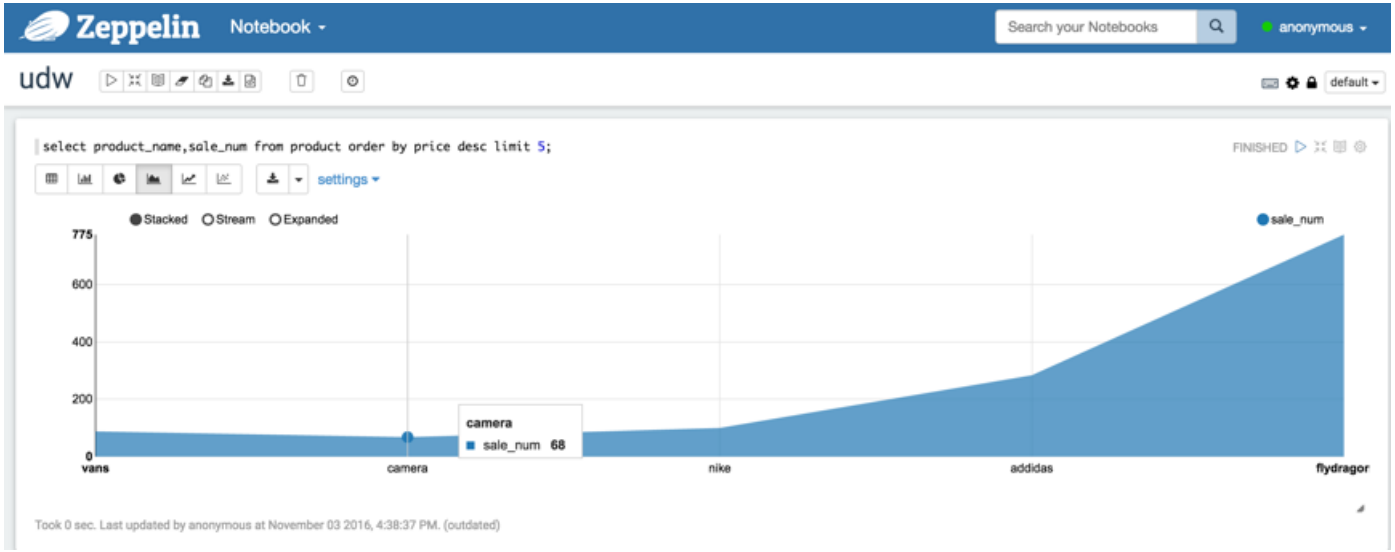
Zeppelin提供的5种可视化视图如下所示：

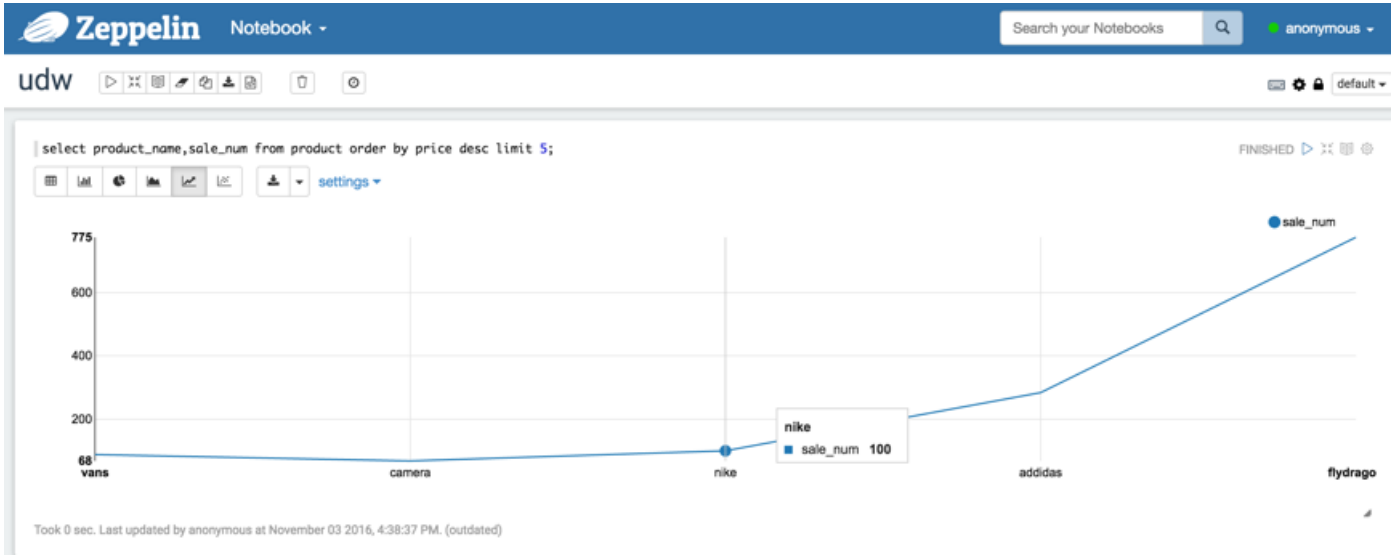
Copyright © 2012-2021 UCloud 优刻得



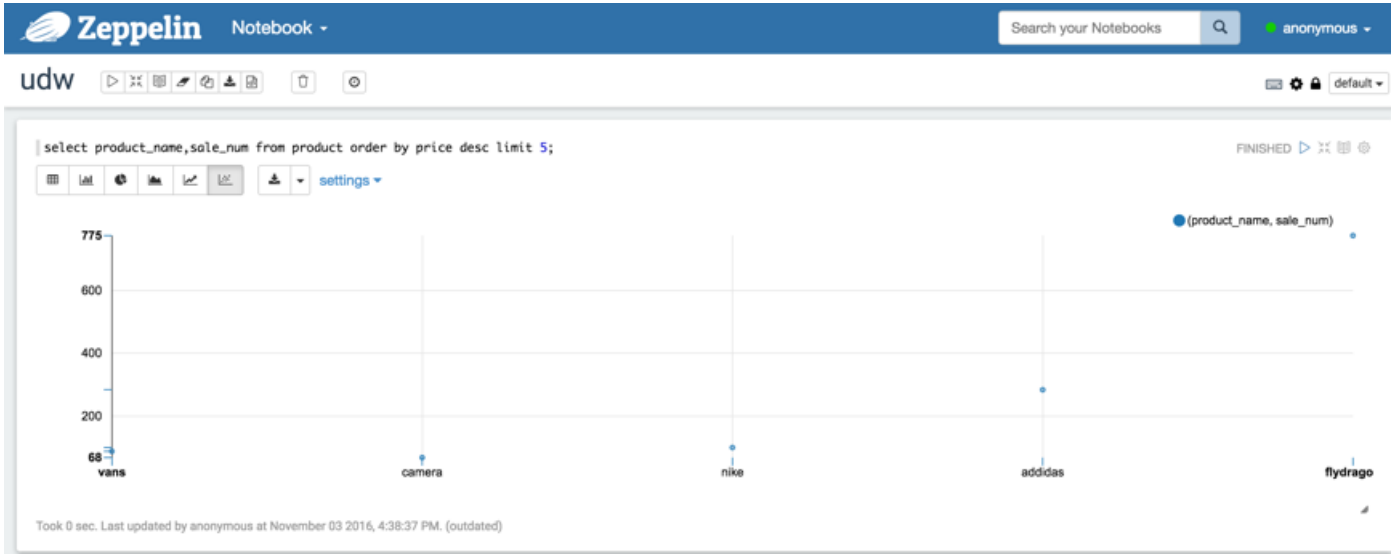


Copyright © 2012-2021 UCloud 优刻得





Copyright © 2012-2021 UCloud 优刻得



二、 UDW 接入 SuperSet

SuperSet 简介

Superset（Caravel）是由 Airbnb(知名在线房屋短租公司)开源的数据分析与可视化平台(曾用名Caravel、Panoramix)，该工具主要特点是可自助分析、自定义仪表盘、分析结果可视化(导出)、用户/角色权限控制，还集成了一个 SQL 编辑器，可以进行 SQL 编辑查询等。

部署 SuperSet

以 Centos 64 操作系统为例(其他操作系统可参考：http://airbnb.io/superset/installation.html)：

1) 安装Python3及组件

SuperSet 只支持 Python2.7 和 Python3.4 以上版本。SuperSet 官网建议使用 Python3。

Copyright © 2012-2021 UCloud 优刻得

安装系统依赖：

```
yum install gcc gcc-c++ libffi-devel python-devel python-pip
python-wheel openssl-devel libsasl2-devel openldap-devel sqlite-devel
zlib-devel bzip2-devel openssl-devel ncurses-devel postgresql-devel -y
```

安装Python3:

```
wget https://www.python.org/ftp/python/3.5.0/Python-3.5.0.tar.xz
tar Jxvf Python-3.5.0.tar.xz
cd Python-3.5.0
./configure --prefix=/usr/local/python3
make && make install
echo 'export PATH=$PATH:/usr/local/python3/bin' >> ~/.bashrc
source ~/.bashrc
rm /usr/bin/python
ln -sv /usr/local/python3/bin/python3.5 /usr/bin/python
```

更新yum配置

编辑 /usr/bin/yum，将第一行的 #!/usr/bin/python 改为 #!/usr/bin/python2.6，保存退出。

至此完成了 python3 的安装。

安装 fab

```
wget https://pypi.python.org/packages/source/f/fab/fab-1.4.2.tar.gz
```

Copyright © 2012-2021 UCloud 优刻得

```
tar zxvf fab-1.4.2.tar.gz
cd fab-1.4.2
python setup.py install
```

升级pip

```
pip install --upgrade pip
```

备注:如果pip升级过程报版本错误,请执行下面操作 请先 `mv /usr/bin/pip /usr/bin/pip.bak` 然后执行 `ln -s /usr/local/python3/bin/pip /usr/bin/pip`

安装psycopg2

```
pip install psycopg2==2.6.2
```

下载 http://udw.cn-bj.ufileos.com/extras.py,替换 `/usr/local/python3/lib/python3.5/site-packages/psycopg2/` 下的 `extras.py` 文件。

2)安装SuperSet

```
pip install superset
```

创建管理用户(后面登录web页面的时候会用到)

```
fabmanager create-admin --app superset
```

初始化数据库

Copyright © 2012-2021 UCloud 优刻得

```
superset db upgrade
```

创建默认角色和权限

```
superset init
```

更新 sqlalchemy

```
pip install sqlalchemy==1.0.16
```

在启动服务之前，还需要修改如下：

1、下载 http://udw.cn-bj.ufileos.com/base.py，替换 /usr/local/python3/lib/python3.5/site-packages/sqlalchemy/dialects/postgresql 目录下的 base.py。

2、下载 http://udw.cn-bj.ufileos.com/default.py，替换 /usr/local/python3/lib/python3.5/site-packages/sqlalchemy/engine 目录下的 default.py。

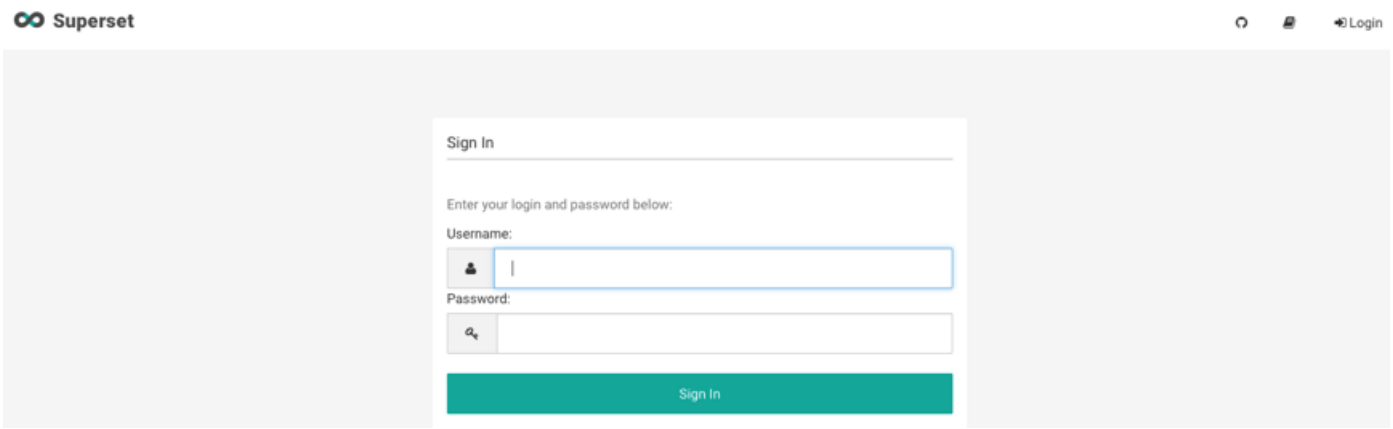
在 8088 端口启动 web 服务器(注意修改相应的防火墙保证8088端口可以被访问)

```
superset runserver -p 8088
```

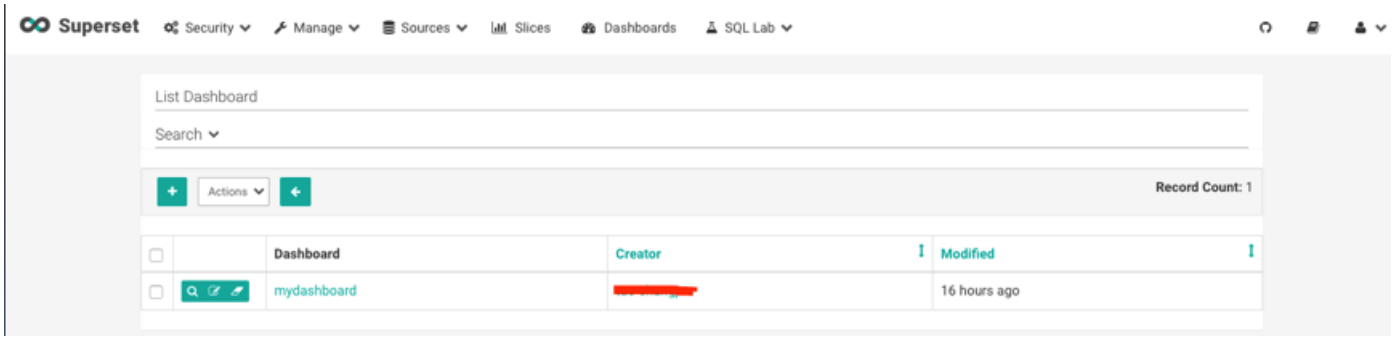
3)验证

SuperSet 默认启动在 8088 端口，在浏览器中访问 SuperSet 主页，访问地址是: http://your_host_ip:8088/,你将看到类似如下的登录页面。

Copyright © 2012-2021 UCloud 优刻得



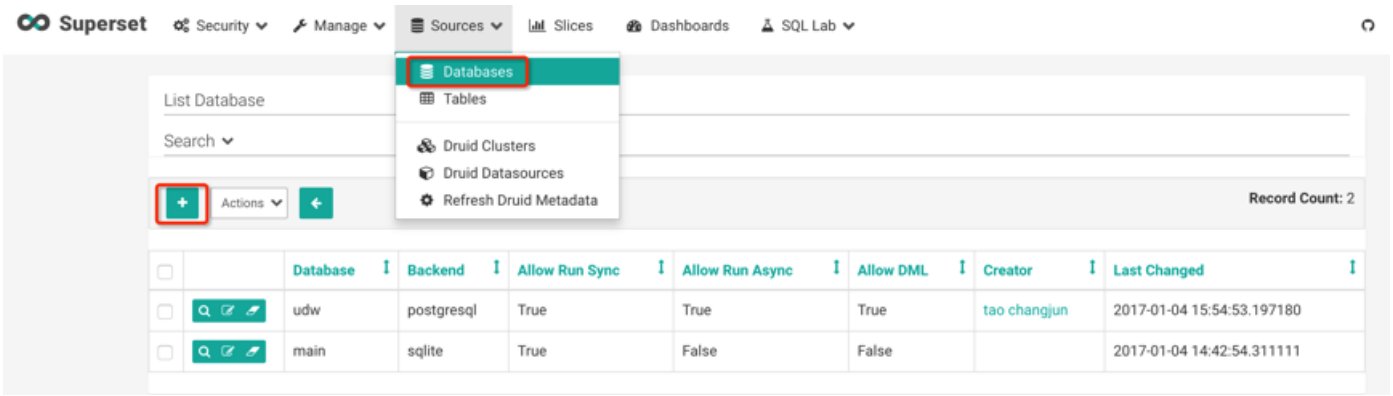
登录之后将看到如下页面：



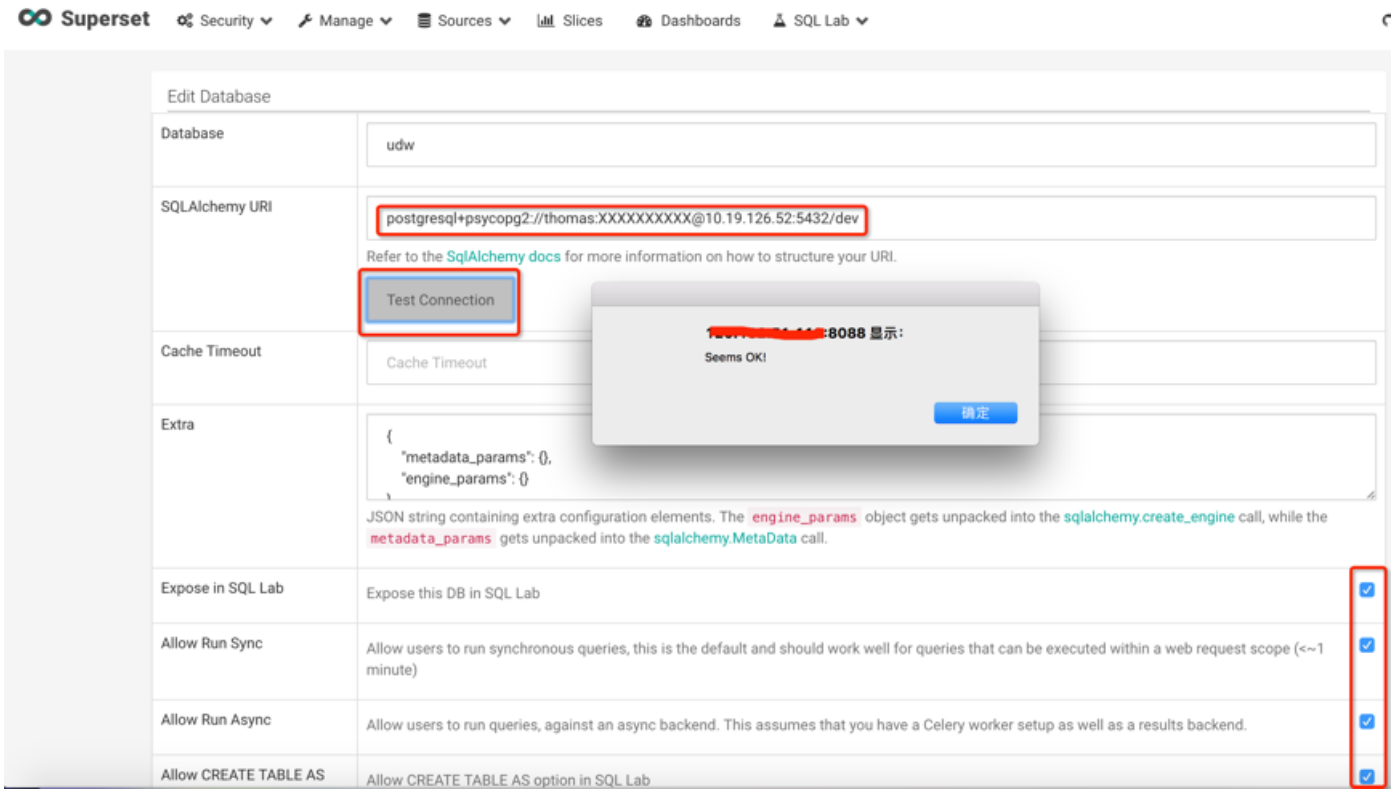
SuperSet 接入 UDW 数据仓库

1）创建数据源(Databases)

Copyright © 2012-2021 UCloud 优刻得



输入参数,测试连接



Copyright © 2012-2021 UCloud 优刻得

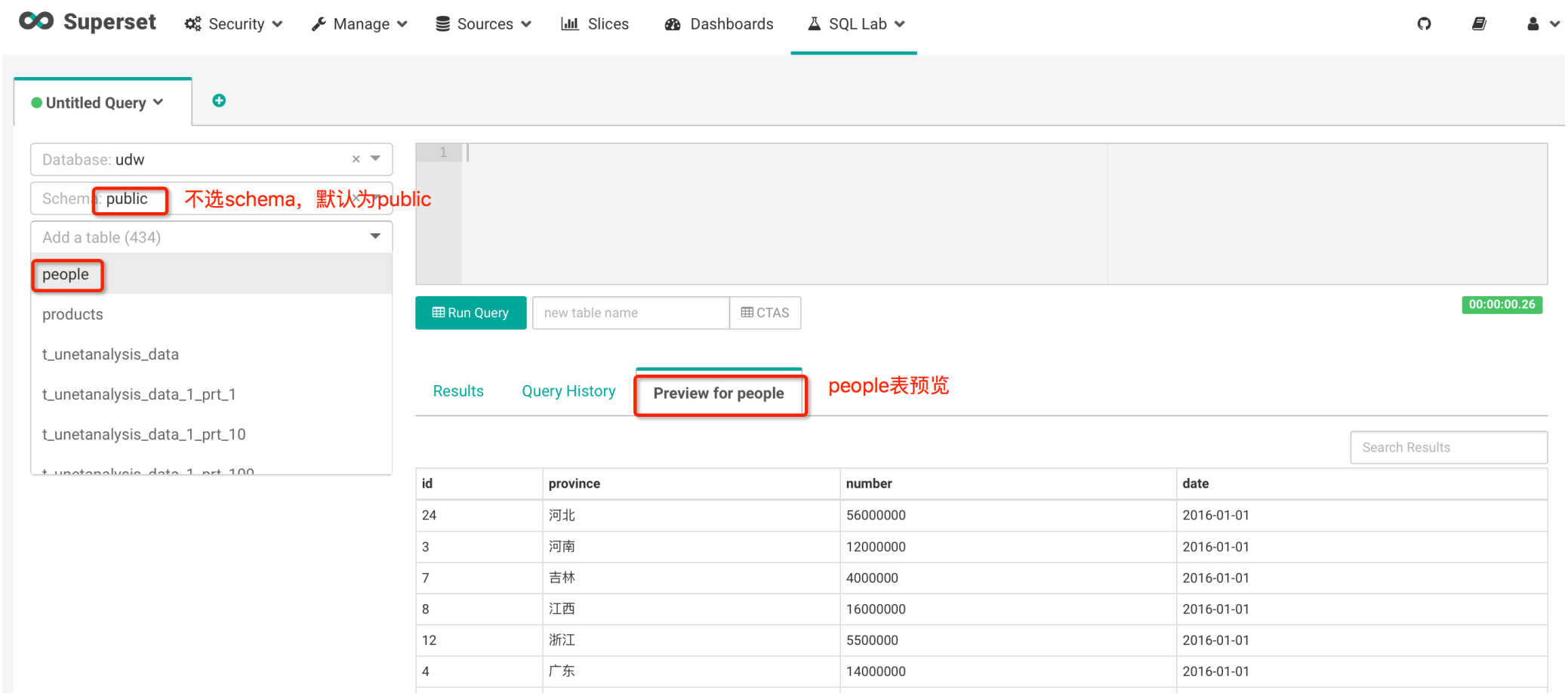
这里需要注意Sqlalchemy Uri的写法：

前缀是：postgresql+psycopg2，后缀是：username:password@host:port/database

点击“TEST CONNECTION”，提示测试连接成功，并且在最下方，列出了数据库dev中所有的表。

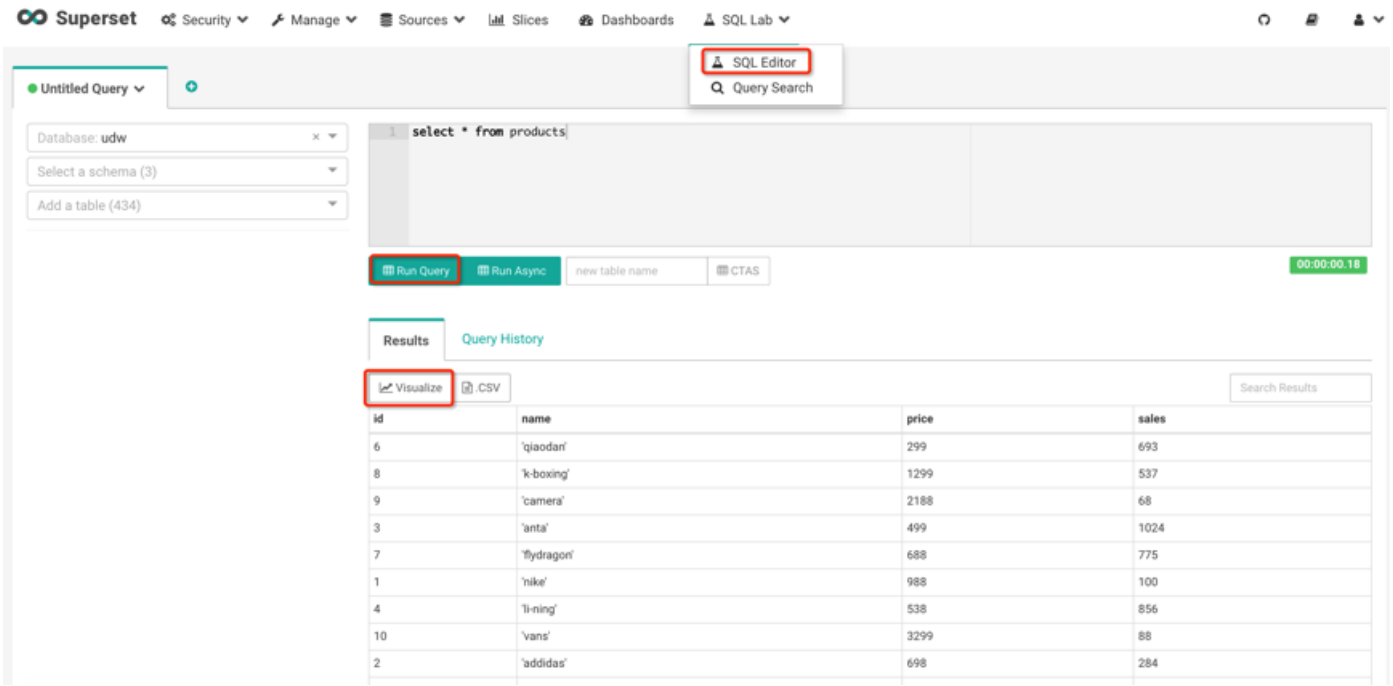
2）执行sql

SuperSet集成了一个SQL编辑器，点击“SQL Editor”，选择schema（不选的话是默认的schema，一般是public），选择一个表可以预览该表的数据，如下图所示：



Copyright © 2012-2021 UCloud 优刻得

选择Database，输入SQL，点击“Run Query”，获取查询结果（注意，此时不要选择schema，否则会报错），如下图所示：

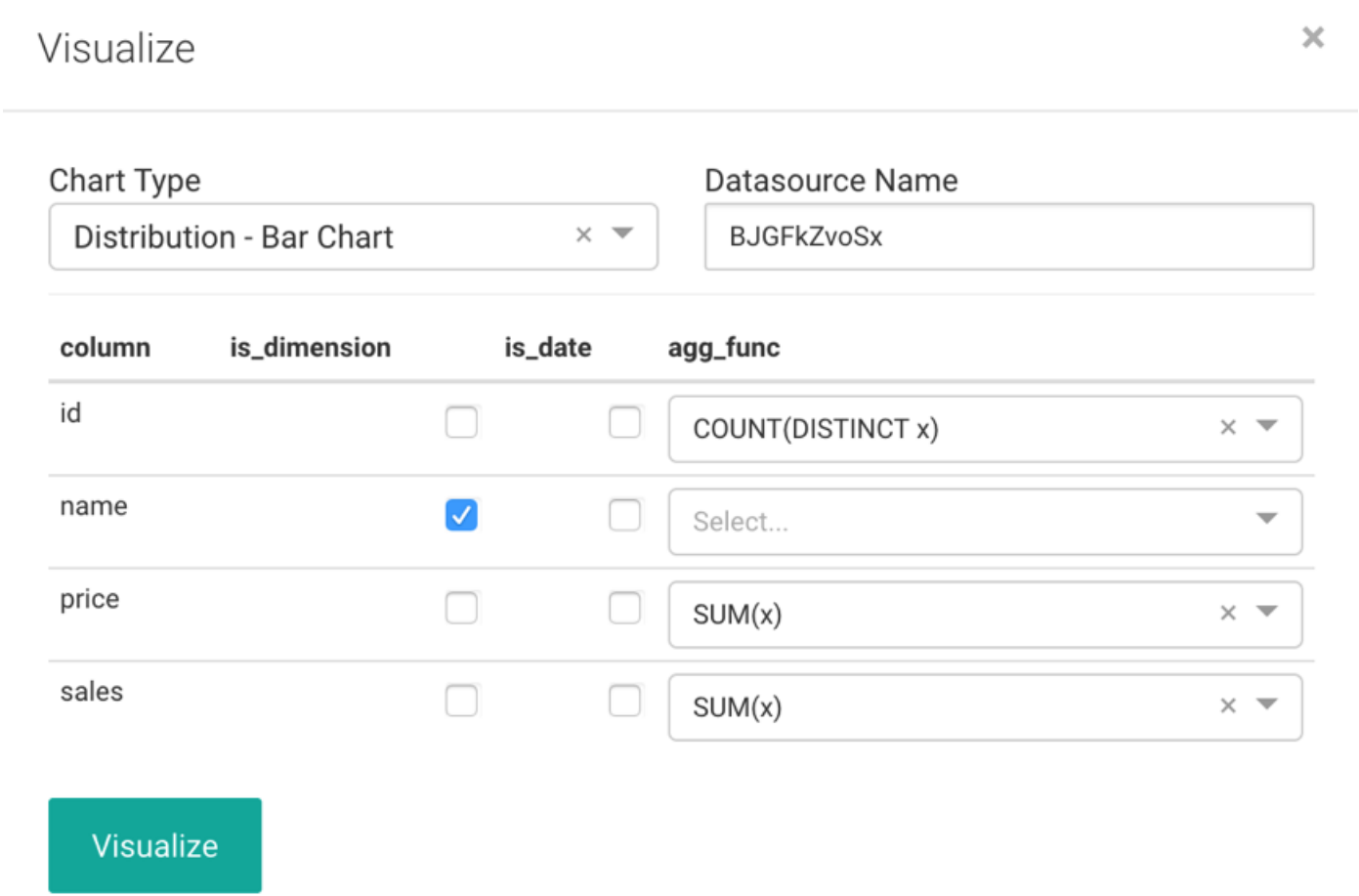


数据可视化

SuperSet支持十几种可视化图表，用于将查询返回的数据做可视化展示。

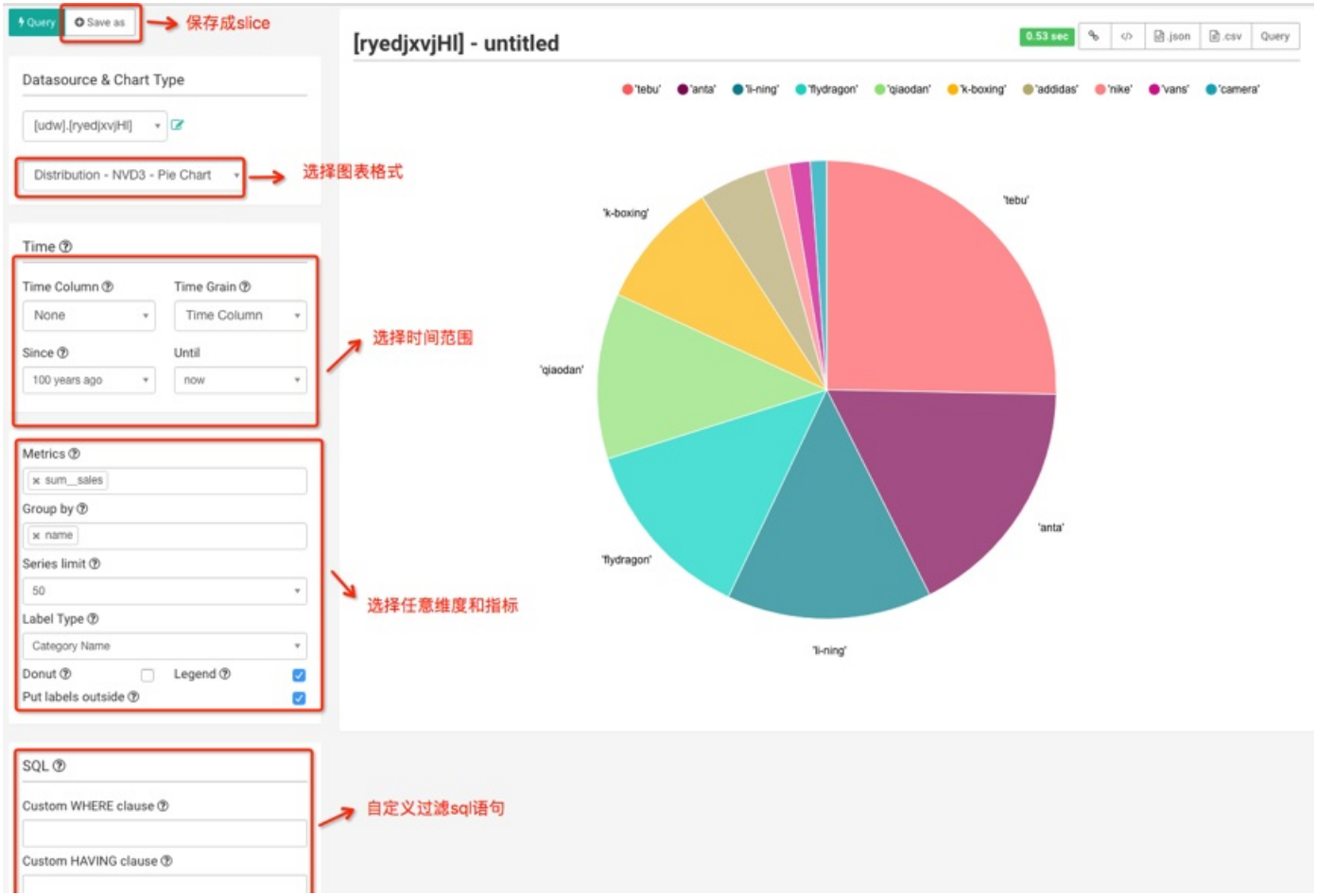
以上面的查询为例，点击“Visualize”进入可视化配置页面如下：

Copyright © 2012-2021 UCloud 优刻得



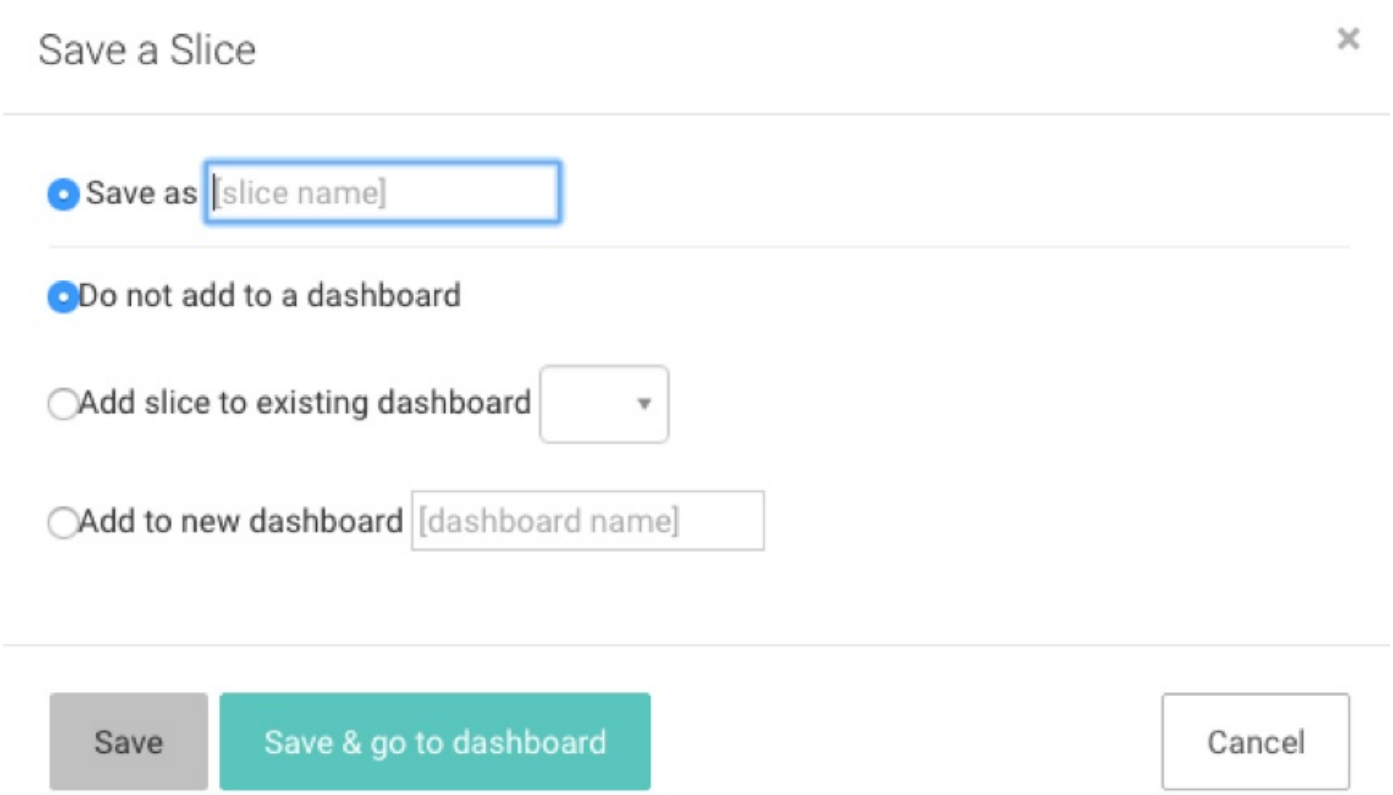
选择维度，勾选Sum、Min、Max、Count Distinct选项，点击“Visualize”则会生成相应的可视化页面。

Copyright © 2012-2021 UCloud 优刻得



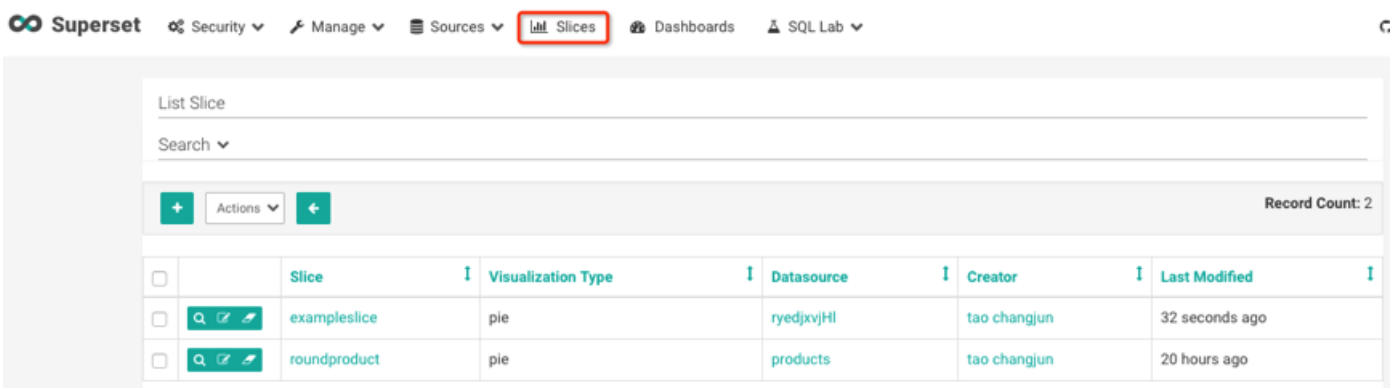
点击Save as可以将一个定制好的数据探索保存成Slice，多个Slice可以组成一个Dashboard。

Copyright © 2012-2021 UCloud 优刻得



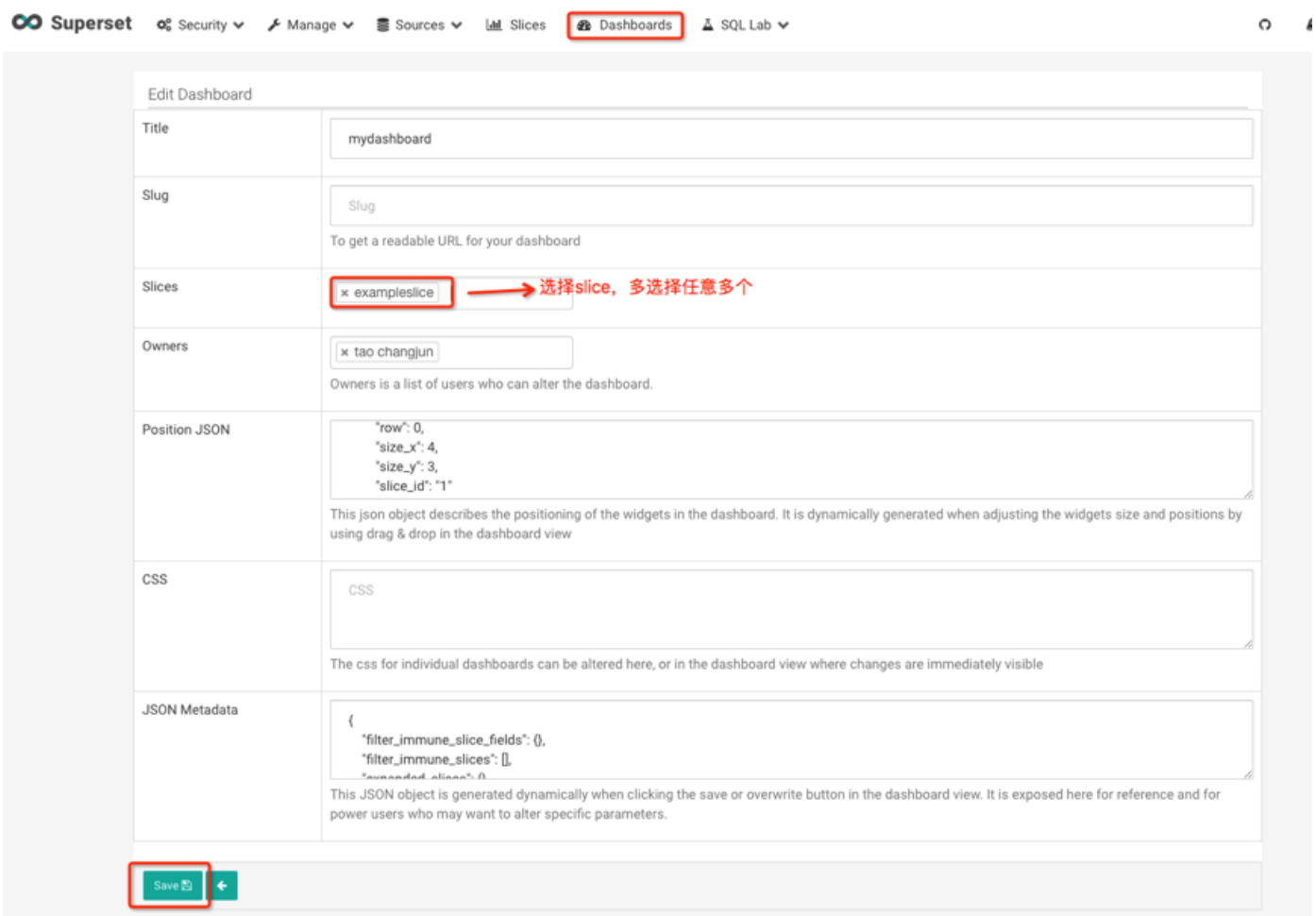
查看所有的slice:

Copyright © 2012-2021 UCloud 优刻得

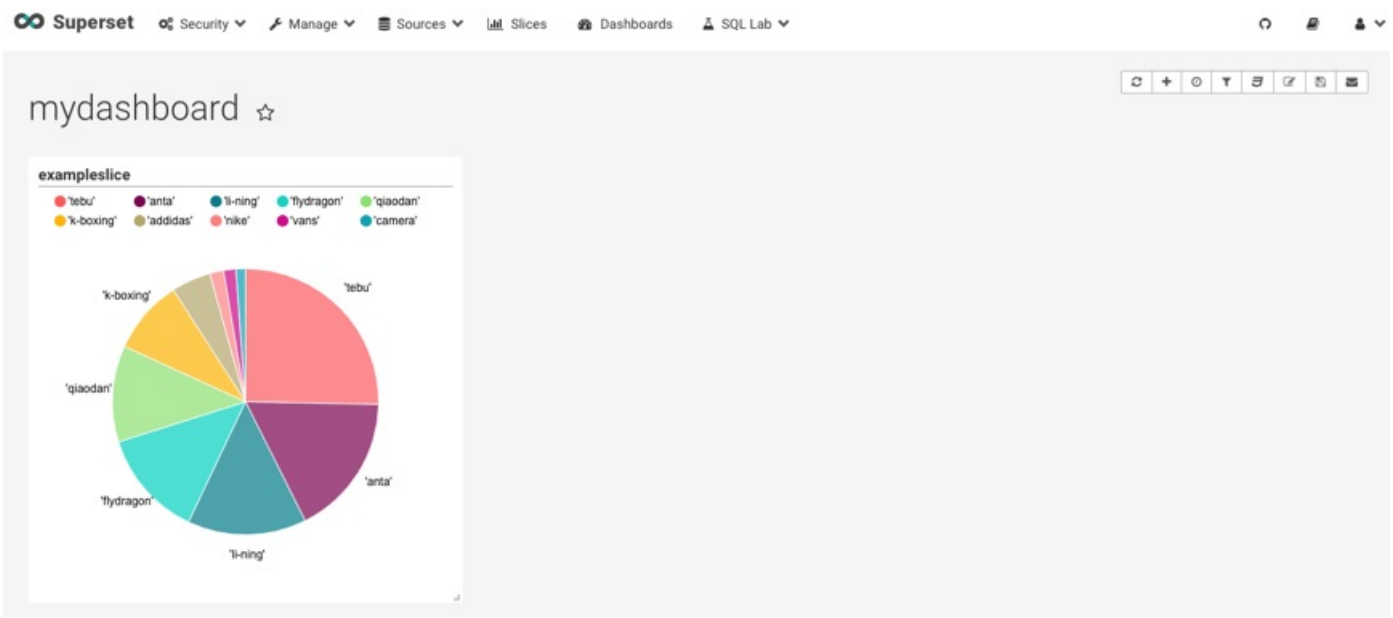


在添加Dashboard页面，指定包含哪些Slice，定制自己的Dashboard。

Copyright © 2012-2021 UCloud 优刻得



Copyright © 2012-2021 UCloud 优刻得



其中，每个Slice对应的模块，可以自由拖拽位置和大小，并保存整个Dashboard的布局。

关于superset的更多信息请参考：

http://airbnb.io/superset/

https://github.com/airbnb/superset

Copyright © 2012-2021 UCloud 优刻得

UDW 使用案例

案例一 利用 logstash+Kafka+UDW 对日志数据分析

Logstash 是目前比流行、使用较多的日志收集和管理系统，Kafka也是企业常用的分布式发布-订阅消息系统，UDW（UCloud Data Warehouse）是大规模并行处理数据仓库产品，下面介绍一些利用 logstash+Kafka+UDW 构建日志收集-存储-分析的全套解决方案。

Logstash收集日志到Kafka

1. 下载安装https://www.elastic.co/downloads/logstash
2. logstash依赖java环境、确保已经安装过java
3. 安装logstash-output-kafka插件
4. 配置logstash收集日志写入Kafka

参考配置如下（更多参数和含义请参考官方文档）：

Copyright © 2012-2021 UCloud 优刻得

5. 启动logstash收集日志到Kafka

执行 bin/logstash agent -f logstash-output-kafka.conf 发送消息到 Kafka

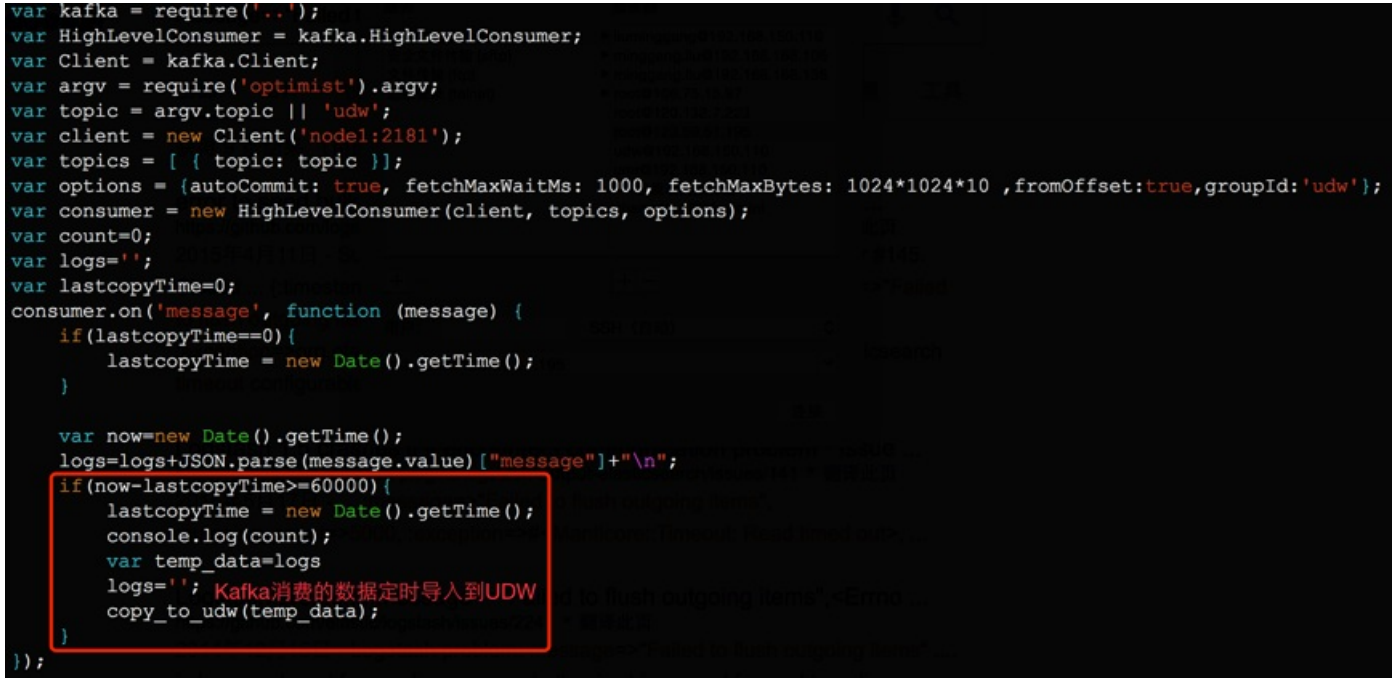
备注:我们除了用 logstash 收集日志到 kafka 之外,我们还可以使用 Flume 收集日志到 Kafka,也可以把 Spark、Storm 中的流式数据写入到 Kafka。更多 kafka 的使用请参考:

http://udw.cn-bj.ufileos.com/kafka%E4%BD%BF%E7%94%A8%E6%89%8B%E5%86%8C.pdf

Kafka数据写入到UDW

下面以Nodejs语言为例、其他语言也可以实现消费Kafka的数据并且写入到UDW。

Copyright © 2012-2021 UCloud 优刻得




```
var kafka = require('..');
var HighLevelConsumer = kafka.HighLevelConsumer;
var Client = kafka.Client;
var argv = require('optimist').argv;
var topic = argv.topic || 'udw';
var client = new Client('node1:2181');
var topics = [ { topic: topic }];
var options = {autoCommit: true, fetchMaxWaitMs: 1000, fetchMaxBytes: 1024*1024*10 ,fromOffset:true,groupId:'udw'};
var consumer = new HighLevelConsumer(client, topics, options);
var count=0;
var logs='';
var lastcopyTime=0;
consumer.on('message', function (message) {
    if(lastcopyTime==0){
        lastcopyTime = new Date().getTime();
    }

    var now=new Date().getTime();
    logs=logs+JSON.parse(message.value)["message"]+"\n";
    if(now-lastcopyTime>=60000){
        lastcopyTime = new Date().getTime();
        console.log(count);
        var temp_data=logs
        logs='';
        copy_to_udw(temp_data);
    }
});
```

如上所示，我们可以定时的把Kafka消费的数据导入到UDW，上面的示例是用nodejs实现的，需要依赖kafka-node，kafka-node的git地址：https://github.com/SOHU-Co/kafka-node

下面是通过copy的方式把从Kafka中消费的数据导入到UDW，为了加快数据插入UDW的速度，强烈建议用copy的方式导入数据到UDW。nodejs的copy数据到postgresql的使用方法请参考：https://github.com/brianc/node-pg-copy-streams。

Copyright © 2012-2021 UCloud 优刻得

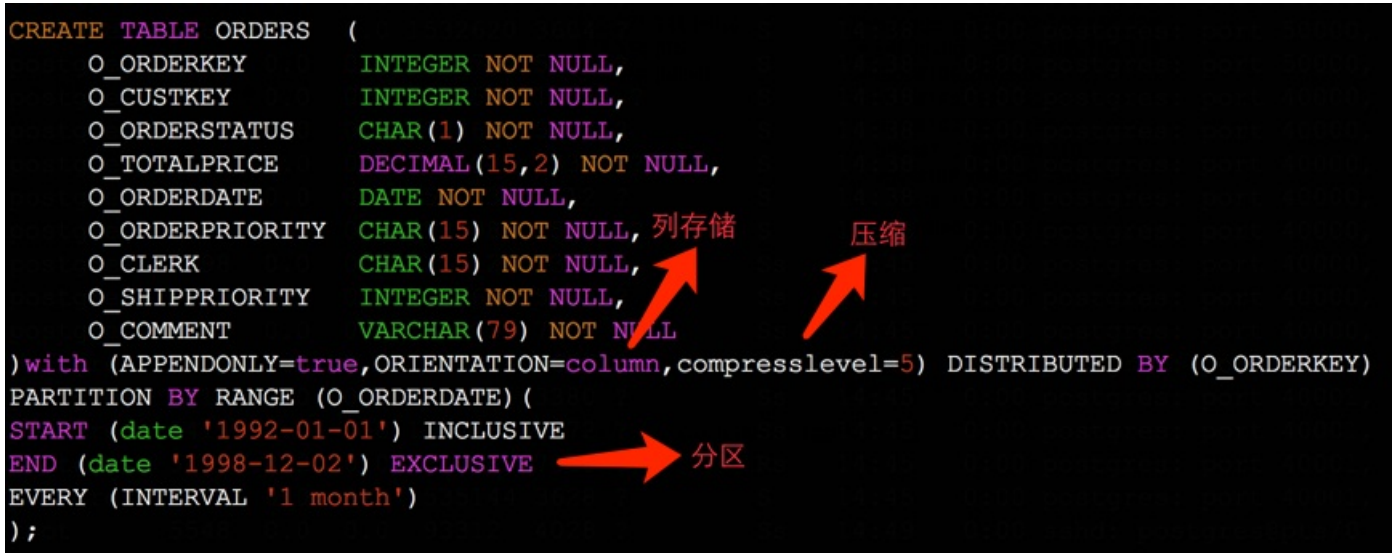
UDW的数据库和表格相关设计和操作，请参考：UDW开发指南

利用UDW进行数据分析

设计表格

在数据分析场景下、往往数据入库之后很少更新，我们创建一个列存储+压缩+分区表的表格。列存储、压缩、分区可以减少磁盘IO从而减少查询和分析时处理的数据量，大大提高查询效率。下面是orders表的建表实例。另外根据查询条件可以适当的建立索引从而进一步优化查询效率。

Copyright © 2012-2021 UCloud 优刻得



数据分析

UDW提供了标准的Postgresql的SQL，如下所示，我们可以通过SQL就很方便的实现对数据的分析。

Copyright © 2012-2021 UCloud 优刻得

```
dev=# select
dev-#     o_orderpriority,count(*) as order_count
dev-# from orders where o_orderdate >= date '1997-09-01'
dev-#     and o_orderdate < date '1997-12-01'
dev-#     and exists (select * from lineitem where
dev(#     l_orderkey = o_orderkey and l_commitdate < l_receiptdate)
dev-# group by o_orderpriority order by o_orderpriority;
 o_orderpriority | order_count
-----------------+-------------
 1-URGENT        |    10405603
 2-HIGH          |    10404353
 3-MEDIUM        |    10402877
 4-NOT SPECIFIED |    10401650
 5-LOW           |    10404247
(5 rows)
```

Copyright © 2012-2021 UCloud 优刻得

```
dev=# select
c_custkey,c_name,sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal,n_name,c_address,c_phone,c_comment
from
customer,orders,lineitem,nation
where c_custkey = o_custkey and l_orderkey = o_orderkey and o_orderdate >= date '1994-09-01'
and o_orderdate < date '1994-11-01'and l_returnflag = 'R' and c_nationkey = n_nationkey
group by c_custkey,c_name,c_acctbal,c_phone,n_name,c_address,c_comment
order by revenue desc limit 10;
 c_custkey |       c_name       |   revenue   | c_acctbal |         n_name           |               c_address                |     c_phone
   |                                                                 c_comment
-----------+--------------------+-------------+-----------+--------------------------+----------------------------------------+---------------
--+-----------------------------------------------------------------------------------------------------------------------
  62196187 | Customer#062196187 | 506645.6184 |    106.00 | KENYA                    | VBCxlVJmDKusQEeLx3icR                  | 24-288-629-898
7 | hins nag along the packages. ruthless ideas nag quickly about the carefully unusual theodolites
 144820735 | Customer#144820735 | 497591.6971 |    240.54 | VIETNAM                  | kHM0Tw EUj2QJj8ZIB                     | 31-419-746-431
8 | lithely. carefully express instructions affix? ironic, ironic accounts boost carefully. blithely pending a
  95615878 | Customer#095615878 | 490570.1309 |    843.91 | INDONESIA                | 72oa9MbgI3bQ,PwtoxBYtvq8frkT F         | 19-240-747-414
6 | side the blithely permanent deposits. carefully even pinto beans use blithe
  37414138 | Customer#037414138 | 477828.1366 |   2370.26 | MOZAMBIQUE               | nj74C 6FpEiBu,gftjXd9 upnc             | 26-101-273-573
9 | slyly ironic packages. slyly ironic forges haggle throughout the slyly ev
 135812596 | Customer#135812596 | 475054.7277 |   4717.02 | ARGENTINA                | 2bajtKKjkiDZodjcr                      | 11-881-781-903
2 | lithely ironic deposits nag carefully through the blithely ironic
  55497769 | Customer#055497769 | 472460.8830 |   8344.42 | EGYPT                    | qVo,CidfCSMwDaxC OkKhogO0qjgikSOkS5gUX | 14-940-655-481
0 | ep blithely. slyly ironic deposits grow furiously furiousl
 111617345 | Customer#111617345 | 470468.2668 |   9014.71 | JAPAN                    | QyexW73sNix6OzWISM0RENoD6,Om6oyjHgIi   | 22-855-886-736
0 | e between the furiously ironic account
  55410830 | Customer#055410830 | 470062.0874 |   5056.84 | JAPAN                    | 2ZNXCUd,He                             | 22-948-516-579
9 | ithely regular packages haggle boldly on the ironic, regular requests. requests use furiously ironic packages. reg
  28585922 | Customer#028585922 | 468597.7618 |   4130.66 | KENYA                    | lFI6DN,51WYlJdstU35oNsz2pfW            | 24-506-584-157
0 | ly according to the hockey players? furiously pending theodolites above the quickly express instructions boost abo
  49828648 | Customer#049828648 | 468307.2272 |   2359.89 | PERU                     | JmzOGCwy9KUWu                          | 27-741-255-124
8 | theodolites. regular, even dolphins detect across the final, bold packages. ironic requests after the ir
(10 rows)
```

数据可视化

为了方便UDW的查询数据可视化话，我们可以把UDW接入第三方的BI系统，请参考我们的文档：UDW接入第三方BI系统

案例二 基于UDW实现网络流分析

背景介绍

网络流分析主要包括对用户的网络流数据进行存储和多维度的分析两部分。用户的网络流的数据每天产生400G左右，数据保留10天。针对网络流数据的分析主要包含流量分析、包量分析、TCP延迟分析、HTTP状态码分析、TCP重传分析等。

Copyright © 2012-2021 UCloud 优刻得

数据存储

创建了一个10个节点的UDW集群，节点类型为ds1.large,磁盘类型SATA，大小为2000GB。(由于UDW集群的高可用方案，集群可用大小为10000GB)。

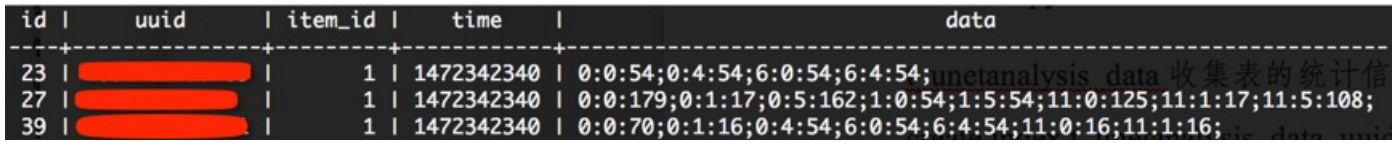
表格设计

采取列存储和压缩的追加表，分布键为id,根据time分区,时间间隔为1天。完整的建表语句如下所示：

```
create table t_unetanalysis_data (
id serial,uuid varchar(64) DEFAULT NULL,
item_id int DEFAULT NULL,
time int DEFAULT NULL,
data varchar(4000) DEFAULT NULL)
with (APPENDONLY=true, ORIENTATION=column, compresslevel=5) DISTRIBUTED BY (id)
partition by range(time) (
START (1469980800) END (1488297600) EVERY (86400)
);
```

其中，id 为记录序号，通过 serial(序列)实现自增；uuid 存储用户组织 ID 或者用户的 IP；item_id 为代表某种分析项的 id(分析项如IP流量、TCP包量、TCP重传率等)；time为时间戳；data为数据。

样本数据如下图所示：



id	uuid	item_id	time	data
23	[redacted]	1	1472342340	0:0:54;0:4:54;6:0:54;6:4:54;
27	[redacted]	1	1472342340	0:0:179;0:1:17;0:5:162;1:0:54;1:5:54;11:0:125;11:1:17;11:5:108;
39	[redacted]	1	1472342340	0:0:70;0:1:16;0:4:54;6:0:54;6:4:54;11:0:16;11:1:16;

Copyright © 2012-2021 UCloud 优刻得

根据查询需要创建索引如下：

```
create index t_unetanalysis_data_uuid_itemid_time on t_unetanalysis_data(uuid, item_id, time);
```

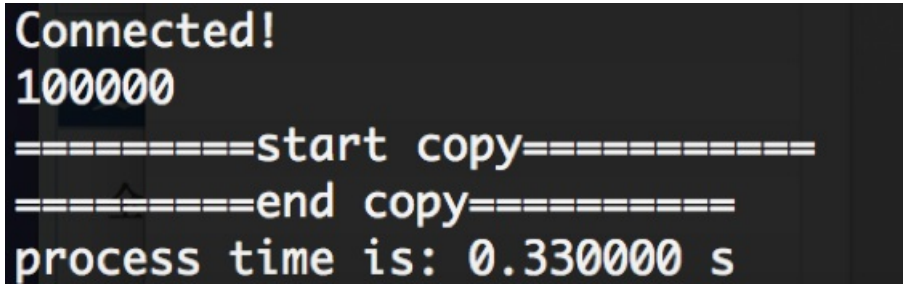
数据导入

为了加快数据插入UDW的速度，强烈建议用copy的方式将数据导入。示例代码如下图所示：

Copyright © 2012-2021 UCloud 优刻得

```python
# coding: utf-8

"""
python使用copy方式导入数据到UDW

"""
import psycopg2
import sys
import os
import time
from io import StringIO

# 连接数据库
conn = psycopg2.connect(database="dev", user="thomas", password="[redacted]", host="10.19.34.238", port="5432")

if(conn):
    print ("Connected!")

cursor = conn.cursor()

# 数据示例
list = []
for i in range(100000):
    list.append('941\t2\t1475884740\t0:0:2;0:1:2;10:0:1;10:1:1;15:0:1;15:1:1;\n')

print len(list)

records = ''
for record in list:
    records += record

print "=========start copy==========="
start = time.clock()
f = StringIO(records)
cursor.copy_from(f,'t_unetanalysis_data', columns=('uuid','item_id','time','data'))

conn.commit()
conn.close()
end = time.clock()
print "=========end copy==========="
print "process time is: %f s" % (end - start)
```

执行 python test_copy_to_udw.py，输出如下：



可以看到 copy 方式导入速度是非常快的。关于 python 的 copy_from 方法请参考：http://initd.org/psycopg/docs/cursor.html。

数据分析

在页面上点击分析指标，选择查询时间段，发送查询请求，后端收到请求后执行如下SQL查询：

```
SELECT time, data FROM t_unetanalysis_data
where uuid='xxx' and item_id=xxx and time>xxx and time<xxx;
```

例如，组织 id 为 50200021 的用户查询一个星期内 ip 的出量（item_id为17），

```
SELECT time, data FROM t_unetanalysis_data
where uuid= '50200021' and item_id=17 and time > 1481472000 and time < 1482076800;
```

耗时平均为 260ms。

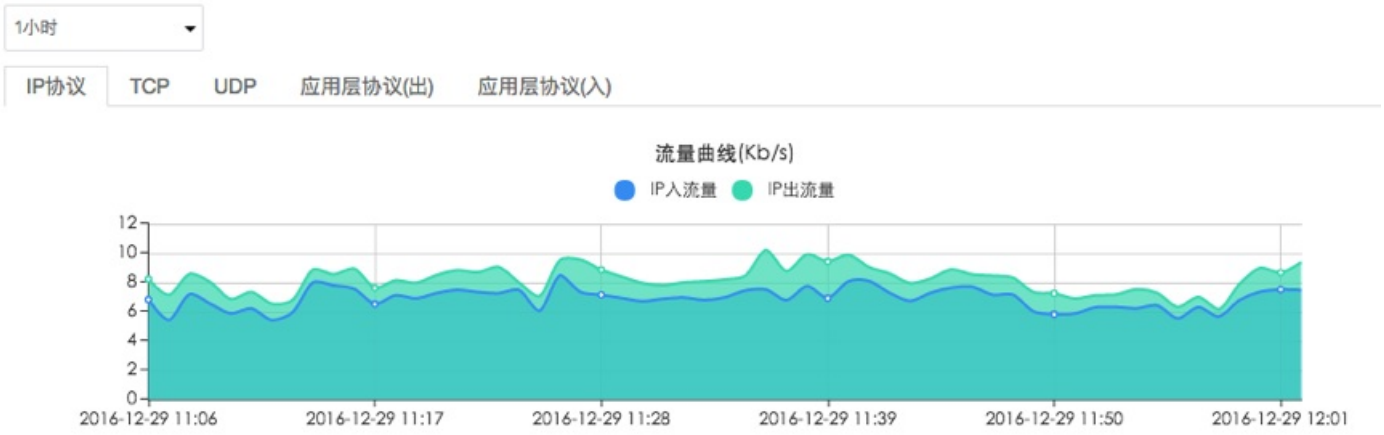
时间范围为 1 天的查询耗时平均为 120ms。

将查询到的数据返回给前端，前端解析数据，绘出图形，展示在页面上。

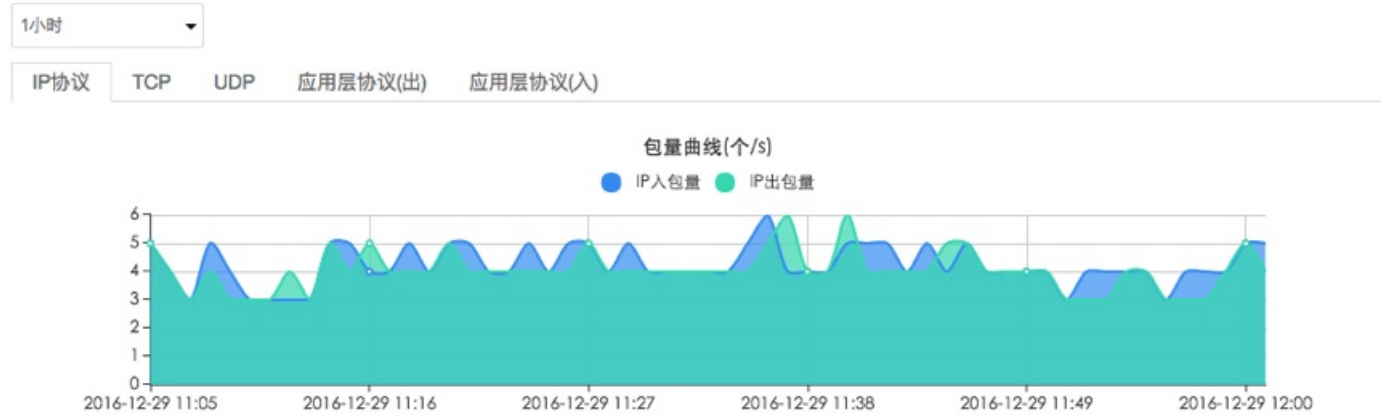
数据可视化

Copyright © 2012-2021 UCloud 优刻得

流量分析：



包量分析：



TCP延迟分析：

Copyright © 2012-2021 UCloud 优刻得

Copyright © 2012-2021 UCloud 优刻得

PXF 扩展

在 5.17 及以上版本的 Udw 集群中默认安装了 PXF 扩展服务，Udw 集群可以通过 PXF 服务访问 HDFS, Hive, HBase 等外部数据，具体使用可以查询对应版本的 GreenPlum PXF 官方文档。

使用 PXF 服务访问外部数据时，需要进行一些有关外部数据的配置，我们在控制台提供了配置上传的功能。如果需要访问 Hadoop 相关的外部数据，必须上传对应 Hadoop 集群的 core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml 配置文件，如果还需要额外访问 Hive 或者 HBase 数据，则需要上传 hive-site.xml 或者 hbase-site.xml 配置文件。

因为配置文件中一般以域名/主机名表示各节点的访问地址，所以还需要额外上传包含 Hadoop 集群各节点的域名/主机名与 IP 对应关系的 hosts 文件，我们会将这个文件中的内容添加到 Udw 集群的 hosts 文件当中。(请尽量确保上传的 hosts 文件只包含集群各节点的 IP 信息，以免造在更新 Udw hosts 文件后造成错误)

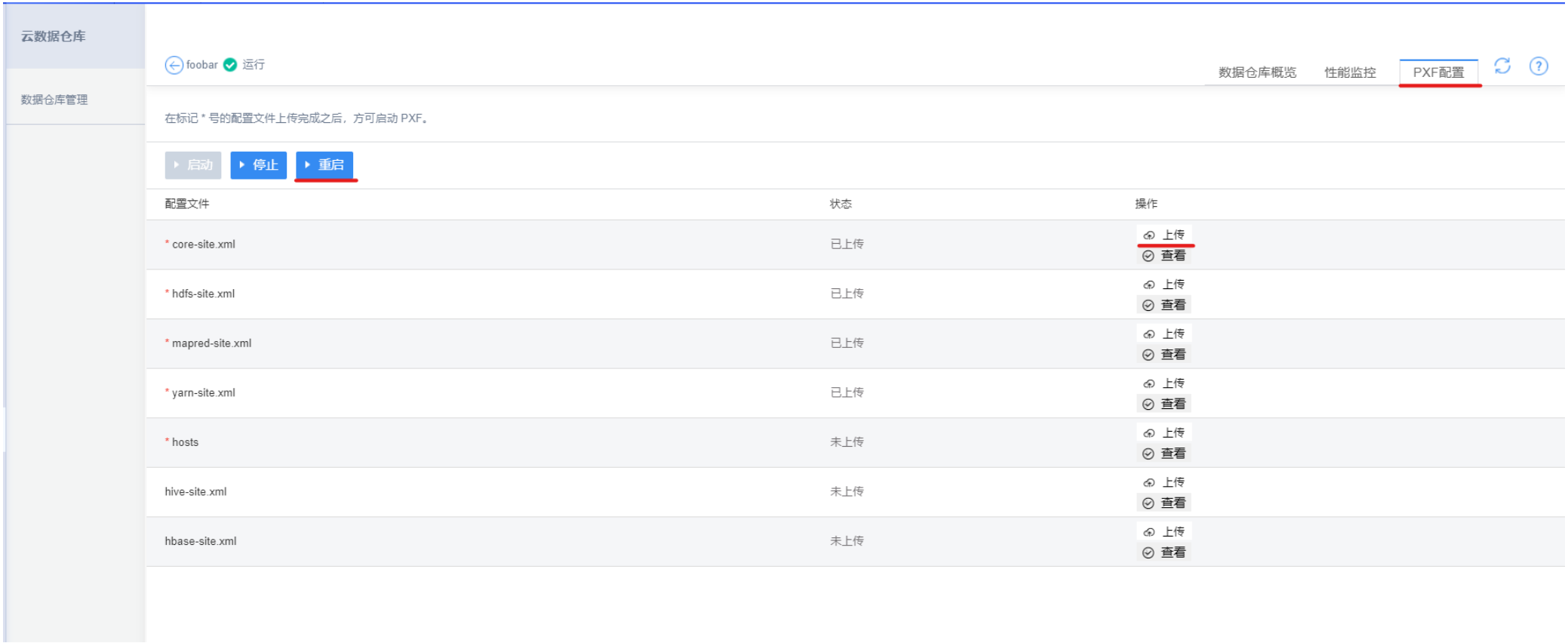
上传配置之后，需要重启 PXF 服务使配置生效，控制台上提供了 PXF 服务的 停止/开启/重启 等操作功能。

配置 PXF 服务

在控制台 PXF 配置页面，有对应的文件列表与上传功能，点击 上传 并选择对应的 Hadoop 集群配置文件或者 hosts 文件，进行配置上传。

配置上传完成后，点击 重启 按钮让配置生效。

Copyright © 2012-2021 UCloud 优刻得



创建 EXTENSION

连接 Greenplum 服务，创建 pxf EXTENSION：

```
CREATE EXTENSION pxf;
```

Copyright © 2012-2021 UCloud 优刻得

授权给需要使用 pxf 外部表的用户：

```
GRANT SELECT ON PROTOCOL pxf TO pan;
GRANT INSERT ON PROTOCOL pxf TO pan;
```

读写 HDFS

在 Hadoop 集群中创建测试数据：

```
$ hdfs dfs -mkdir -p /data/pxf_examples

$ echo 'Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67' > /tmp/pxf_hdfs_simple.txt

$ hdfs dfs -put /tmp/pxf_hdfs_simple.txt /data/pxf_examples/
```

连接 Greenplum 服务，创建外部表访问 HDFS 数据：

```
CREATE EXTERNAL TABLE pxf_hdfs_textsimple(location text, month text, num_orders int, total_sales float8)
LOCATION ('pxf://data/pxf_examples/pxf_hdfs_simple.txt?PROFILE=hdfs:text')
FORMAT 'TEXT' (delimiter=E',');
```

Copyright © 2012-2021 UCloud 优刻得

查询外部表数据：

```
postgres=# select * from pxf_hdfs_textsimple;
location | month | num_orders | total_sales
-----------+-------+------------+-------------
Prague | Jan | 101 | 4875.33
Rome | Mar | 87 | 1557.39
Bangalore | May | 317 | 8936.99
Beijing | Jul | 411 | 11600.67
(4 rows)
```

创建可写外部表并插入数据：

```
CREATE WRITABLE EXTERNAL TABLE pxf_hdfs_writabletbl (location text, month text, num_orders int, total_sales float8)
LOCATION ('pxf://data/pxf_examples/pxfwritable_hdfs_textsimple?PROFILE=hdfs:text&COMPRESSION_CODEC=org.apache.hadoop.io.compress.GzipCodec')
FORMAT 'TEXT' (delimiter=':');

INSERT INTO pxf_hdfs_writabletbl VALUES ( 'Cleveland', 'Oct', 3812, 96645.37 );
INSERT INTO pxf_hdfs_writabletbl VALUES ( 'Frankfurt', 'Mar', 777, 3956.98 );
```

在 HDFS 中查看数据：

```
$ hdfs dfs -cat /data/pxf_examples/pxfwritable_hdfs_textsimple/* | zcat
Cleveland:Oct:3812:96645.37
Frankfurt:Mar:777:3956.98
```

Copyright © 2012-2021 UCloud 优刻得

PXF 默认会以 postgres 这个用户名访问 HDFS，所以如果遇到权限错误，请将要访问/写入的 HDFS 目录授权给 postgres 用户

访问 Hive

准备 Hive 测试数据：

```
$ echo 'Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67
San Francisco,Sept,156,6846.34
Paris,Nov,159,7134.56
San Francisco,Jan,113,5397.89
Prague,Dec,333,9894.77
Bangalore,Jul,271,8320.55
Beijing,Dec,100,4248.41
' > /tmp/pxf_hive_datafile.txt
```

创建 Hive 表并写入数据：

```
CREATE TABLE sales_info (location string, month string,
```

Copyright © 2012-2021 UCloud 优刻得

```
number_of_orders int, total_sales double)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS textfile;

LOAD DATA LOCAL INPATH '/tmp/pxf_hive_datafile.txt'
INTO TABLE sales_info;
```

连接 Greenplum 服务，创建 PXF 外部表：

```
CREATE EXTERNAL TABLE salesinfo_hiveprofile(location text, month text, num_orders int, total_sales float8)
LOCATION ('pxf://default.sales_info?PROFILE=Hive')
FORMAT 'custom' (formatter='pxfwritable_import');
```

通过 PXF 外部表查询 Hive 数据：

```
postgres=# SELECT * FROM salesinfo_hiveprofile;
location | month | num_orders | total_sales
---------------+-------+------------+-------------
Prague | Jan | 101 | 4875.33
Rome | Mar | 87 | 1557.39
Bangalore | May | 317 | 8936.99
Beijing | Jul | 411 | 11600.67
San Francisco | Sept | 156 | 6846.34
Paris | Nov | 159 | 7134.56
San Francisco | Jan | 113 | 5397.89
```

Copyright © 2012-2021 UCloud 优刻得

```
Prague | Dec | 333 | 9894.77
Bangalore | Jul | 271 | 8320.55
Beijing | Dec | 100 | 4248.41
| | |
(11 rows)
```

访问 HBase

如果需要支持 filter pushdown 特性，请根据 Udw 集群版本，下载对应的 pxf-hbase-*.jar，并复制到 HBase 集群每个节点的 HBASE_CLASSPATH 目录

udw-6.2.1(pxf-hbase-5.10.1.jar) udw-5.17(pxf-hbase-5.2.1.jar)

准备测试数据，创建 HBase 表 order_info，该表有 2 个 column families：product, shipping_info：

```
create 'order_info', 'product', 'shipping_info'
```

在 order_info 表中插入测试数据：

```
put 'order_info', '1', 'product:name', 'tennis racquet'
put 'order_info', '1', 'product:location', 'out of stock'
put 'order_info', '1', 'shipping_info:state', 'CA'
put 'order_info', '1', 'shipping_info:zipcode', '12345'
put 'order_info', '2', 'product:name', 'soccer ball'
put 'order_info', '2', 'product:location', 'on floor'
put 'order_info', '2', 'shipping_info:state', 'CO'
```

Copyright © 2012-2021 UCloud 优刻得

```
put 'order_info', '2', 'shipping_info:zipcode', '56789'
put 'order_info', '3', 'product:name', 'snorkel set'
put 'order_info', '3', 'product:location', 'warehouse'
put 'order_info', '3', 'shipping_info:state', 'OH'
put 'order_info', '3', 'shipping_info:zipcode', '34567'
```

以直接映射的方式创建 PXF 外部表，以 HBase 的 `<column-family>:<column-qualifier>` 为外部表的列名：

```
CREATE EXTERNAL TABLE orderinfo_hbase ("product:name" varchar, "shipping_info:zipcode" int)
LOCATION ('pxf://order_info?PROFILE=HBase')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

通过 PXF 外部表访问 HBase 数据：

```
postgres=# select * from orderinfo_hbase;
product:name | shipping_info:zipcode
---------------+----------------------
tennis racquet | 12345
soccer ball | 56789
snorkel set | 34567
(3 rows)
```

Copyright © 2012-2021 UCloud 优刻得

使用 pg_dump 迁移数据

安装 greenplum-db-clients

为了获取 pg_dump 工具，需要安装 greenplum-db-clients，安装方法可以查看 https://gpdb.docs.pivotal.io/6-2/client_tool_guides/installing.html，根据指引下载适用的安装包并进行安装。

下面以CentOS 7.x和Greenplum 6.2.1版本示例

1)下载 6.2.1 的Greenplum客户端安装包 greenplum-db-clients-6.2.1-rhel7-x86_64.rpm，执行命令 yum install greenplum-db-clients-6.2.1-rhel7-x86_64.rpm 进行安装。

2)安装之后将 /usr/local/greenplum-db-clients/bin 加入系统 PATH 环境变量。具体操作如下：

在/etc/profile 文件中增加 export PATH=$PATH:/usr/local/greenplum-db-clients/bin,并执行source /etc/profile

使用 pg_dump 导出数据

pg_dump 完整说明可以查看文档 https://www.postgresql.org/docs/9.4/app-pgdump.html，常用参数如下：

- -d dbname, --dbname=dbname：指定 database
- -h host, --host=host：指定连接地址/IP
- -p port, --port=port：指定访问端口
- -U username, --username=username：指定访问用户名

Copyright © 2012-2021 UCloud 优刻得

- -W, --password：提示输入密码
- -a, --data-only：只导出数据
- -s, --schema-only：只导出定义
- -n schema, --schema=schema：指定 schema，可以使用多次 -n 指定多个 schema
- -t table, --table=table：指定 table，可以使用多次 -t 指定多个 table

示例如下：

导出 public schema 下所有表定义到 public_all.sql 文件：

```
pg_dump -d postgres -h 10.9.141.93 -p 5432 -U admin -W \
-s -t "public.*" -f public_all.sql
```

导出 public schema 下 src table 的定义到 public_src.sql 文件：

```
pg_dump -d postgres -h 10.9.141.93 -p 5432 -U admin -W \
-s -t "public.src" -f public_src.sql
```

导出 public schema 下所有表数据到 public_all_data.sql：

```
pg_dump -d postgres -h 10.9.141.93 -p 5432 -U admin -W \
-a -t "public.*" -f public_all_data.sql
```

导出 public schema 下 src table 的数据到 public_src_data.sql 文件：

Copyright © 2012-2021 UCloud 优刻得

```
pg_dump -d postgres -h 10.9.141.93 -p 5432 -U admin -W \
-a -t "public.src" -f public_src_data.sql
```

使用 psql 重建数据

重新创建 public schema 下所有表：

```
psql -d postgres -h 10.9.52.209 -p 5432 -U admin -W -f public_all.sql
```

重新写入 public schema 下所有表的数据：

```
psql -d postgres -h 10.9.52.209 -p 5432 -U admin -W -f public_all_data.sql
```

Copyright © 2012-2021 UCloud 优刻得

利用 hdfs 外部表迁移数据

利用外部 hdfs 作为中转，将原 greenplum 集群中的表数据迁移至目的 greenplum 集群，分为以下 4 步：

1. 在原 greenplum 集群中创建 hdfs pxf 可写外部表

假设需要迁移数据的表为 src，其结构如下所示：

```
postgres=# \d+ src;
Table "public.src"
Column | Type | Modifiers | Storage | Stats target | Description
------------+-----------------+-----------+----------+--------------+-------------
location | text | | extended | |
month | text | | extended | |
num_orders | integer | | plain | |
total_sales | double precision | | plain | |
Distributed by: (location)
```

根据其表中字段以及定义，创建对应的 hdfs pxf 可写外部表：

```
CREATE WRITABLE EXTERNAL TABLE pxf_hdfs_src (location text, month text, num_orders int, total_sales float8)
LOCATION ('pxf://data/udw/pxf_hdfs_src?PROFILE=hdfs:text&COMPRESSION_CODEC=org.apache.hadoop.io.compress.GzipCodec')
```

Copyright © 2012-2021 UCloud 优刻得

```
FORMAT 'TEXT' (delimiter=':');
```

2. 将原 greenplum 集群表数据写入 hdfs

把 hdfs 作为中转存储，将 src 表的数据写入 hdfs：

```
insert into pxf_hdfs_src select * from src;
```

3. 在目的 greenplum 集群中创建 hdfs pxf 可读表

根据原 greenplum 集群中 pxf 外部表在目的集群创建相同结构与位置的 hdfs pxf 只读表：

```
CREATE EXTERNAL TABLE pxf_hdfs_src(location text, month text, num_orders int, total_sales float8)
LOCATION ('pxf://data/udw/pxf_hdfs_src?PROFILE=hdfs:text&COMPRESSION_CODEC=org.apache.hadoop.io.compress.GzipCodec')
FORMAT 'TEXT' (delimiter=E':');
```

4. 从 hdfs 外部表中读取数据并写入目的 greenplum 集群

```
insert into src select * from pxf_hdfs_src;
```

Copyright © 2012-2021 UCloud 优刻得

FAQs

创建好数据仓库之后怎么连接到UDW?

请参考帮助文档“快速上手->连接数据仓库”部分介绍。https://docs.ucloud.cn/udw/gettingstart

UDW支持从mysql导入数据吗?

支持。我们提供了一种从MySQL向UDW导入数据的工具。这个工具可以全量、也可以增量从MySQL导入到UDW单线程可以到达每秒4000-8000条,多线程可以达到4w-10w条每秒。具体使用请参考mysql数据导入到udw.pdf

HDFS/Hive与UDW之间可以导入导出数据吗?

可以。为了方便udw和hdfs/hive之间的数据导入和导出,我们提供个两种方案:

1. 用sqoop实现hdfs和udw直接的数据导入导出,使用方法请参考:hdfs和hive中数据导入导出到udw
2. 创建hdfs外部表,使用方法请参考:创建hdfs外部表

UDW中怎么kill掉正在执行的SQL语句?

Copyright © 2012-2021 UCloud 优刻得

查看哪些SQL语句正在执行，语句如下：

```
SELECT datname,procpid,query_start, current_query,waiting,client_addr FROM pg_stat_activity WHERE waiting='t';
```

datname表示数据库名 procpid表示当前的SQL对应的PID query_start表示SQL执行开始时间 current_query表示当前执行的SQL语句 waiting表示是否正在执行，t表示正在执行，f表示已经执行完成 client_addr表示客户端IP地址

找出procpid，假设为1234。然后执行

```
SELECT pg_terminate_backend(1234);
```

如何通过外网访问UDW?

udw 默认是通过内网访问的，为了数据安全性，尽量不要通过外网访问 UDW，如果有访问外网的需求请参考：

前提：有一台可以访问 udw 的 uhost，并且这台 uhost 上可以访问外网 ip。

例如：uhost 内网 ip 是 10.10.0.9 外网 IP 是 192.168.120.110，udw 的访问ip 是 10.10.10.1，我们在 uhost 机器上建立 ssh 隧道即可通过 192.168.120.110访问 udw。在 uhost 机器上执行如下命令：

```
ssh -C -f -N -g -L 5432:10.10.10.1:5432 root@10.10.0.9
```

备注：请注意开放外网防火墙端口 5432（也可以把 udw 端口映射到 uhost上其他端口上），网络防火墙配置请参考：https://docs.ucloud.cn/unet/firewall/introduction

Copyright © 2012-2021 UCloud 优刻得

节点扩容时数量有没有什么限制?

集群采用 Grouped Mirror 分布策略,因此新增节点数必须大于等于 2,以确保新增 Primary Segment 的 Mirror Segment 不在同一台机器上。

Copyright © 2012-2021 UCloud 优刻得

数据仓库价格

数据仓库价格根据节点类型及配置不同，详细价格如下。

机型	名称	配置	月价格（元/月）	年价格（元/年）
计算密集型	dc1.large	2核 12G 300G(SSD)	700	7000
存储密集型	ds1.large	4核 24G 2000G(SATA)	1500	15000
高IO独享型	dc2.large	8核 32G 900G(SSD)	2300	23000
RSSD云盘型	rc1.large	4核 32G 600G(SSD)	1500	15000
RSSD云盘型	rc1.xlarge	8核 64G 1200G(SSD)	3000	30000

Copyright © 2012-2021 UCloud 优刻得