openGauss的AI特性

实验指导书

2022秋

华为技术有限公司

目录

[1 实验环境介绍 4](#_Toc119658948)

[1.1 实验介绍 4](#_Toc119658949)

[1.1.1 关于本实验 4](#_Toc119658950)

[1.1.2 读者知识背景 4](#_Toc119658951)

[1.1.3 实验设备介绍 4](#_Toc119658952)

[2 准备工作：ECS弹性云服务器购买 5](#_Toc119658953)

[2.1 实验介绍 5](#_Toc119658954)

[2.1.1 关于本实验 5](#_Toc119658955)

[2.1.2 实验目的 5](#_Toc119658956)

[2.2 登录华为云 5](#_Toc119658957)

[2.3 准备虚拟私有云VPC环境 6](#_Toc119658958)

[2.4 购买云服务器ECS并登录 10](#_Toc119658959)

[2.5 思考题 16](#_Toc119658960)

[3 关卡一：openGauss数据库的编译和安装 17](#_Toc119658961)

[3.1 实验介绍 17](#_Toc119658962)

[3.1.1 关于本实验 17](#_Toc119658963)

[3.1.2 实验目的 17](#_Toc119658964)

[3.2 实验任务及步骤 17](#_Toc119658965)

[3.2.1 编译前准备 17](#_Toc119658966)

[3.2.2 数据库安装编译 22](#_Toc119658967)

[3.2.3 关卡验证 26](#_Toc119658968)

[3.2.4 思考题 26](#_Toc119658969)

[4 关卡二：openGauss数据导入及基本操作 27](#_Toc119658970)

[4.1 实验介绍 27](#_Toc119658971)

[4.1.1 关于本实验 27](#_Toc119658972)

[4.1.2 实验目的 27](#_Toc119658973)

[4.2 实验任务及步骤 27](#_Toc119658974)

[4.2.1 数据初始化 27](#_Toc119658975)

[4.2.2 关卡验证 31](#_Toc119658976)

[4.2.3 思考题 31](#_Toc119658977)

[5 关卡三：openGauss的AI4DB特性应用 32](#_Toc119658978)

[5.1 实验介绍 32](#_Toc119658979)

[5.1.1 关于本实验 32](#_Toc119658980)

[5.1.2 实验目的 32](#_Toc119658981)

[5.2 实验任务及步骤 32](#_Toc119658982)

[5.2.1 将X-Tuner 安装到系统中 32](#_Toc119658983)

[5.2.2 使用X-Tuner进行参数优化 35](#_Toc119658984)

[5.2.3 Index-advisor：索引推荐 38](#_Toc119658985)

[5.2.4 关卡验证 41](#_Toc119658986)

[5.2.5 思考题 41](#_Toc119658987)

[6 关卡四：openGauss的DB4AI特性应用 42](#_Toc119658988)

[6.1 实验介绍 42](#_Toc119658989)

[6.1.1 关于本实验 42](#_Toc119658990)

[6.1.2 实验目的 42](#_Toc119658991)

[6.2 实验任务及步骤 42](#_Toc119658992)

[6.2.1 利用DB4AI原生AI引擎训练并预测模型 42](#_Toc119658993)

[6.2.2 关卡验证 46](#_Toc119658994)

[6.2.3 思考题 46](#_Toc119658995)

[7 清理工作：资源释放 47](#_Toc119658996)

[7.1 实验介绍 47](#_Toc119658997)

[7.1.1 关于本实验 47](#_Toc119658998)

[7.1.2 实验目的 47](#_Toc119658999)

[7.2 删除弹性云服务器及相关资源 47](#_Toc119659000)

[7.3 思考题 48](#_Toc119659001)

[8 缩略语表 49](#_Toc119659002)

# 实验环境介绍

## 实验介绍

### 关于本实验

openGauss是关系型数据库，采用客户端/服务器，单进程多线程架构，支持单机和一主多备部署方式，备机可读，支持双机高可用和读扩展。

本实验主要内容为openEuler弹性云服务上单机安装部署openGauss数据库，并进行简单的数据库相关操作；完成数据库相关操作后，进行openGauss的AI4DB的特性实验，提升数据库性能工作；最后完成openGauss的DB4AI特性实验（选做），完成分类测试和回归测试算法。

### 读者知识背景

* 具有数据库知识背景，熟悉数据库常用基本操作（查询、插入、删除、更新等）；
* 了解基本Linux知识，熟悉常用软件如vi等；
* 熟悉华为云，会申请、释放云资源，会基本操作。

### 实验设备介绍

本实验环境为华为云环境，需要购买openEuler弹性云服务。

* 设备介绍

为了满足openGauss安装部署实验需要，建议每套实验环境采用以下配置：

实验软件配套关系

|  |  |
| --- | --- |
| 软件名称 | 软件版本 |
| Linux操作系统 | openEuler 20.3 LTS |
| 工具 | WinSCP, TPCH |
| Python | Python 3.7.X |

# 准备工作：ECS弹性云服务器购买

## 实验介绍

### 关于本实验

本实验通过在华为云上购买弹性云服务器ECS，了解弹性云服务器ECS配置流程，掌握弹性云服务器ECS的连接和操作。

### 实验目的

* 掌握ECS弹性云服务器购买流程；
* 掌握ECS弹性云服务器配置参数。

## 登录华为云

进入华为云官网。

华为云官网：https://www.huaweicloud.com/，进入华为云官网，点击登录。



输入账号名和密码，点击登录。



如果还没有注册，点击免费注册，按步骤进行注册后进行登录。

## 准备虚拟私有云VPC环境

在后续创建ECS时，选择该步骤中所配置的VPC、子网和安全组。

登录华为云账号，在虚拟私有云VPC页面下，点击“立即使用”。

<https://www.huaweicloud.com/product/vpc.html> 

在网络控制台界面中，点击“创建虚拟私有云”。



填写如下配置信息，然后点击“立即创建”。

基本信息

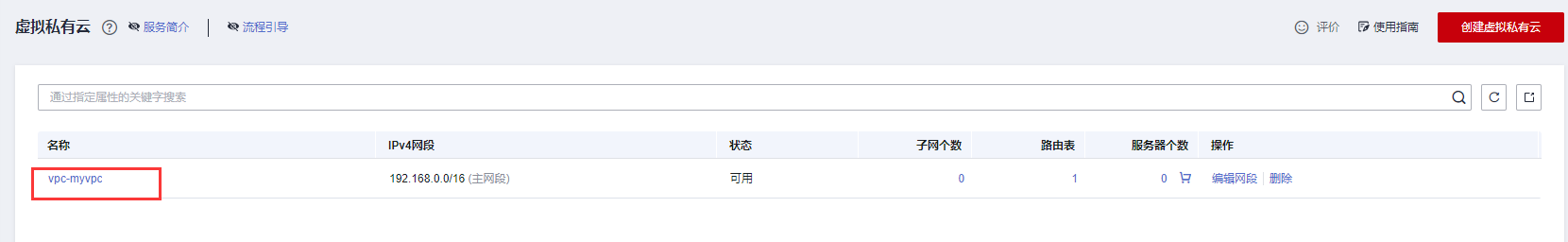
* 区域：华北-北京四
* 名称：vpc-myvpc
* 网段：默认

子网配置

* 可用区：可用区1
* 名称：subnet-myvpc
* 子网网段：默认



返回到网络控制台，可看见VPC与子网已创建。



在“安全组”页面中找到default安全组。点击“配置规则”。

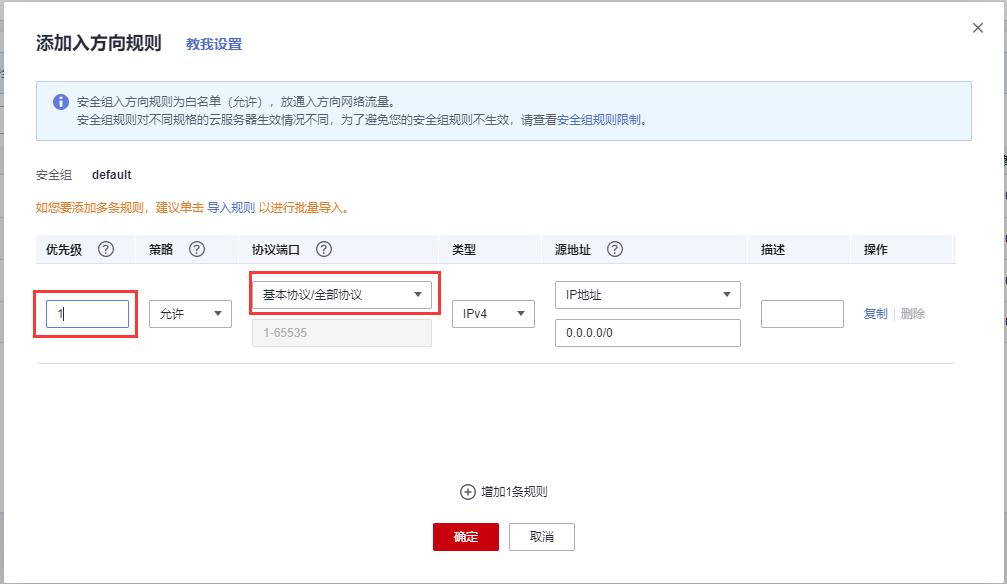




选择“入方向规则”页签，点击“添加规则”。



协议端口选择“基本协议/全部协议”，优先级填“1”，点击“确定”，完成安全组规则的添加。



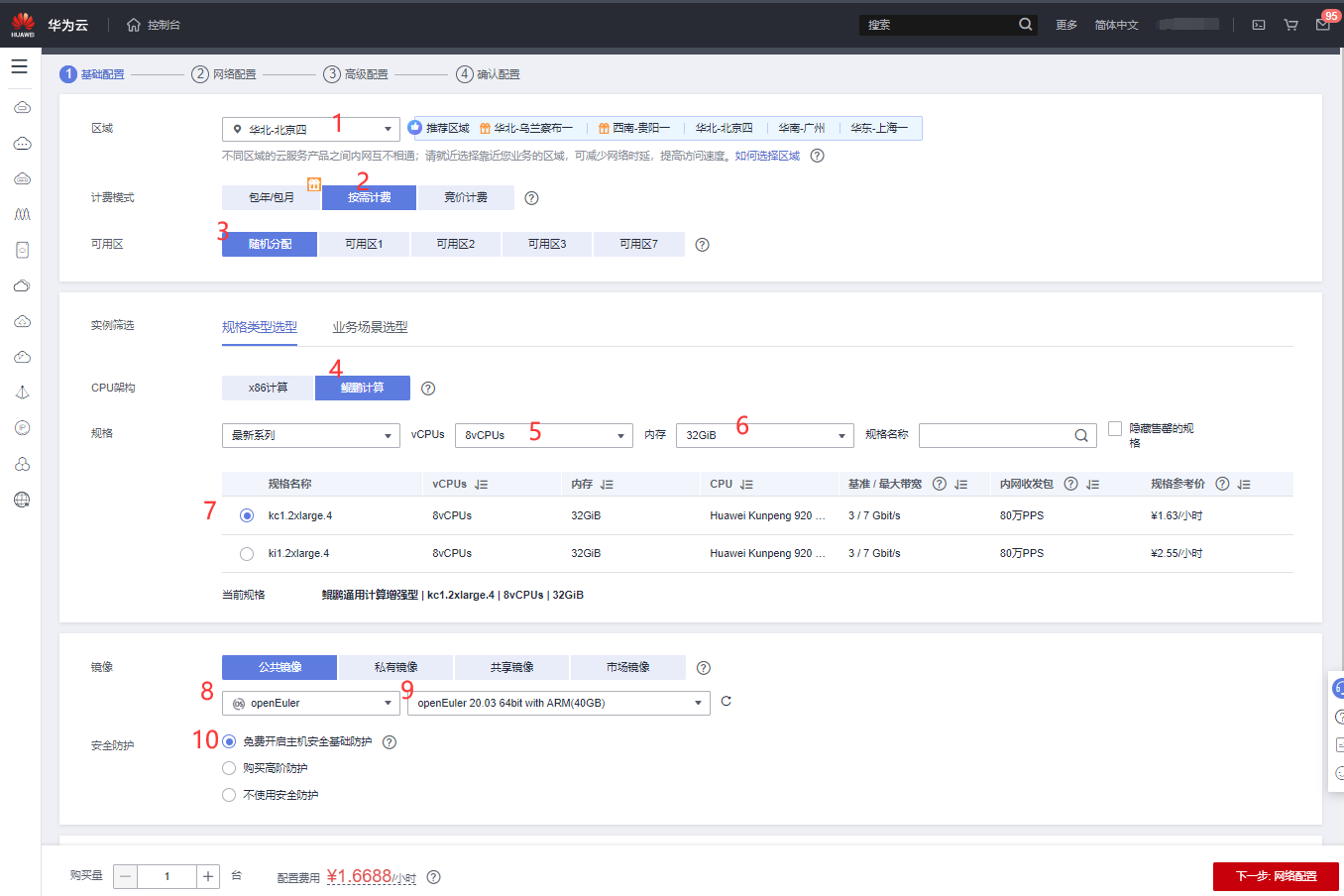
## 购买云服务器ECS并登录

登录华为云账号，在弹性云服务器ECS页面下，点击“立即购买”。

<https://www.huaweicloud.com/product/ecs.html> 

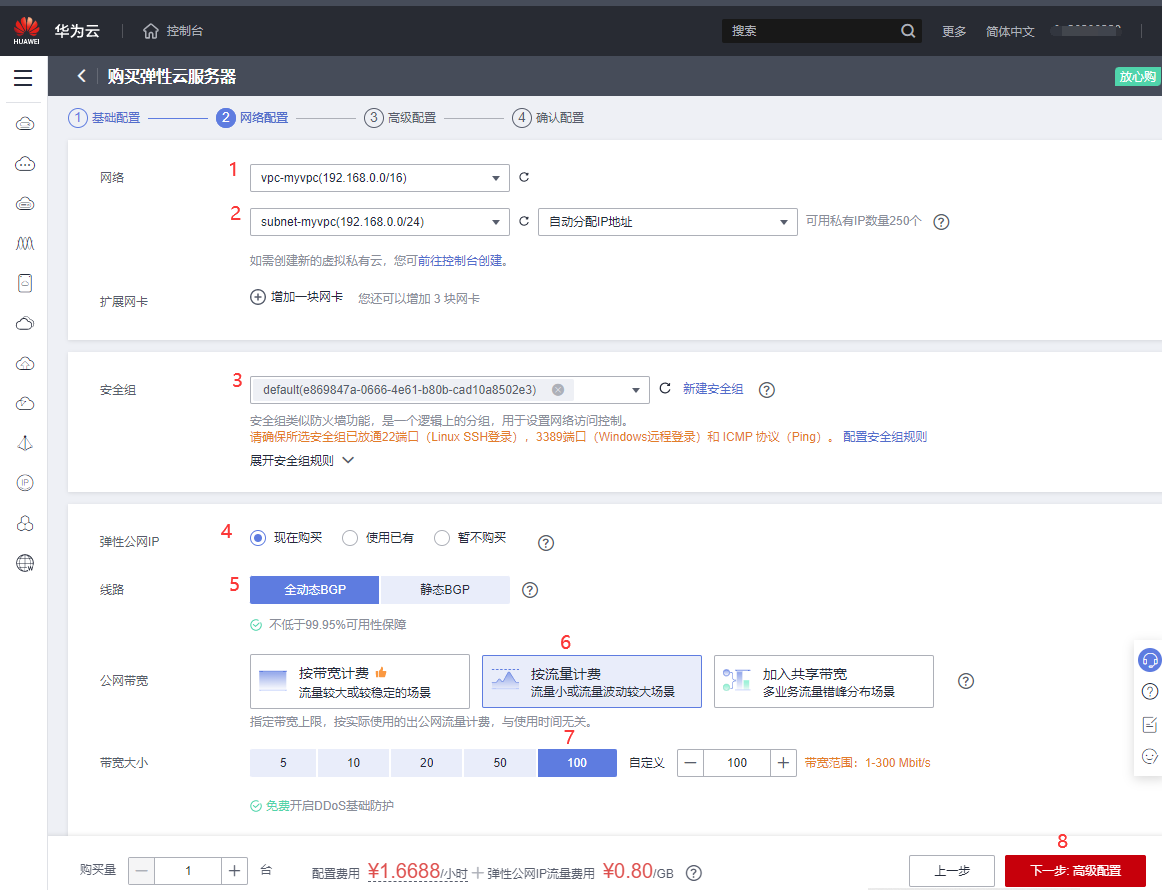
填写如下基础配置信息，然后点击“下一步”。

* 区域：**华北-北京四**
* 计费模式：按需计费
* 可用区：随机分配
* CPU架构：鲲鹏计算
* 规格：鲲鹏通用计算增强型 | kc1.2xlarge.4 | 8vCPUs | 32GB
* 镜像：公共镜像openEuler 20.03 64bit with ARM(40GB)
* 系统盘：通用型SSD | 40G



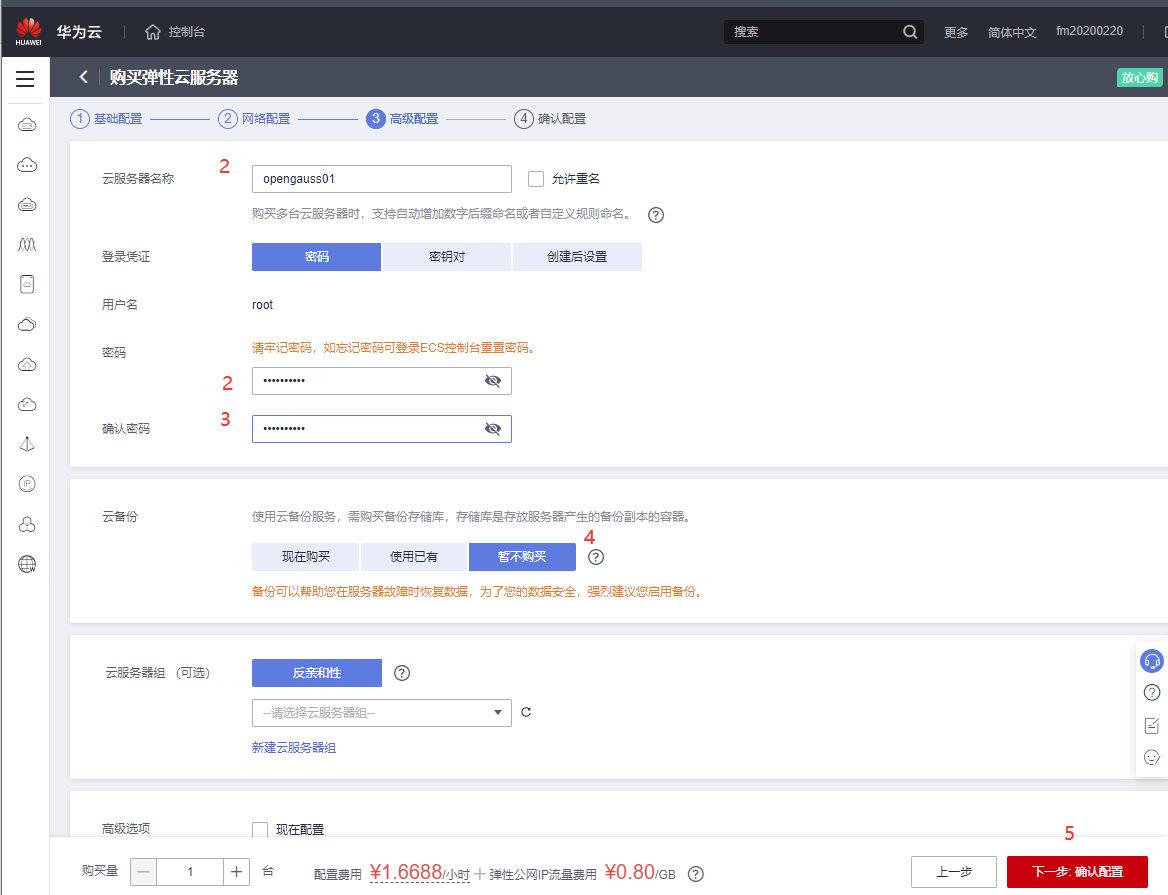
填写如下网络配置信息，然后点击“下一步”。

* 网络：选择在2.3节创建的网络和子网，如vpc-myvpc和subnet-myvpc
* 安全组： default
* 弹性公网IP：现在购买
* 规格：全动态BGP
* 计费方式：按流量计费
* 带宽：100 Mbit/s

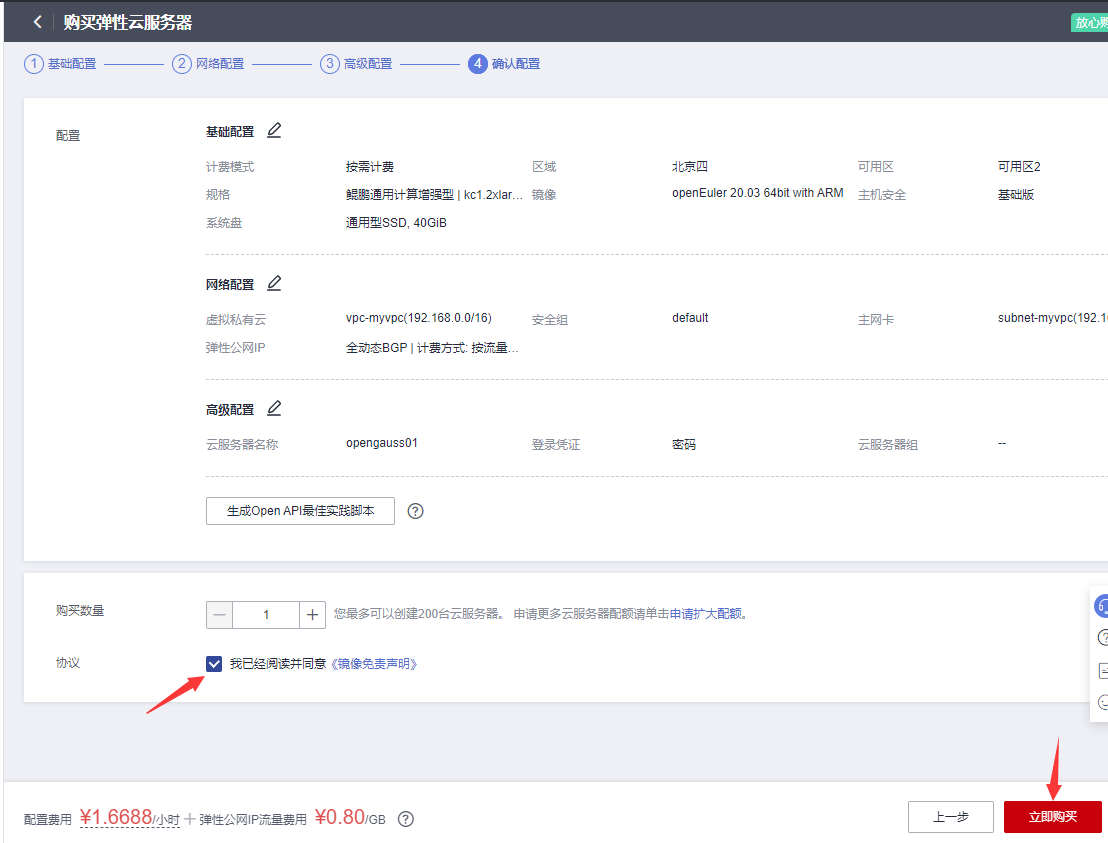


填写如下高级配置信息，然后点击“下一步”。

* 云服务器名称：opengauss01（服务器名称建议不要使用下划线）
* 登录凭证：密码
* 设置密码【请记住密码，后面登录要用到】
* 云备份：暂不购买



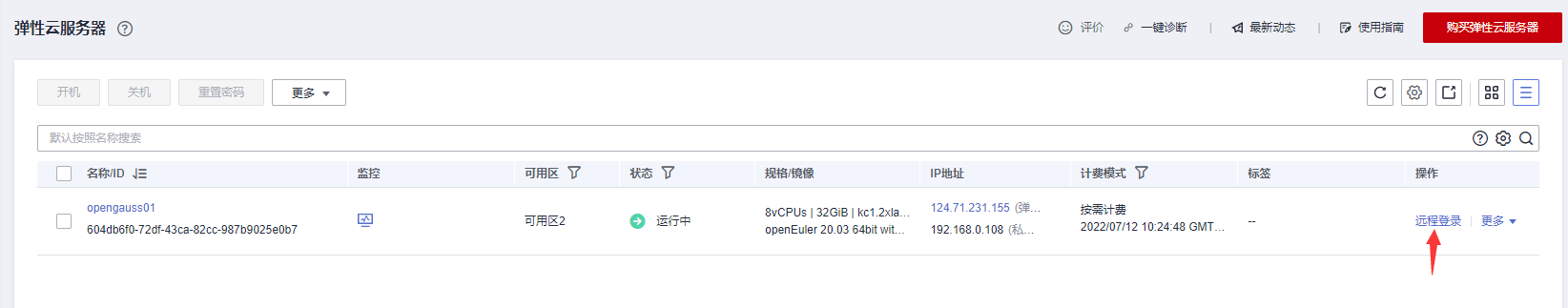
在确认配置界面，勾选“我已经阅读…….”后，点击“立即购买”。



购买完成后，在云服务器控制台可以看到正常运行中的ECS。

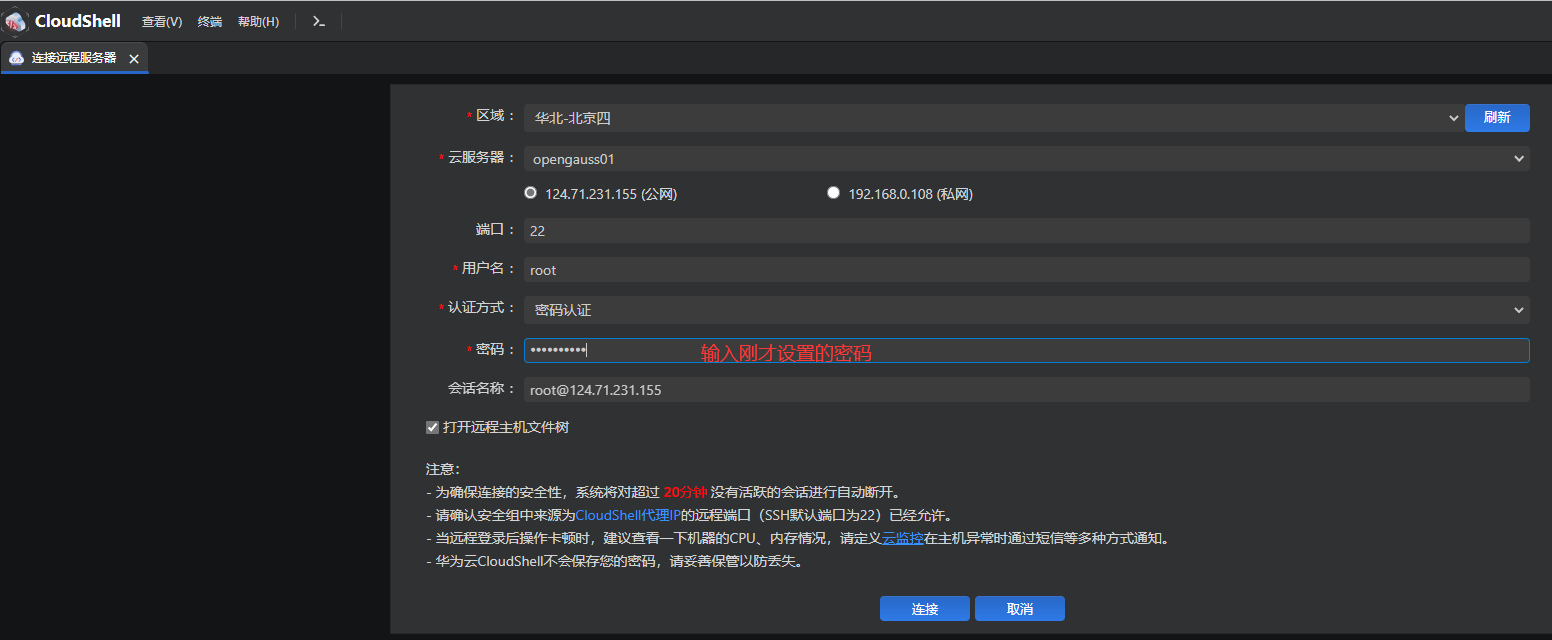


当状态显示“运行中”，可刷新列表，点击“远程登录”。

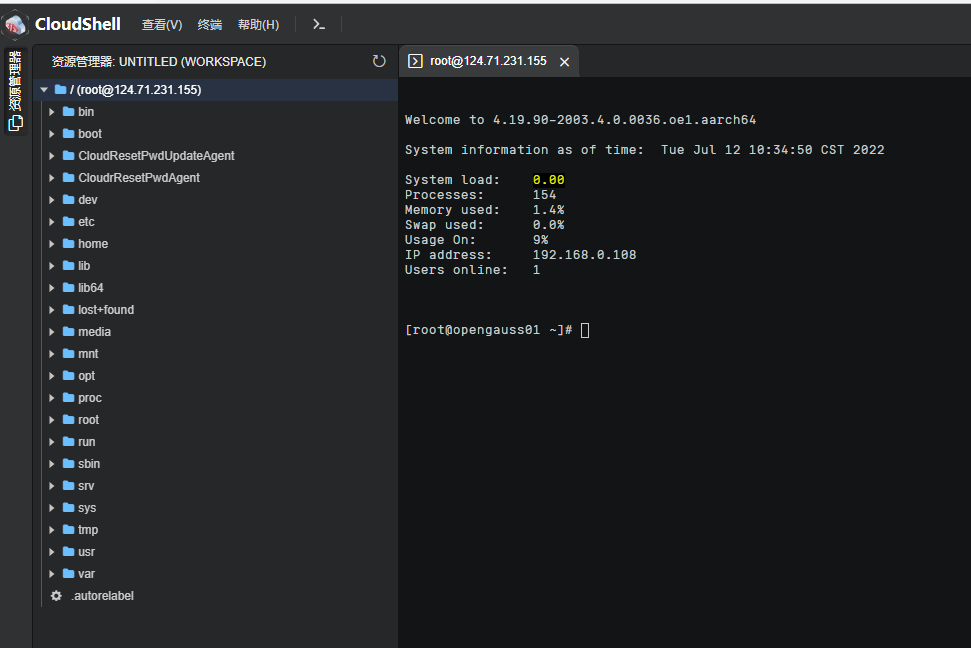




用root密码，点击“连接”。



登录成功：



# 关卡一：openGauss数据库的编译和安装

## 实验介绍

### 关于本实验

本实验通过第三方工具，完成openGauss的编译和安装。

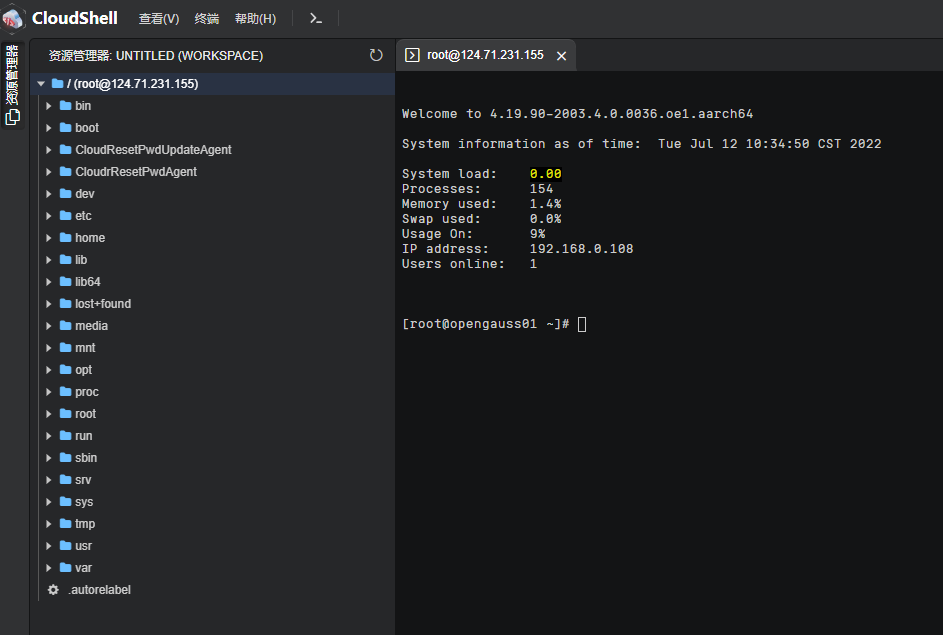
### 实验目的

* 掌握openGauss编译和安装。
* 掌握数据库初始化。

## 实验任务及步骤

### 安装准备

用root用户名，使用之前设置的密码登录ECS。



上传安装脚本到服务器。

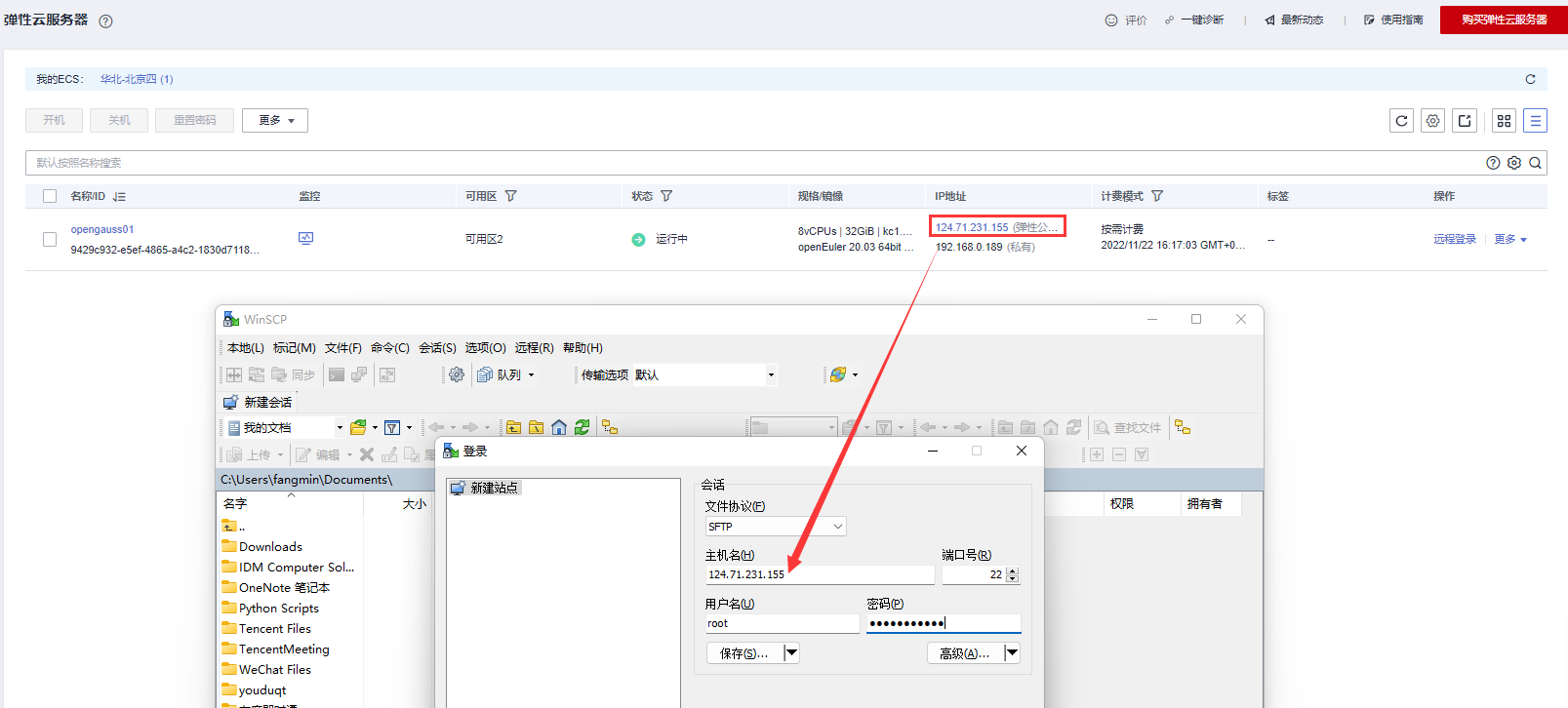
1. 安装winscp工具，winscp安装包已随堂下发，默认安装即可。

\* WinSCP是一款支持SSH（Secure SHell）的SCP（Secure CoPy）文件传输软件。如果已经有类似软件可以不装。

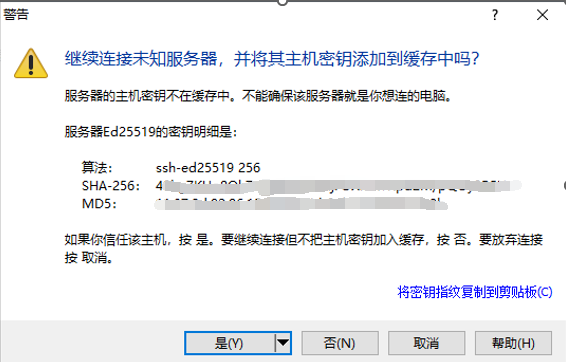


1. 配置winscp站点。

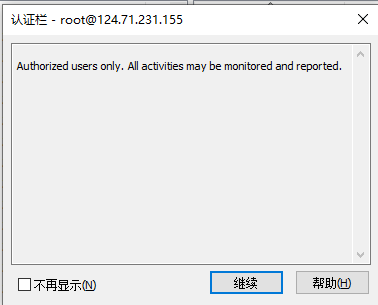
主机名从弹性云服务器列表中复制。用户名、密码就是刚才新建的弹性云服务器的用户名、密码。



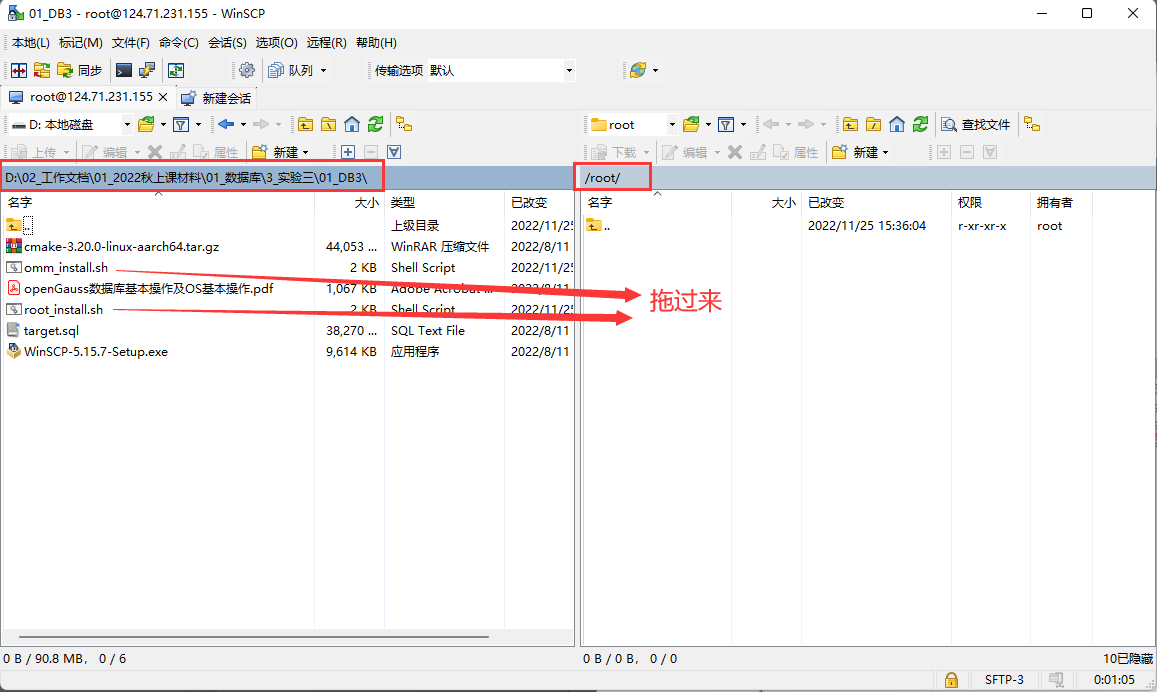
下面点击“是”：



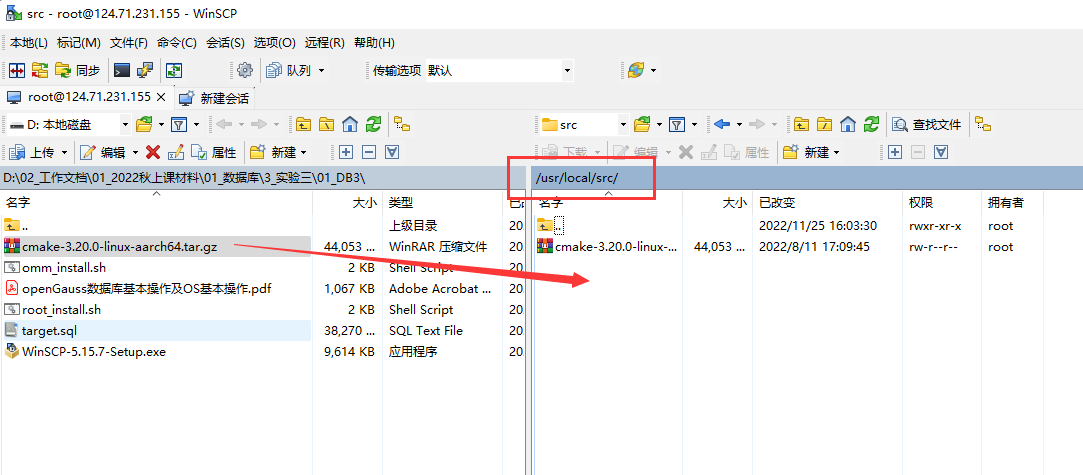
下面点击“继续”：



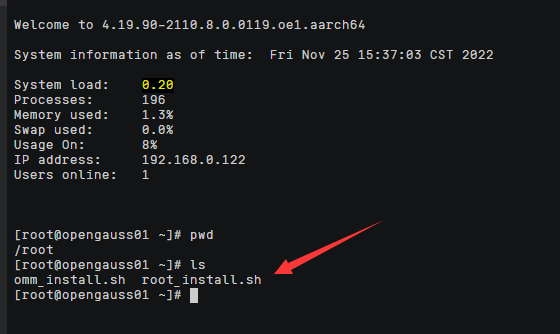
1. 将root\_install.sh和omm\_install.sh，上传至服务器/root/下。



1. 将cmake-3.20.0-linux-aarch64.tar.gz上传至服务器/usr/local/src下。



1. 传输完毕后，可以回到CloudShell上看到这几个文件。



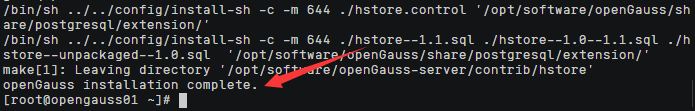
开始执行编译安装脚本

[root@opengauss01 ~]# **sh root\_install.sh**

脚本运行过程中需要新建一个omm用户，初始化omm用户的密码。【请记住密码】



这里传输数据和安装需要十五分钟左右，请耐心等待。你可以利用这个时间看看后面的操作。



下面开始初始化数据库，先切换到omm用户。

[root@opengauss01 bin]# **su - omm**

在初始化数据库时，需要设置数据库密码，并且要使用复杂密码，如下命令。

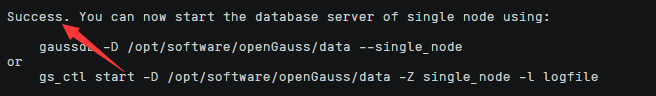
gs\_initdb -D $PGDATA --nodename=**hostname** --locale="en\_US.UTF-8" -Atrust -w **{password}**

实际使用中，将{password}部分进行替换。

例如（此处只是作为举例，建议设置为复杂密码，注意密码要用英文的单引号括起来）：

[omm@opengauss01 ~]$ **gs\_initdb -D $PGDATA --nodename=opengauss01 --locale="en\_US.UTF-8" -Atrust -w 'Huawei#!13'**

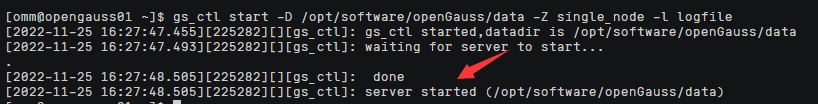
显示结果如下，表示初始化成功：



这一步如果报错，请确认一下自己当前是omm用户执行命令。

启动数据库。

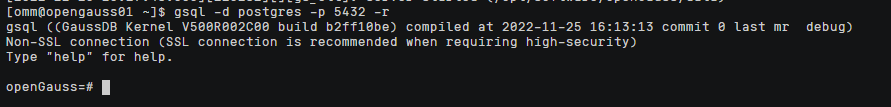
[omm@opengauss01 ~]$ **gs\_ctl start -D /opt/software/openGauss/data -Z single\_node -l logfile**



数据库启动成功。

数据库登录。

[omm@opengauss01 openGauss-server]$ **gsql -d postgres -p 5432 -r**



修改omm账号密码（可选步骤）。

openGauss=# **ALTER USER omm identified by 'Huawei@13' replace 'Huawei#!13';**

说明：ALTER USER omm identified by '新密码' replace '原密码';

查询数据库版本。

openGauss=# **select version();**

version

-------------------------------------------------------------------------------------------------------------------------

(GaussDB Kernel V500R002C00 build b2ff10be) compiled at 2021-11-22 16:34:45 commit 0 last mr debug on aarch64-unknown-linux-gnu, compiled by g++ (GCC) 7.3.0, 64-bit

(1 row)

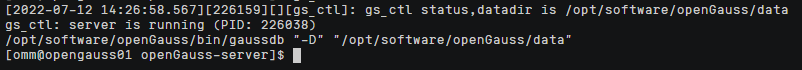
退出数据库。

openGauss=# **\q**

### 关卡验证

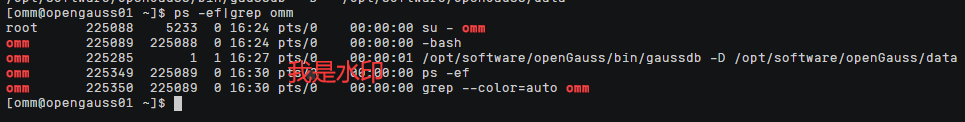
首先需要对数据库状态进行验证，**执行结果截图粘贴至实验报告**。

[omm@opengauss01 ~]$ **gs\_ctl status**



其次，查询数据库进程，需包含数据库服务器的主机名，**执行结果截图粘贴至实验报告**。

[omm@opengauss01 ~]$ **ps -ef|grep omm**



# 关卡二：openGauss数据导入及基本操作

## 实验介绍

### 关于本实验

本实验关卡，在安装完成的openGauss数据库上，进行TPC-H和target数据初始化操作，为后续实验做数据准备。

### 实验目的

* 掌握openGauss数据库的基本维护方法；

## 实验任务及步骤

### 数据初始化

使用root登录，进入/opt/software目录下。

[omm@opengauss01 ~]$ **exit**

[root@opengauss01 ~]# **cd /opt/software**

下载TPC-H测试包。

[root@opengauss01 software]# **git clone https://gitee.com/xzp-blog/tpch-kit.git**

将tpch-hit目录的属组修改为omm用户。

[root@opengauss01 software]# **chown omm:dbgrp -R /opt/software/tpch-kit/**

切换至用户omm，并进入测试包的目录。

[root@opengauss01 software]# **su - omm**

[omm@opengauss01 ~]$ **cd /opt/software/tpch-kit/dbgen/**

生成MakeFile文件。

[omm@opengauss01 dbgen]$ **make -f Makefile**

连接openGauss数据库。

[omm@opengauss01 dbgen]$ **gsql -d postgres -p 5432 -r**

创建测试数据库tpch并退出。

openGauss=# **CREATE DATABASE tpch;**

openGauss=# **\q**

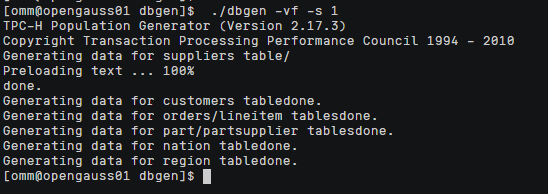
执行创建对象脚本。

[omm@opengauss01 dbgen]$ **gsql tpch -f dss.ddl**

产生测试数据。

[omm@opengauss01 dbgen]$ **./dbgen -vf -s 1**

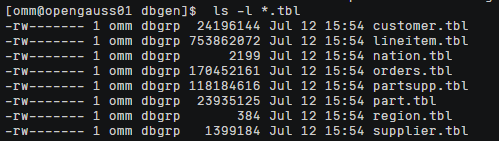
当返回如下内容，表示数据生产完成。



使用ls命令查看已经产生的数据：

[omm@opengauss01 dbgen]$ **ls -l \*.tbl**

回显生成的数据。



编辑加载脚本LoadData.sh。

[omm@opengauss01 dbgen]$ **vi LoadData.sh**

将以下内容添加进LoadData.sh脚本中，然后保存并退出编辑。

**for i in `ls \*.tbl`; do**

**table=${i/.tbl/}**

**echo "Loading $table..."**

**sed 's/|$//' $i > /tmp/$i**

**gsql tpch -q -c "TRUNCATE $table"**

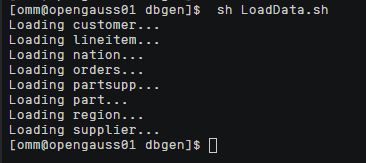
**gsql tpch -c "\\copy $table FROM '/tmp/$i' CSV DELIMITER '|'"**

**done**

执行加载脚本LoadData.sh，将数据加载进数据库中。

[omm@opengauss01 dbgen]$ **sh LoadData.sh**

返回结果（这里装载数据需要几分钟时间，请耐心等待）：

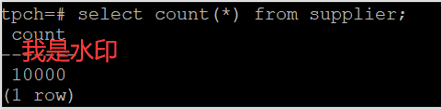


登录数据库验证，**将查询结果截图粘贴至实验报告**。

[omm@opengauss01 dbgen]$ **gsql -d tpch -p 5432 -r**

tpch=# **select count(\*) from supplier;**

此处应有返回记录，数据量为10000。



退出数据库。

tpch=# **\q**

将相关查询拷贝进查询目录中，并进入queries/目录中。

[omm@opengauss01 dbgen]$ **cp dists.dss queries/**

[omm@opengauss01 dbgen]$ **cp qgen queries/**

[omm@opengauss01 dbgen]$ **cd queries/**

生成查询语句。

[omm@opengauss01 queries]$ **vi genda.sh**

将以下内容复制到genda.sh中，保存并退出编辑。

**for i in {1..22}; do**

**./qgen -d $i>$i\_new.sql**

**./qgen -d $i\_new | sed 's/limit -1//' | sed 's/limit 100//' | sed 's/limit 10//' | sed 's/limit 20//' | sed 's/day (3)/day/' > queries.sql**

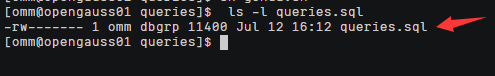
**done**

执行genda.sh脚本，生成查询语句

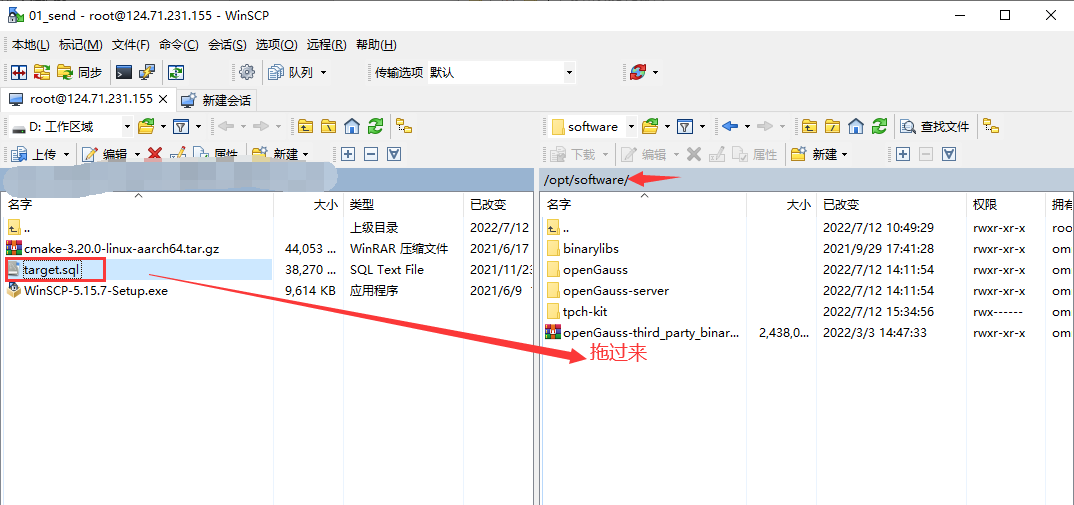
[omm@opengauss01 queries]$ **sh genda.sh**

查看生成的查询语句

[omm@opengauss01 queries]$ **ls -l queries.sql**



上传target.sql脚本至/opt/software目录下（可以使用winscp工具传输文件，target.sql已随堂下发），具体如下图：



切换至root用户，将脚本的属组修改为omm。

[omm@opengauss01 queries]$ **exit**

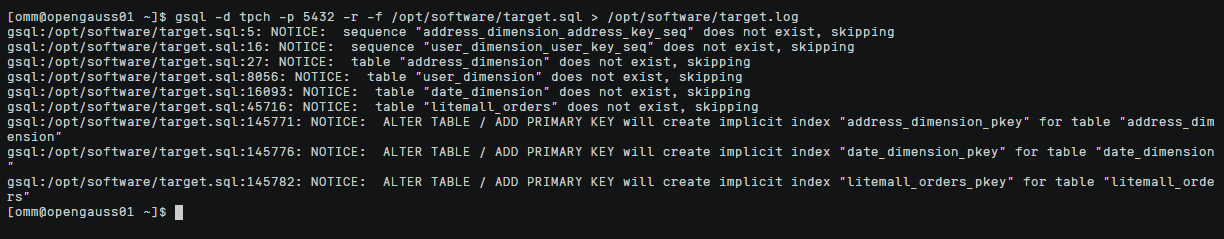
[root@opengauss01 software]# **chown omm:dbgrp /opt/software/target.sql**

切回omm用户，将target.sql中的内容导入到tpch数据库中

[root@opengauss01 software]# **su - omm**

[omm@opengauss01 ~]$ **gsql -d tpch -p 5432 -r -f /opt/software/target.sql > /opt/software/target.log**

数据量较大，数据导入过程需要较长时间，请耐心等待。当返回如下内容，表示数据导入完成。

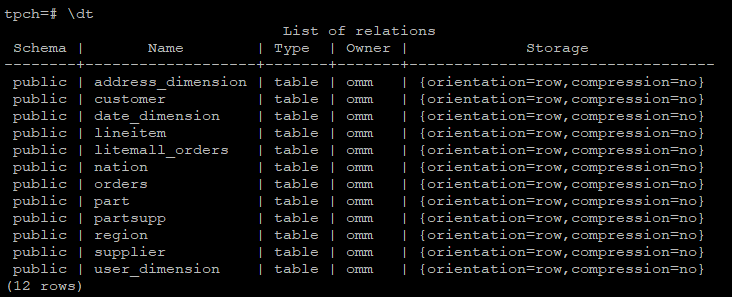


登录数据库进行验证，**将查询结果截图粘贴至实验报告**

[omm@opengauss01 ~]$ **gsql -d tpch -p 5432 -r**

tpch=# **\dt**

步骤11产生的8张表，加上步骤20产生的4张表，总共有12张表。



查询customer表的数据，**将查询结果截图粘贴至实验报告**。

tpch=# **select \* from customer limit 10;**

退出数据库。

tpch=# **\q**

### 关卡验证

本关卡进行数据初始化，需要将步骤12、21、22进行**截图粘贴至实验报告**。

### 思考题

数据初始化中用到的TPC-H，这是什么？

# 关卡三：openGauss的AI4DB特性应用

## 实验介绍

### 关于本实验

本实验关卡，在安装完成的openGauss数据库上，使用AI4DB的功能，对数据库参数进行优化；对TPCH测试中的SQL语句进行优化。

### 实验目的

* 掌握openGauss数据库的AI4DB的功能；
* 掌握openGauss数据库的参数修改方法；
* 掌握openGauss数据库的索引创建与使用方法。

## 实验任务及步骤

### 将X-Tuner 安装到系统中

切换成root用户，安装依赖包。

[omm@opengauss01 ~]$ **exit**

[root@opengauss01 ~]#**cd /root**

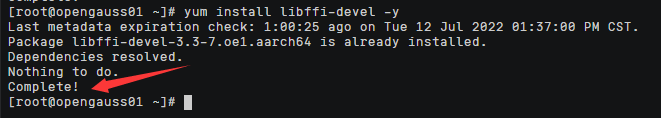
[root@opengauss01 ~]#**yum -y groupinstall "Development tools" --nogpgcheck**



[root@opengauss01 ~]#**yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel libpcap-devel xz-devel --nogpgcheck**



[root@opengauss01 ~]#**yum install libffi-devel -y**



下载Python3.8.3版本

[root@opengauss01 ~]#**wget https://repo.huaweicloud.com/python/3.8.3/Python-3.8.3.tgz**

[root@opengauss01 ~]#**tar -zxvf Python-3.8.3.tgz**

创建编译安装目录

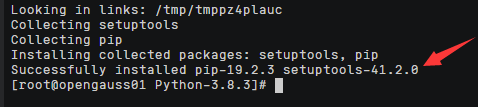
[root@opengauss01 ~]#**mkdir /usr/local/python3**

[root@opengauss01 ~]#**cd Python-3.8.3**

[root@opengauss01 Python-3.8.3]#**./configure --prefix=/usr/local/python3**

[root@opengauss01 Python-3.8.3]#**make && make install**

等待数分钟后，出现Successfully回显，表明安装完成。



创建软链接

修改Python以及pip的软链接 。（可一次性copy下面的5条命令执行）

**rm -rf /usr/bin/python**

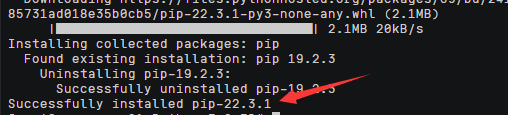
**ln -s /usr/local/python3/bin/python3 /usr/bin/python**

**rm -rf /usr/bin/pip**

**ln -s /usr/local/python3/bin/pip3 /usr/bin/pip**

**pip install --upgrade pip**

看到下图表示执行完毕：



修改镜像源地址

[root@opengauss01 Python-3.8.3]#**mkdir ~/.pip/**

[root@opengauss01 Python-3.8.3]#**vi ~/.pip/pip.conf**

添加如下配置

[global]

index-url = https://repo.huaweicloud.com/repository/pypi/simple

trusted-host = repo.huaweicloud.com

timeout = 120

使用root用户，进入到xtuner的安装目录。

[root@opengauss01 Python-3.8.3]# **cd /opt/software/openGauss/bin/dbmind/xtuner/**

[root@opengauss01 xtuner]# **pip install -r requirements-aarch64.txt**

[root@opengauss01 xtuner]#**cd tuner**

[root@opengauss01 tuner]#**export PYTHONPATH='..'**

设置配置文件setup.cfg。

[[root@opengauss01 tuner]#**cd ../**

[root@opengauss01 xtuner]# **vi setup.cfg**

将以下内容添加进**setup.cfg**配置中。

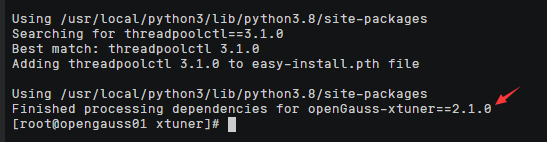
**[easy\_install]**

**index\_url = https://pypi.tuna.tsinghua.edu.cn/simple**

安装X-Tuner。

[root@opengauss01 xtuner]# **python setup.py install**

安装过程中会出现很多warning，忽略即可，等待数分钟后，出现Finished回显，表明安装完成。



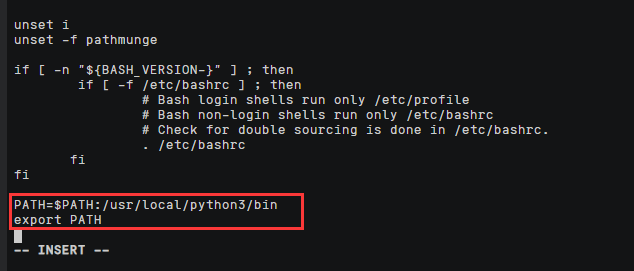
配置gs\_xtuner环境变量

[root@opengauss01 xtuner]# **vi /etc/profile**

添加如下配置到文件末尾。

PATH=$PATH:/usr/local/python3/bin

export PATH

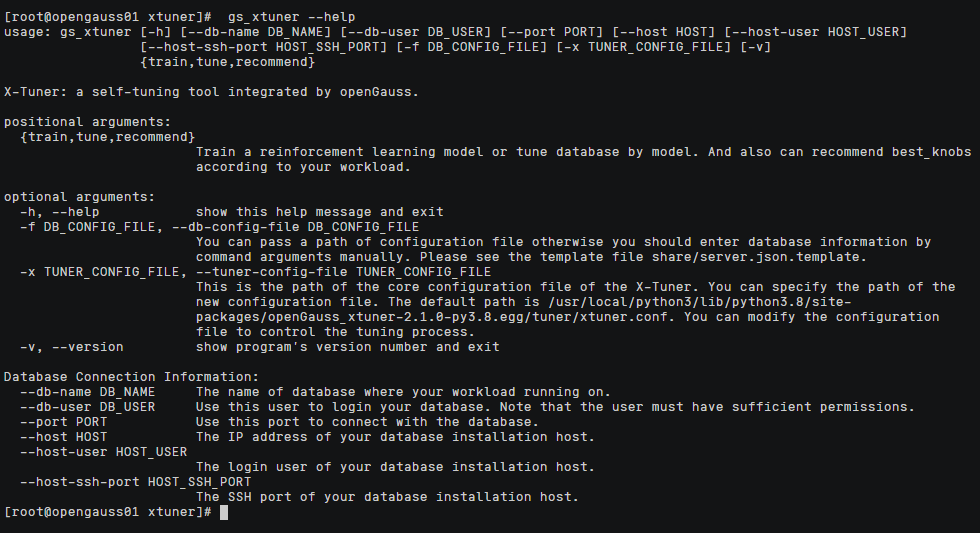


使环境变量生效。

[root@opengauss01 xtuner]# **source /etc/profile**

检测是否成功。

[root@opengauss01 xtuner]# **gs\_xtuner --help**



### 使用X-Tuner进行参数优化

切换到omm用户，将环境变量加入omm用户的.bashrc文件中。

[root@opengauss01 xtuner]# **su - omm**

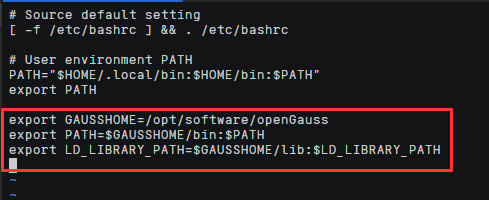
[omm@opengauss01 ~]$ **vi ~/.bashrc**

在文件末尾加入以下内容并保存退出。

**export GAUSSHOME=/opt/software/openGauss**

**export PATH=$GAUSSHOME/bin:$PATH**

**export LD\_LIBRARY\_PATH=$GAUSSHOME/lib:$LD\_LIBRARY\_PATH**



使环境变量生效。

[omm@opengauss01 ~]$ **source ~/.bashrc**

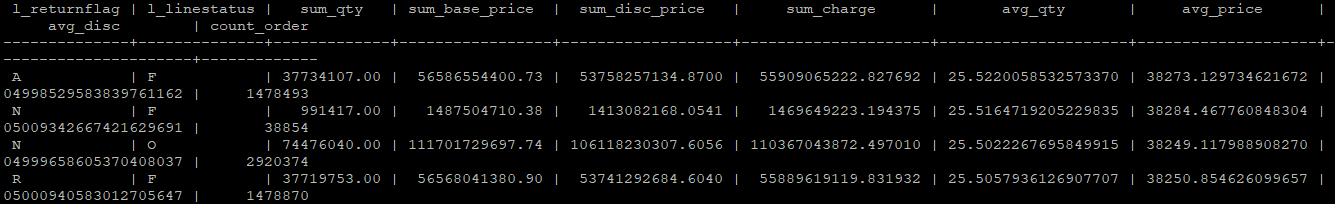
执行TPCH脚本，获得测试时间，**将执行结果截图粘贴至实验报告**。

[omm@opengauss01 ~]$ **gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log**

这里因为数据量大，且数据库是初始状态，未经优化，所以需要等待二十多分钟后才能执行完成，中间可以通过开启另一个CloudShell连接，然后使用tail工具来监控执行进展。

在另一个CloudShell连接窗口中执行如下内容进行监控。

[root@opengauss01 ~]$ **tail -f /opt/software/tpch-kit/dbgen/queries/queries01.log**

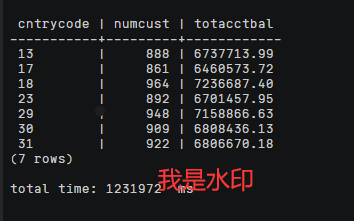


当监控中返回total time: 字样时，表示执行完成，接着关闭新开启的CloudShell连接。

在原来CloudShell连接窗口中查看queries01.log。

[omm@opengauss01 ~]$ **tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log**

得到结果如下，**将结果截图**：



切换至root用户，执行X-Tuner进行参数建议优化，**执行结果截图粘贴至实验报告**。

需要输入正确的omm数据库用户密码及omm操作系统用户密码（密码为关卡一中创建omm时设置的用户密码）。

[omm@opengauss01 ~]$ **exit**

[root@opengauss01 xtuner]# **gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm**

返回结果为：

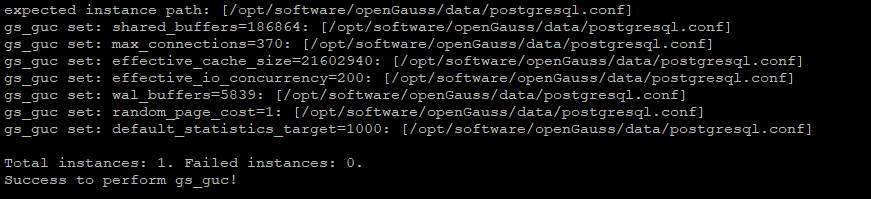


使用omm用户，对以上参数进行优化。

[root@opengauss01 xtuner]# **su - omm**

[omm@opengauss01 ~]$ **gs\_guc set -D /opt/software/openGauss/data/ -c "shared\_buffers = 187388" -c "max\_connections = 370" -c "max\_prepared\_transactions = 370" -c "effective\_cache\_size = 21602940" -c "effective\_io\_concurrency = 200" -c "wal\_buffers = 5855" -c "random\_page\_cost = 1" -c "default\_statistics\_target = 1000"**

返回结果为：



使用omm用户，重启数据库。（刚才修改的配置有几个需要重启才能生效）

[omm@opengauss01 ~]$ **gs\_ctl stop**

[omm@opengauss01 ~]$ **gs\_ctl start -D /opt/software/openGauss/data -Z single\_node -l logfile**

获取参数值，**将执行结果截图粘贴至实验报告**。

[omm@opengauss01 ~]$ **cd /opt/software/openGauss/data**

[omm@opengauss01 data]$ **cat postgresql.conf|grep -E 'shared\_buffers|max\_connections|effective\_cache\_size|effective\_io\_concurrency|wal\_buffers|random\_page\_cost|default\_statistics\_target'**

再次执行步骤2，**将执行结果截图粘贴至实验报告，**对比优化前的执行时间。

【附加题】回到步骤4，gs\_guc set命令只修改了其中几个推荐优化的参数，同学们可以把推荐优化的参数都设置一下，再进行测试，**将执行结果截图粘贴至实验报告**，跟步骤7的结果进行对比。（记得修改参数后，如果restart是True的参数，需要重启才能生效）

### Index-advisor：索引推荐

登录数据库。

[omm@opengauss01 data]$ **gsql -d tpch -p 5432 -r**

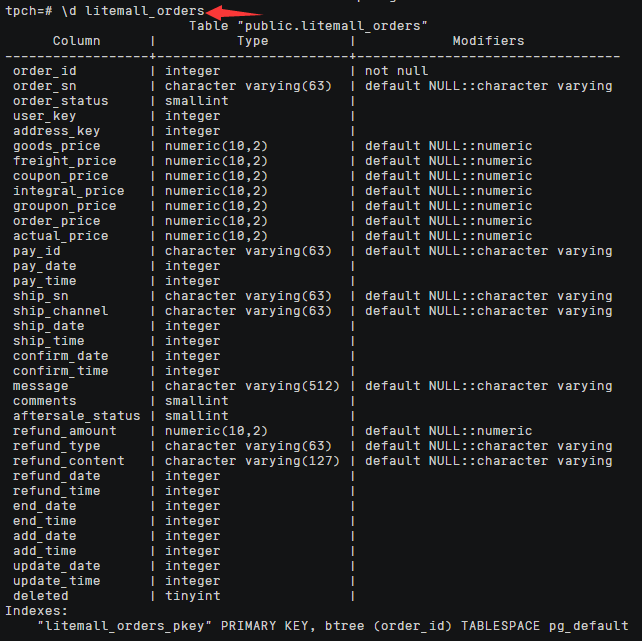
备用知识。

下面会以一个查询来演示索引推荐的用法。这个查询涉及的表有litemall\_orders订单表、address\_dimension地址表、date\_dimension日期表。



每个表的表结构可以用如下命令查看：

tpch=# **\d litemall\_orders**



使用SQL查询2020年3月每个省的订单收入，并按收入额降序排序。

tpch=# **SELECT ad.province AS province, SUM(o.actual\_price) AS GMV**

**FROM litemall\_orders o,**

**address\_dimension ad,**

**date\_dimension dd**

**WHERE o.address\_key = ad.address\_key**

**AND o.add\_date = dd.date\_key**

**AND dd.year = 2020**

**AND dd.month = 3**

**GROUP BY ad.province**

**ORDER BY SUM(o.actual\_price) DESC;**

使用explain，对该SQL加以分析，**将执行结果截图粘贴至实验报告**。

tpch=# **EXPLAIN**

**SELECT ad.province AS province, SUM(o.actual\_price) AS GMV**

**FROM litemall\_orders o,**

**address\_dimension ad,**

**date\_dimension dd**

**WHERE o.address\_key = ad.address\_key**

**AND o.add\_date = dd.date\_key**

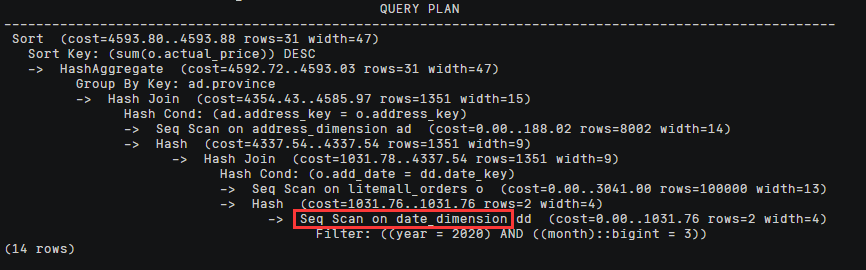
**AND dd.year = 2020**

**AND dd.month = 3**

**GROUP BY ad.province**

**ORDER BY SUM(o.actual\_price) DESC;**

获得执行计划结果为：



使用索引推荐功能，对查询语句进行推荐。

tpch=# **select \* from gs\_index\_advise('**

**SELECT ad.province AS province, SUM(o.actual\_price) AS GMV**

**FROM litemall\_orders o,**

**address\_dimension ad,**

**date\_dimension dd**

**WHERE o.address\_key = ad.address\_key**

**AND o.add\_date = dd.date\_key**

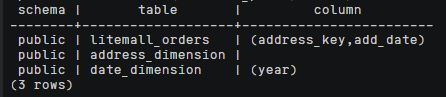
**AND dd.year = 2020**

**AND dd.month = 3**

**GROUP BY ad.province**

**ORDER BY SUM(o.actual\_price) DESC');**

获得索引推荐结果如下：



在litemall\_orders和date\_dimension表上创建虚拟索引列。

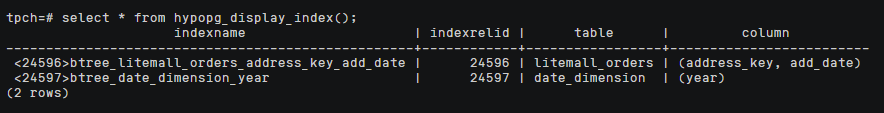
tpch=# **select \* from hypopg\_create\_index('create index on litemall\_orders(address\_key,add\_date)');**

tpch=# **select \* from hypopg\_create\_index('create index on date\_dimension(year)');**

查看创建的虚拟索引列。

tpch=# **select \* from hypopg\_display\_index();**

获得创建虚拟列的结果：



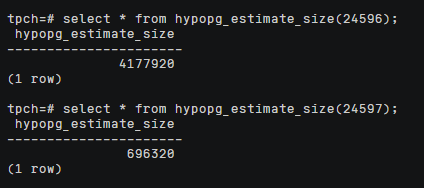
获取索引虚拟列大小结果（单位为：字节）。

其中24596和24597为上一步中查询到的indexrelid，此处查询需要进行替换成自己系统中查询到的值。

tpch=# **select \* from hypopg\_estimate\_size(24596);**

tpch=# **select \* from hypopg\_estimate\_size(24597);**

返回结果为：



开启GUC参数enable\_hypo\_index。

该参数控制数据库的优化器进行EXPLAIN时是否考虑创建的虚拟索引。通过对特定的查询语句执行explain，用户可根据优化器给出的执行计划评估该索引是否能够提升该查询语句的执行效率。

tpch=# **set enable\_hypo\_index = on;**

再次使用explain，对刚才执行的SQL加以分析，**将执行结果截图粘贴至实验报告**。

tpch=# **EXPLAIN**

**SELECT ad.province AS province, SUM(o.actual\_price) AS GMV**

**FROM litemall\_orders o,**

**address\_dimension ad,**

**date\_dimension dd**

**WHERE o.address\_key = ad.address\_key**

**AND o.add\_date = dd.date\_key**

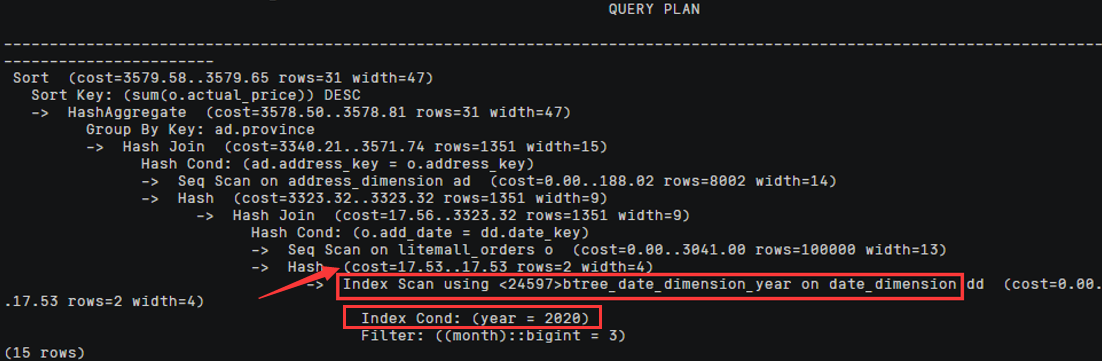
**AND dd.year = 2020**

**AND dd.month = 3**

**GROUP BY ad.province**

**ORDER BY SUM(o.actual\_price) DESC;**

获得执行计划：



对步骤5创建的索引虚拟列进行清理。

tpch=# **select \* from hypopg\_reset\_index();**

【附加题】对queries.sql里面的语句进行索引推荐，创建虚拟索引列，重新执行queries.sql查询，**将执行结果截图粘贴至实验报告**。

先退出数据库，在操作系统omm用户环境下调用queries.sql脚本。

tpch=# **\q**

[omm@opengauss01 data]$ **gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log**

等待二十多分钟后才能执行完成（当出现耗时表示执行完成），然后查询queries02.log日志文件：

[omm@opengauss01 data]$ **tail -10 /opt/software/tpch-kit/dbgen/queries/queries02.log**

### 关卡验证

1. 使用X-Tuner完成参数优化建议，并针对建议值完成openGauss数据库参数优化，对步骤2、3、6、7结果**截图粘贴至实验报告**；
2. 完成索引推荐功能的优化，使用索引推荐以及虚拟索引列的创建，对比创建的索引虚拟列前后的执行时间，对步骤3和9进行**截图粘贴至实验报告**。

# 关卡四【附加题】：openGauss的DB4AI特性应用

## 实验介绍

### 关于本实验

本实验关卡，在安装完成的openGauss数据库上，使用DB4AI的功能，利用openGauss的DB4AI性能进行房价的预测。

### 实验目的

* 掌握openGauss数据库的DB4AI的功能。

## 实验任务及步骤

### 利用DB4AI原生AI引擎训练并预测模型

在数据库中创建一张表，用于数据的训练与预测。

在OMM用户环境下，登录数据库。

[omm@opengauss01 data]$ **gsql -d postgres -p 5432 -r**

然后用以下语句来创建表。

openGauss=# **DROP TABLE IF EXISTS houses;**

openGauss=# **CREATE TABLE houses (id INT, tax INT, bedroom INT, bath FLOAT, price INT, size INT, lot INT);**

**INSERT INTO houses VALUES**

**(1 , 590 , 2 , 1 , 50000 , 770 , 22100),**

**(2 , 1050 , 3 , 2 , 85000 , 1410 , 12000),**

**(3 , 20 , 3 , 1 , 22500 , 1060 , 3500),**

**(4 , 870 , 2 , 2 , 90000 , 1300 , 17500),**

**(5 , 1320 , 3 , 2 , 133000 , 1500 , 30000),**

**(6 , 1350 , 2 , 1 , 90500 , 820 , 25700),**

**(7 , 2790 , 3 , 2.5 , 260000 , 2130 , 25000),**

**(8 , 680 , 2 , 1 , 142500 , 1170 , 22000),**

**(9 , 1840 , 3 , 2 , 160000 , 1500 , 19000),**

**(10 , 3680 , 4 , 2 , 240000 , 2790 , 20000),**

**(11 , 1660 , 3 , 1 , 87000 , 1030 , 17500),**

**(12 , 1620 , 3 , 2 , 118600 , 1250 , 20000),**

**(13 , 3100 , 3 , 2 , 140000 , 1760 , 38000),**

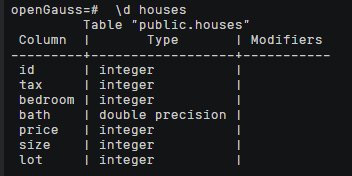
**(14 , 2070 , 2 , 3 , 148000 , 1550 , 14000),**

**(15 , 650 , 3 , 1.5 , 65000 , 1450 , 12000);**

通过 \d 命令观察新创建的表 house 结构。

openGauss=# **\d houses**

可以得到表结构信息：



通过CREATE MODEL语句，基于SVM算法创建一个二分类模型。

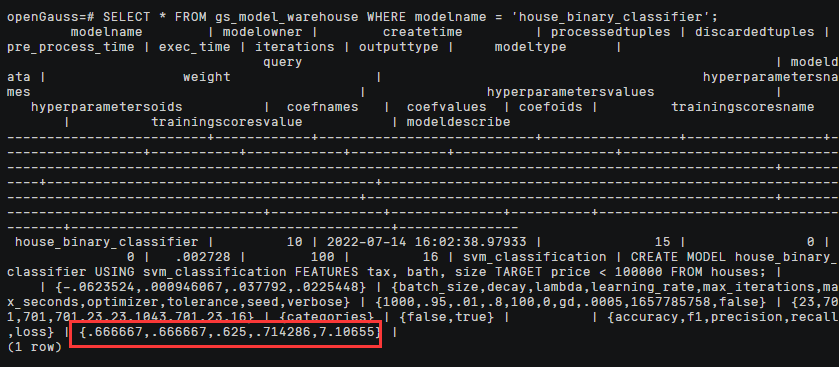
openGauss=# **CREATE MODEL house\_binary\_classifier USING svm\_classification FEATURES tax, bath, size TARGET price < 100000 FROM houses;**

执行CREATE MODEL 语句后，会开启模型的训练，并输出训练过程中使用的超参数信息：



在gs\_model\_warehouse系统表中查看训练后的模型信息。

openGauss=# **SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';**



通过系统表的返回信息可以看到训练过程的预测准确率为 0.666667。

这对于二分类来说准确率并不高，这也跟数据量只有15条有关。我们后面演示一下通过修改超参数来提升模型表现。

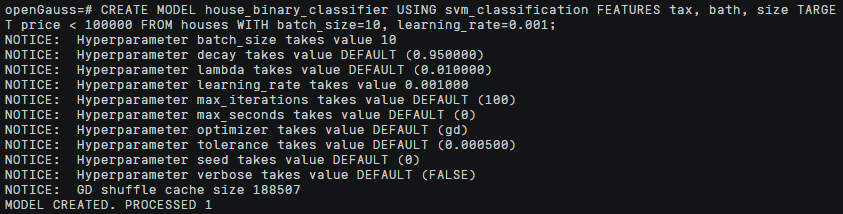
使用 DROP MODEL 语句删除模型。

openGauss=# **DROP MODEL house\_binary\_classifier;**

使用更优的超参数提高SVM算法的训练表现。

openGauss=# **CREATE MODEL house\_binary\_classifier USING svm\_classification FEATURES tax, bath, size TARGET price < 100000 FROM houses WITH batch\_size=10, learning\_rate=0.001;**

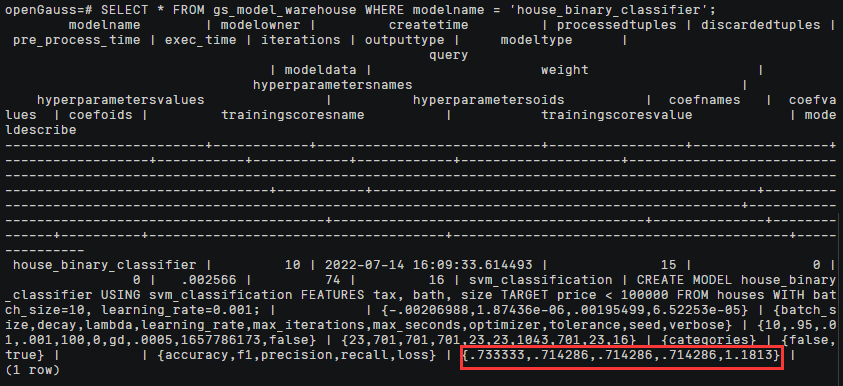
在该步骤中，修改训练时的学习率（learning\_rate）为0.001，批大小（batch\_size）设置为10。



观察新模型的信息。

openGauss=# **SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';**

在返回信息中，我们可以看到新模型的表现：

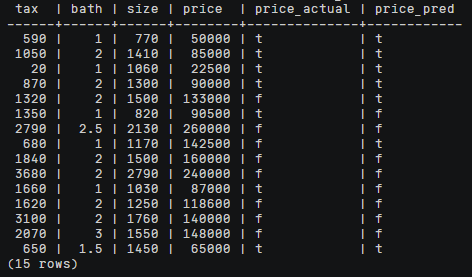


整体准确率要相对默认超参数有所提升。

使用PREDICT BY 语句预测样本数据。

openGauss=# **SELECT tax, bath, size, price, price < 100000 AS price\_actual, PREDICT BY house\_binary\_classifier (FEATURES tax, bath, size) AS price\_pred FROM houses;**

返回结果为：



price\_pred 列的输出结果即为预测结果，price\_actual 列的结果为真实值。

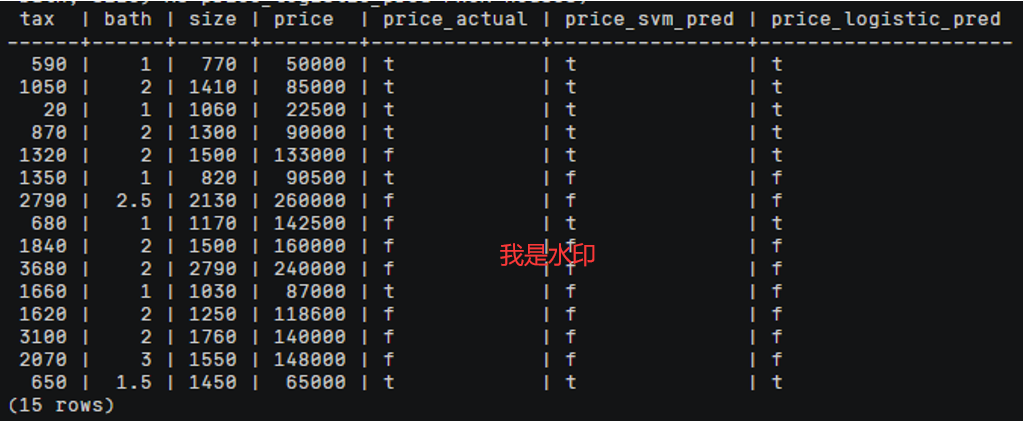
通过 CREATE MODEL 语句创建一个逻辑回归（logistic regression）模型。

openGauss=# **CREATE MODEL house\_logistic\_classifier USING logistic\_regression FEATURES tax, bath, size target price < 100000 FROM houses WITH learning\_rate=0.001;**

利用训练好的逻辑回归模型预测数据，并与SVM算法进行比较，**执行结果截图粘贴至实验报告**。

openGauss=# **SELECT tax, bath, size, price, price < 100000 AS price\_actual, PREDICT BY house\_binary\_classifier (FEATURES tax, bath, size) AS price\_svm\_pred, PREDICT BY house\_logistic\_classifier (FEATURES tax, bath, size) AS price\_logistic\_pred FROM houses;**

输出结果如下：



退出数据库。

openGauss=# **\q**

### 关卡验证

完成模型训练和预测过程，将svm算法和逻辑回归模型预测结果对比**截图粘贴至实验报告**。

# 清理工作：资源释放

## 实验介绍

### 关于本实验

本实验通过在华为云上删除弹性云服务器ECS实例，帮助学员和读者掌握在华为云上清除弹性云服务器ECS资源。

### 实验目的

* 掌握弹性云服务器资源释放操作；
* 熟悉华为云操作。

## 删除弹性云服务器及相关资源

完成实验后请务必删除华为云上的收费资源，以免造成不必要的收费。找到创建的弹性云服务器ECS，按照如下步骤进行删除。

打开云服务器控制台，在需要删除的云服务器后面选择“更多>删除”。



在弹出对话框中勾选“释放云服务器绑定的弹性公网IP地址”和“删除云服务器挂载的数据盘”，然后点击“是”。



查看到列表中已没有资源时，表示弹性云服务器已删除，**执行结果截图粘贴至实验报告**。



## 关卡验证

完成资源释放，**执行结果截图粘贴至实验报告**。

# 缩略语表

|  |  |  |
| --- | --- | --- |
| 缩略语 | 英文全称 | 中文全称 |
| PC | Personal Computer | 个人电脑 |
| EIP | Elastic IP | 弹性IP |
| VPC | Virtual Private Cloud | 虚拟私有云 |
| SG | Security Groups | 安全组 |
| ECS | Elastic Cloud Server | 弹性云服务器 |