# openGauss数据库安装

## 实验介绍

### 关于本实验

本实验主要描述openGauss数据库在openEuler弹性云服务器上单机安装部署。

### 实验目的

* 了解openGauss数据库部署方式；
* 掌握openGauss数据库安装部署方法。

## 购买弹性云服务器ECS（openEuler ARM操作系统）

### 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，单击登录。



输入账号名和密码，单击登录。



如果还没有注册，单击免费注册，按步骤注册后进行登录。

### 购买弹性云服务器ECS

在华为云主页（https://www.huaweicloud.com/）点击产品，选择＂精选推荐＂下的＂计算＂，再选择＂弹性云服务器ECS＂。



**③**

**②**

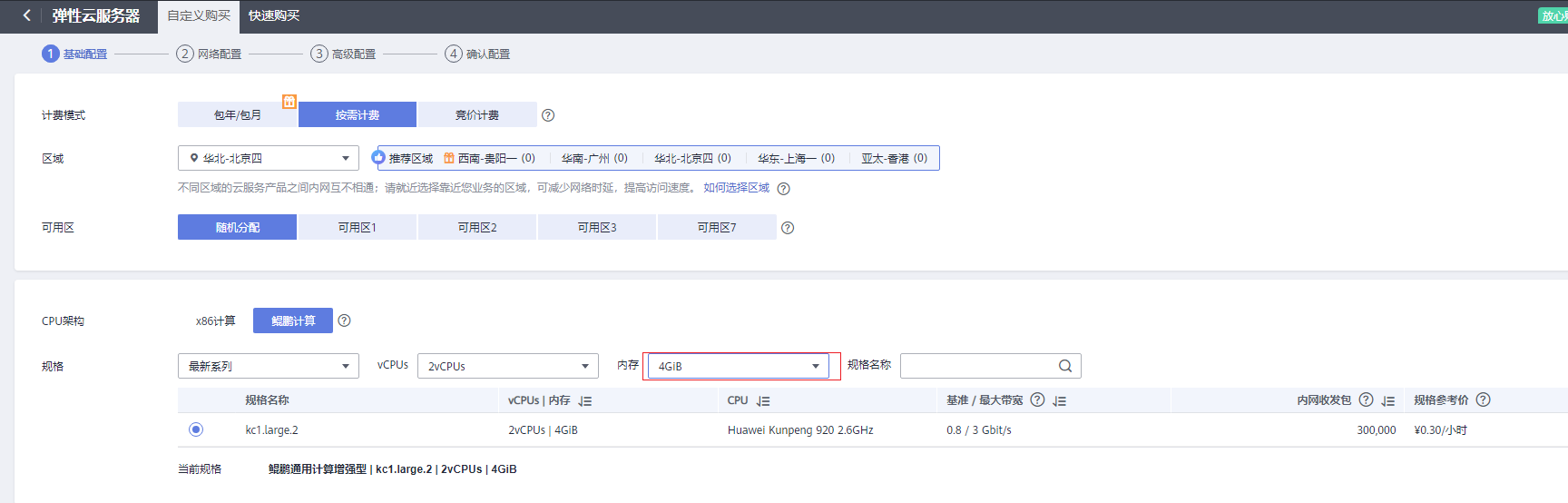
进入弹性云服务器ECS购买界面。



自定义购买进行基础配置。

ECS基础配置

|  |  |
| --- | --- |
| 配置选项 | 配置值 |
| 计费模式 | 按需计费（一定要选按需计费，注意配置费用） |
| 区域 | 华北-北京四（推荐，其他区域可能会导致无法选择openEuler公共镜像） |
| CPU架构 | 鲲鹏计算 |
| 规格 | 最新系列 2vCPUs|4GiB |
| 镜像 | 公共镜像：  openEuler  openEuler 20.03 64bit with ARM(40GB) |



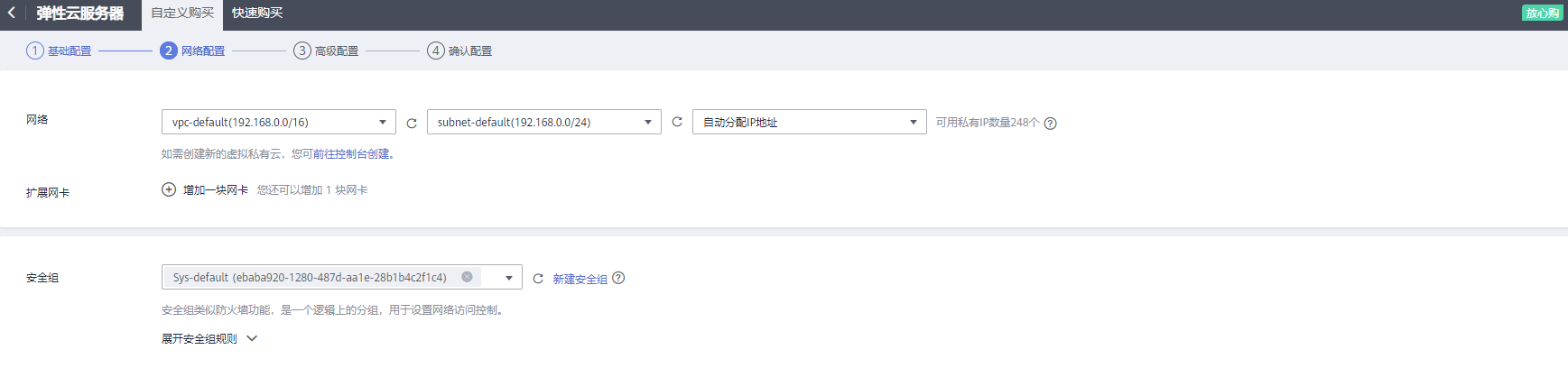


其余默认即可，单击＂下一步网络配置＂。

自定义购买进行网路配置。

ECS网络配置

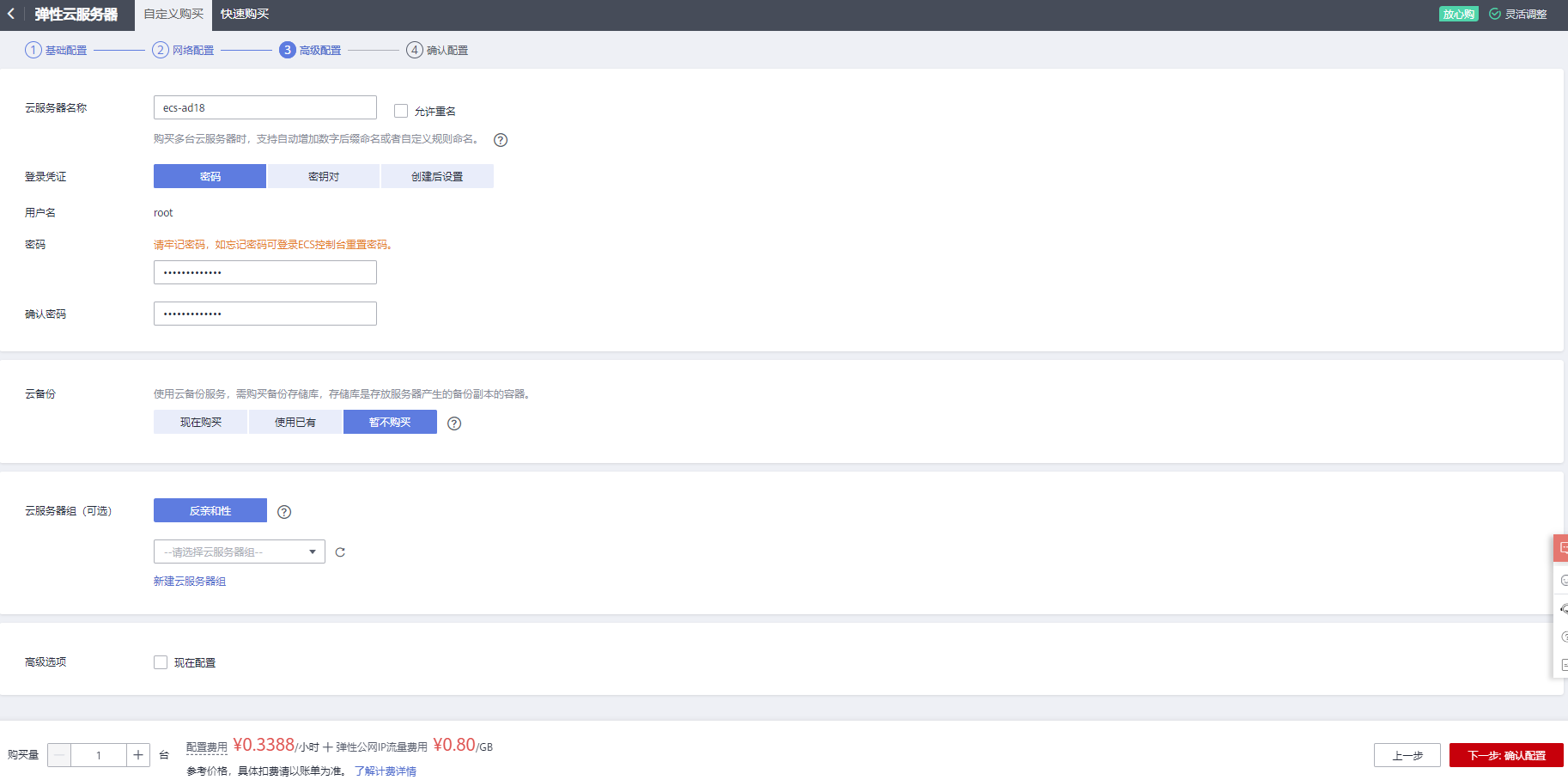
|  |  |
| --- | --- |
| 配置选项 | 配置值 |
| 网络 | vpc-default（192.168.0.0/16）（选现有默认网络即可） |
| 弹性公网IP | 现在购买 |
| 公网带宽 | 按流量计费 |
| 带宽大小 | 5 |





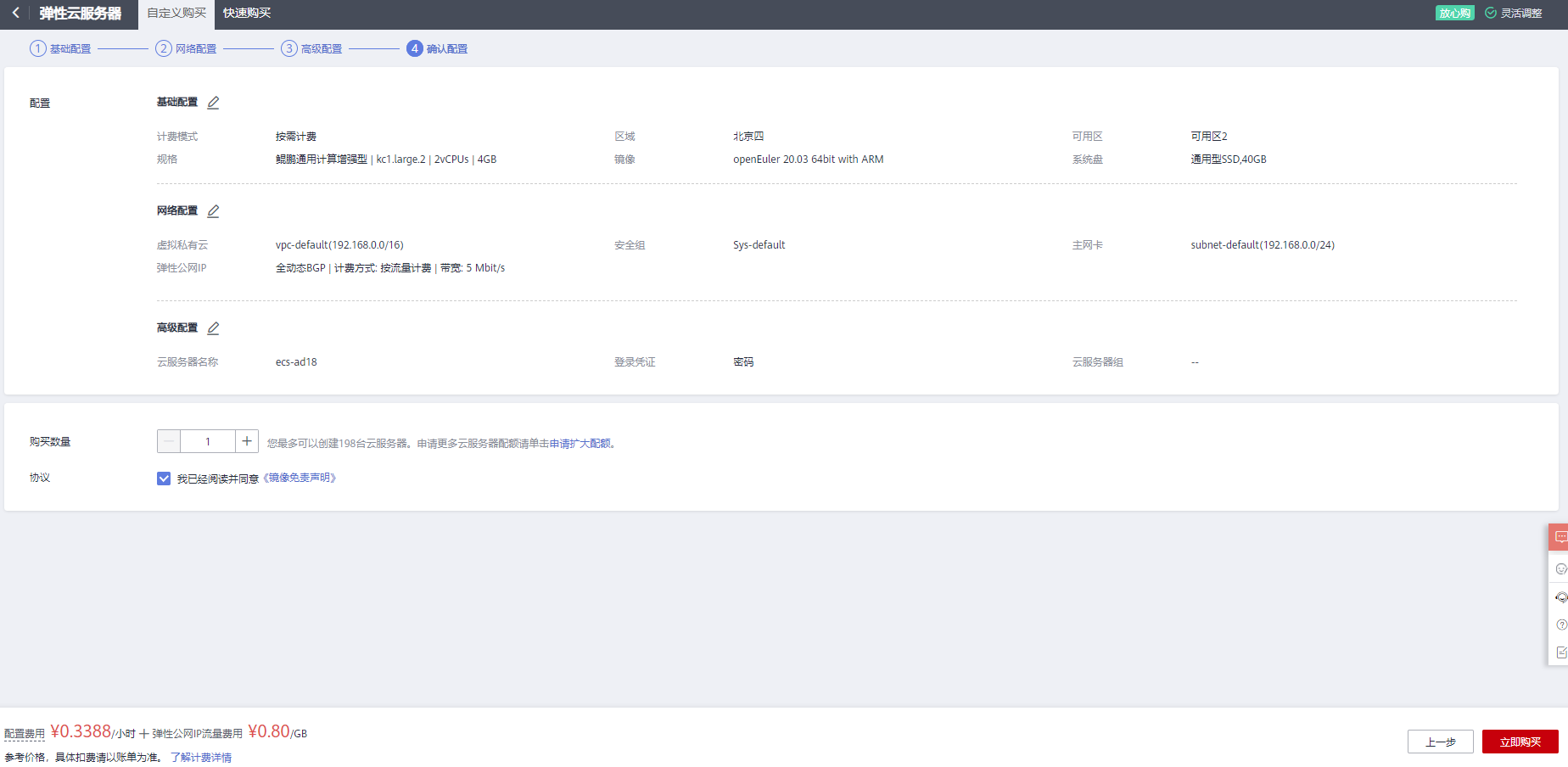
其余默认即可，单击＂下一步高级配置＂。

自定义购买进行高级配置。

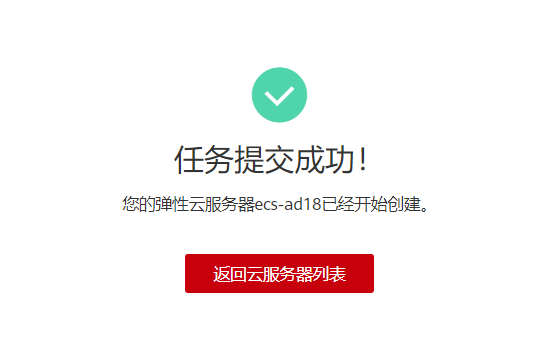


记住用户名为root，然后输入自定义密码和确认密码，其余默认即可，单击＂下一步确认设置＂。

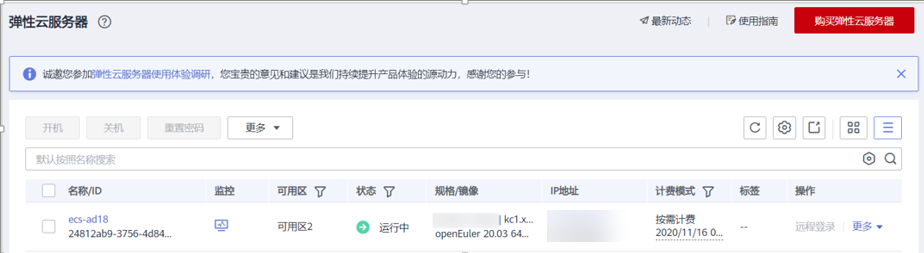
确认配置购买成功。



确认设置信息，尤其是配置费用，然后勾选协议＂我已经阅读并同意《镜像免责声明》＂，点击立即购买。



查看云服务器列表。



状态列显示＂运行中＂，则表示购买成功！

注意：本次购买鲲鹏服务器价格为公测价格，具体价格以华为云官网为准。

## 修改操作系统配置

### 连接服务器

方式一：

单击服务列表上的＂远程登录＂按钮进行登录，具体如下：

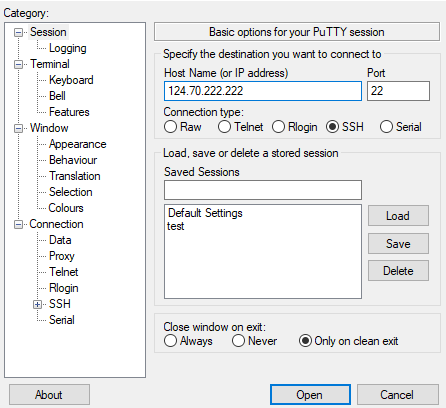


单击＂CloudShell登录＂，等加载完成后，输入root用户密码后点击＂连接＂按钮即可登录。



方式二：

使用SSH工具（比如：PuTTY等）从本地电脑通过配置弹性云服务器的弹性公网IP地址（如：124.70.222.222）来连接ECS，并使用root用户来登录。



### 设置字符集参数

将各数据库节点的字符集设置为相同的字符集，可以在/etc/profile文件中添加＂export LANG=XXX＂（XXX为Unicode编码）。

在/etc/profile文件中添加＂export LANG= en\_US.UTF‐8＂。

[root@ecs-c9bf ~]# **cat >>/etc/profile<<EOF**

**export LANG=en\_US.UTF‐8**

**EOF**

输入如下命令，使配置修改生效。

[root@ecs-c9bf ~]# **source /etc/profile**

### 修改python版本并安装libaio包

之后安装过程中openGauss用户互信，openEuler服务器需要用到Python-3.7.x命令，但是默认Python版本为Python-2.7.x，所以需要切换Python版本。

进入/usr/bin目录。

[root@ecs-c9bf ~]# **cd /usr/bin**

备份python文件。

[root@ecs-c9bf bin] # **mv python python.bak**

建立Python3软连接。

[root@ecs-c9bf bin] # **ln -s python3 /usr/bin/python**

验证Python版本。

[root@ecs-c9bf bin] # **python -V**

显示如下，即为Python版本切换成功：

Python 3.7.4

Python版本切换成功，后续安装需要libaio包，下载进行安装。

[root@ecs-c9bf ~]# **yum install libaio\* -y**

## 安装openGauss数据库

### 下载数据库安装包

以root用户登录待安装openGauss的主机，并按规划创建存放安装包的目录。

[root@ecs-c9bf bin]# **mkdir -p /opt/software/openGauss**[root@ecs-c9bf bin]# **chmod 755 -R /opt/software**

注：

* 不建议把安装包的存放目录规划到openGauss用户的home目录或其子目录下，可能导致权限问题。
* openGauss安装用户omm须具有/opt/software/openGauss目录的读写权限。

下载数据库安装包到安装目录。

切换到安装目录：

[root@ecs-c9bf bin]# **cd /opt/software/openGauss**

使用wget下载安装包：

[root@ecs-c9bf openGauss]# **wget https://opengauss.obs.cn-south-1.myhuaweicloud.com/2.0.0/arm/openGauss-2.0.0-openEuler-64bit-all.tar.gz**

注：https://opengauss.obs.cn-south-1.myhuaweicloud.com/2.0.0/arm/openGauss-2.0.0-openEuler-64bit-all.tar.gz是数据库安装包下载网址，输入时不需要进行换行。

下载成功显示如下：

……

2021-06-14 13:57:23 (9.33 MB/s) - ‘openGauss-2.0.0-openEuler-64bit-all.tar.gz’ saved [58468915/58468915]

### 创建XML配置文件

* 安装openGauss前需要创建XML文件。XML文件包含部署openGauss的服务器信息、安装路径、IP地址以及端口号等。用于告知openGauss如何部署。用户需根据不同场合配置对应的XML文件。
* 以单节点配置的方案为例，说明如何创建XML配置文件。

以root用户登录待安装openGauss的主机，切换到存放安装包的目录。

[root@ecs-c9bf bin]# **cd /opt/software/openGauss**

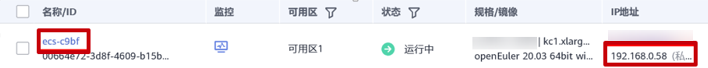
创建XML配置文件，用于数据库安装。

[root@ecs-c9bf openGauss]# **vi clusterconfig.xml**

输入＂i＂进入INSERT模式，添加文本如下。

红色加粗字体内容为示例，可自行替换，其中＂ecs-c9bf＂是弹性云服务器的名称，＂192.168.0.58＂为弹性云服务器的IP地址（私有），其他value值可以不进行修改。

弹性云服务器名称及私有IP查看：



<?xml version="1.0" encoding="UTF-8"?>

<ROOT>

<CLUSTER>

<PARAM name="clusterName" value="dbCluster" />

<PARAM name="nodeNames" value="**ecs-c9bf**" />

<PARAM name="backIp1s" value="**192.168.0.58**"/>

<PARAM name="gaussdbAppPath" value="/opt/gaussdb/app" />

<PARAM name="gaussdbLogPath" value="/var/log/gaussdb" />

<PARAM name="gaussdbToolPath" value="/opt/huawei/wisequery" />

<PARAM name="corePath" value="/opt/opengauss/corefile"/>

<PARAM name="clusterType" value="single-inst"/>

</CLUSTER>

<DEVICELIST>

<DEVICE sn="1000001">

<PARAM name="name" value="**ecs-c9bf**"/>

<PARAM name="azName" value="AZ1"/>

<PARAM name="azPriority" value="1"/>

<PARAM name="backIp1" value="**192.168.0.58**"/>

<PARAM name="sshIp1" value="**192.168.0.58**"/>

<!--dbnode-->

<PARAM name="dataNum" value="1"/>

<PARAM name="dataPortBase" value="26000"/>

<PARAM name="dataNode1" value="/gaussdb/data/db1"/>

</DEVICE>

</DEVICELIST>

</ROOT>

点击＂Esc＂退出INSERT模式，然后输入＂:wq＂后回车退出编辑并保存文本。

配置文件参数附加说明

|  |  |
| --- | --- |
| 参数 | 说明 |
| clusterName | openGauss名称。 |
| nodeNames | openGauss中主机名称。 |
| backIp1s | 主机在后端存储网络中的IP地址（内网IP）。所有openGauss主机使用后端存储网络通讯。 |
| gaussdbAppPath | openGauss程序安装目录。此目录应满足如下要求：  •磁盘空间>1GB。  •与数据库所需其它路径相互独立，没有包含关系。 |
| gaussdbLogPath | openGauss运行日志和操作日志存储目录。此目录应满足如下要求：  •磁盘空间建议根据主机上的数据库节点数规划。数据库节点预留1GB空间的基础上，再适当预留冗余空间。  •与openGauss所需其它路径相互独立，没有包含关系。  此路径可选。不指定的情况下，openGauss安装时会默认指定＂$GAUSSLOG/安装用户名＂作为日志目录。 |
| tmpdbPath | 数据库临时文件存放目录。  若不配置tmpdbPath，默认存放在/opt/huawei/wisequery/perfadm\_db目录下。 |
| gaussdbToolPath | openGauss系统工具目录，主要用于存放互信工具等。此目录应满足如下要求：  •磁盘空间>100MB。  •固定目录，与数据库所需其它目录相互独立，没有包含关系。  此目录为可选配置项。不指定的情况下，openGauss安装时会默认指定＂/opt/huawei/wisequery＂作为数据库系统工具目录。 |
| corePath | openGauss core文件的指定目录。 |

须知：

* ＂/opt/huawei/newsql/tools＂存放互信等工具，避免权限问题，不要把实例数据目录放在此目录下。
* 安装目录和数据目录须为空或者不存在，否则可能导致安装失败。
* 在对数据库节点的实例进行具体配置时，需确保配置的目录之间不相互耦合。即各个配置目录不关联，删除其中任意一个目录，不会级联删除其它目录。如gaussdbAppPath为＂/opt/gaussdb/app＂，gaussdbLogPath为＂/opt/gaussdb/app/omm＂。当gaussdbAppPath目录被删除时，会级联删除gaussdbLogPath目录，从而引起其它问题。
* 若需要安装脚本自动创建安装用户时，各配置的目录需保证不与系统创建的默认用户目录耦合关联。
* 配置openGauss路径和实例路径时，路径中不能包含＂|＂,＂;＂,＂&＂,＂$＂,＂<＂,＂>＂,＂`＂,＂\\＂,＂'＂,＂\＂,＂{＂,＂}＂,＂(＂,＂)＂,＂[＂,＂]＂,＂~＂,＂\*＂,＂?＂特殊字符。

### 初始化安装环境

为了保证openGauss的正确安装，请首先对主机环境进行配置。

#### 准备安装用户及环境

创建完openGauss配置文件后，在执行安装前，为了后续能以最小权限进行安装及openGauss管理操作，保证系统安全性，需要运行安装前置脚本gs\_preinstall，准备好安装用户及环境。

#### 前提条件

已完成安装准备的所有任务。

#### 注意事项

* 用户需要检查上层目录权限，保证安装用户对安装包和配置文件目录读写执行的权限；
* xml文件中主机的名称与IP映射配置正确；
* 只能使用root用户执行gs\_preinstall命令。

#### 操作步骤

修改performance.sh文件。

使用vi打开文件＂/etc/profile.d/performance.sh＂，具体如下：

[root@ecs-c9bf openGauss]# **vi /etc/profile.d/performance.sh**

输入＂i＂，进入INSERT模式。用#注释sysctl -w vm.min\_free\_kbytes=112640 &> /dev/null这行内容。

CPUNO=`cat /proc/cpuinfo|grep processor|wc -l`

export GOMP\_CPU\_AFFINITY=0-$[CPUNO - 1]

**#sysctl -w vm.min\_free\_kbytes=112640 &> /dev/null**

sysctl -w vm.dirty\_ratio=60 &> /dev/null

sysctl -w kernel.sched\_autogroup\_enabled=0 &> /dev/null

按＂Esc＂键，退出INSERT模式，接着输入＂:wq＂后回车，保存退出。

为确保openssl版本正确，执行预安装前加载安装包中lib库。

执行命令如下，其中packagePath为用户安装包放置的路径，本示例中为/opt/software/openGauss。

[root@ecs-c9bf openGauss]# **vi /etc/profile**

输入i，进入INSERT模式，在文件的底部添加如下代码，加载安装包中lib库。按下＂Esc＂退出INSERT模式，输入＂:wq＂后回车，保存后退出。

**export packagePath=/opt/software/openGauss**

**export LD\_LIBRARY\_PATH=$packagePath/script/gspylib/clib:$LD\_LIBRARY\_PATH**

配置完成后，使设置生效。

[root@ecs-c9bf openGauss]# **source /etc/profile**

在安装包所在的目录下，解压安装包。

[root@ecs-c9bf openGauss]# **cd /opt/software/openGauss**

解压openGauss-2.0.0-openEuler-64bit-all.tar.gz包。

[root@ecs-c9bf openGauss]# **tar -zxvf openGauss-2.0.0-openEuler-64bit-all.tar.gz**

解压openGauss-2.0.0-openEuler-64bit-om.tar.gz包。

[root@ecs-c9bf openGauss]# **tar -zxvf openGauss-2.0.0-openEuler-64bit-om.tar.gz**

解压后，执行ls命令查看内容如下：

[root@ecs-c9bf openGauss]# **ls**

clusterconfig.xml openGauss-Package-bak\_392c0438.tar.gz

lib script

openGauss-2.0.0-openEuler-64bit-all.tar.gz simpleInstall

openGauss-2.0.0-openEuler-64bit-om.sha256 upgrade\_sql.sha256

openGauss-2.0.0-openEuler-64bit-om.tar.gz upgrade\_sql.tar.gz

openGauss-2.0.0-openEuler-64bit.sha256 version.cfg

openGauss-2.0.0-openEuler-64bit.tar.bz2

安装包解压后，会在/opt/software/openGauss路径下自动生成script子目录，并且在script目录下生成gs\_preinstall等各种OM工具脚本。

使用gs\_preinstall准备好安装环境，切换到gs\_preinstall命令所在目录。

[root@ecs-c9bf openGauss]# **cd /opt/software/openGauss/script/**

执行ls命令查看script中内容。

[root@ecs-c9bf script]# **ls**

gs\_backup gs\_checkperf gs\_om gspylib gs\_uninstall \_\_init\_\_.py

gs\_check gs\_collector gs\_postuninstall gs\_ssh gs\_upgradectl killall

gs\_checkos gs\_install  **gs\_preinstall**  gs\_sshexkey impl local

采用交互模式执行，并在执行过程中会创建openGauss omm用户互信：

[root@ecs-c9bf script]# **python gs\_preinstall -U omm -G dbgrp -X /opt/software/openGauss/clusterconfig.xml**

命令中的omm为操作系统用户（同时omm也是openGauss的数据库管理员账号，在下面的1.4.4环节中会创建），dbgrp为运行openGauss的操作系统用户的群组名称，/opt/software/openGauss/clusterconfig.xml为openGauss配置文件路径。

在执行过程中，用户根据提示选择是否创建互信，填写 yes。

此时会创建操作系统omm用户，并对omm创建trust互信，并要求设置密码，设置为Admin@123（建议用户自定义设置密码）。

Are you sure you want to create the user[**omm**] and create trust for it (yes/no)? **yes**

Please enter password for cluster user.

Password: **--说明：此处输入密码时，屏幕上不会有任何反馈，不用担心，这是LINUX操作系统对密码的保护.**

Please enter password for cluster user again.

Password: **--说明：此处输入密码时，屏幕上不会有任何反馈，不用担心，这是LINUX操作系统对密码的保护.**

Successfully created [omm] user on all nodes.

成功后显示为：

……

Setting finish flag.

Successfully set finish flag.

Preinstallation succeeded.

### 执行安装

执行前置脚本准备好openGauss安装环境之后，按照启动安装过程部署openGauss。

#### 前提条件

* 已成功执行前置脚本gs\_preinstall；
* 服务器操作系统和网络均正常运行。

#### 操作步骤

修改文件权限。

[root@ecs-c9bf script]# **chmod -R 755 /opt/software/openGauss/script**

登录到openGauss的主机，并切换到omm用户。

[root@ecs-c9bf script]# **su - omm**

注：

* omm指的是前置脚本gs\_preinstall中-U参数指定的用户。
* 安装脚本gs\_install必须以前置脚本中指定的omm执行，否则，脚本执行会报错。

使用gs\_install安装openGauss。

执行以下命令进行安装

gs\_install -X /opt/software/openGauss/clusterconfig.xml --gsinit-parameter="--encoding=UTF8" --dn-guc="max\_process\_memory=4GB" --dn-guc="shared\_buffers=256MB" --dn-guc="bulk\_write\_ring\_size=256MB" --dn-guc="cstore\_buffers=16MB"

参数说明如下：

Encoding：设置字符集；

max\_process\_memory：设置一个数据库节点可用的最大物理内存；

shared\_buffers：设置openGauss使用的共享内存大小；

bulk\_write\_ring\_size：大批量数据写入触发时，该操作使用的环形缓冲区大小；

cstore\_buffers：设置列存所使用的共享缓冲区的大小。

具体执行如下：

[omm@ecs-c9bf ~]$ **gs\_install -X /opt/software/openGauss/clusterconfig.xml --gsinit-parameter="--encoding=UTF8" --dn-guc="max\_process\_memory=4GB" --dn-guc="shared\_buffers=256MB" --dn-guc="bulk\_write\_ring\_size=256MB" --dn-guc="cstore\_buffers=16MB"**

/opt/software/ openGauss/clusterconfig.xml为openGauss配置文件的路径。在执行过程中，用户需根据提示输入数据库管理员omm用户的密码，密码具有一定的复杂度，为保证用户正常使用该数据库，请记住输入的数据库密码。

按照设置密码要求，设置密码为GaussDB@123（**建议用户自定义设置复杂密码**）：

encrypt cipher and rand files for database.

Please enter password for database:  **--说明：此处输入密码时，屏幕上不会有任何反馈**

Please repeat for database: **--说明：此处输入密码时，屏幕上不会有任何反馈**

begin to create CA cert files

设置的密码要符合复杂度要求：

* 最少包含8个字符；
* 不能和用户名和当前密码（ALTER）相同，或和当前密码反序；
* 至少包含大写字母（A-Z），小写字母（a-z），数字，非字母数字字符（限定为~!@#$%^&\*()-\_=+\|[{}];:,<.>/?）四类字符中的三类字符。

显示如下内容即为安装成功。

……

Successfully deleted instances from all nodes.

Checking node configuration on all nodes.

Initializing instances on all nodes.

Updating instance configuration on all nodes.

Check consistence of memCheck and coresCheck on database nodes.

Configuring pg\_hba on all nodes.

Configuration is completed.

Successfully started cluster.

Successfully installed application.

end deploy..

### 安装生成的目录

安装后的目录及各目录下的文件说明请参见**下表**。

安装后的目录及各目录下的文件说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 项目目录说明 | 目录 | 子目录 | 说明 |
| 1 | 集群openGauss安装目录 | /opt/gaussdb/app | etc | cgroup工具配置文件。 |
| include | 存放数据库运行所需要的头文件。 |
| lib | 存放数据库的库文件的目录。 |
| share | 存放数据库运行所需要的公共文件，如配置文件模板。 |
| 2 | 集群openGauss数据目录 | /gaussdb/data | data\_dnxxx | DBnode实例的数据目录，其中主实例的目录名为＂data\_dnxxx＂，  备实例的为data\_dnSxxx。xxx代表DBnode编号。 |
| 3 | 集群openGauss日志目录 | /var/log/gaussdb/用户名 | bin | 二进制程序的日志目录。 |
| gs\_profile | 数据库内核性能日志目录。 |
| om | OM的日志目录。例如：  部分local脚本产生的日志，增删数据库节点接口的日志，gs\_om接口的日志，前置接口的日志，节点替换接口的日志等。 |
| pg\_audit | 数据库审计日志目录。 |
| pg\_log | 数据库节点实例的运行日志目录。 |
| 4 | 集群openGauss系统工具目录 | /opt/huawei/wisequery | script | 用于openGauss用户进行openGauss管理的脚本文件。 |
| lib | bin目录下的二进制文件依赖的库文件。 |

# 数据库性能检查

## 实验介绍

### 关于本实验

openGauss 不仅提供了gs\_checkperf工具来帮助用户了解openGauss的负载情况。

本实验主要是通过gs\_checkperf工具来检查openGauss数据库性能以及通过EXPLAIN来进行SQL语句优化。

### 实验目的

掌握gs\_checkperf工具的基本使用。

## 通过gs\_checkperf工具来检查数据库性能

说明：

gs\_checkperf可以对以下级别进行检查：

* openGauss级别（主机CPU占用率、Gauss CPU占用率、I/O使用情况等）
* 节点级别（CPU使用情况、内存使用情况、I/O使用情况）
* 会话/进程级别（CPU使用情况、内存使用情况、I/O使用情况）
* SSD性能（写入、读取性能）

其中检查SSD性能要用root用户执行，检查openGauss性能要用openGauss安装用户执行

本实验为检查openGauss性能。

用root用户登录装有openGauss数据库服务的操作系统然后用 su – omm命令切换至OMM用户环境，登录后信息如下。

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Tue Jul 21 09:21:11 CST 2021

System load: 0.01

Processes: 109

Memory used: 6.7%

Swap used: 0.0%

Usage On: 15%

IP address: 192.168.0.96

Users online: 1

[root@ecs-e1b3 ~]# **su - omm**

Last login: Fri Jul 10 19:05:39 CST 2021 on pts/0

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Tue Jul 21 09:21:25 CST 2021

System load: 0.01

Processes: 111

Memory used: 7.0%

Swap used: 0.0%

Usage On: 15%

IP address: 192.168.0.96

Users online: 1

[omm@ecs-e1b3 ~]$

检测准备。

先启动数据库服务。

[omm@ecs-e1b3 ~]$ **gs\_om -t start;**

Starting cluster.

=========================================

=========================================

Successfully started.

用gs\_checkperf检查。

[omm@ecs-e1b3 ~]$ **gs\_checkperf**

Cluster statistics information:

Host CPU busy time ratio : .72 %

MPPDB CPU time % in busy time : .33 %

Shared Buffer Hit ratio : 97.33 %

In-memory sort ratio : 0

Physical Reads : 466

Physical Writes : 175

DB size : 47 MB

Total Physical writes : 175

Active SQL count : 3

Session count : 4

确认openGauss数据库服务是否正常。

[omm@ecs-e1b3 ~]$ **gs\_om -t status;**

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

cluster\_state : Unavailable

redistributing : No

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

cluster\_state : Normal 表示已启动，可以正常使用。如果状态为Unavailable表示不可用

为了实验继续进行，请先启动数据库服务。

启动数据库服务（如果数据库服务是正常的，此步骤可以不执行）。

[omm@ecs-e1b3 ~]$ **gs\_om -t start;**

Starting cluster.

=========================================

=========================================

Successfully started.

然后连接postgres数据库。

[omm@ecs-e1b3 ~]$ **gsql -d postgres -p 26000 -r**

gsql ((openGauss 2.0.0 build 38a9312a) compiled at 2021-05-27 14:57:08 commit 472 last mr 549 )

Non-SSL connection (SSL connection is recommended when requiring high-security)

Type ＂help＂ for help.

postgres=#

对PMK模式下的表进行统计信息收集。

postgres=# **analyze pmk.pmk\_configuration;**

ANALYZE

postgres=# **analyze pmk.pmk\_meta\_data;**

ANALYZE

postgres=# **analyze pmk.pmk\_snapshot;**

ANALYZE

postgres=# **analyze pmk.pmk\_snapshot\_datanode\_stat;**

ANALYZE

postgres=#

说明：

gs\_checkperf工具的监控信息依赖于pmk模式下的表的数据，如果pmk模式下的表未执行analyze操作，则可能导致gs\_checkperf工具执行失败。

执行简要性能检查。

用 \q 先退出postgres数据库，然后在操作系统用户 omm 环境下去执行gs\_checkperf检查工具，具体如下。

postgres=#

postgres=# **\q**

[omm@ecs-e1b3 ~]$ **gs\_checkperf**

Cluster statistics information:

Host CPU busy time ratio : 1.66 % -----主机CPU占用率

MPPDB CPU time % in busy time : 2.51 % ----Gauss CPU占用率

Shared Buffer Hit ratio : 99.14 % ----共享内存命中率

In-memory sort ratio : 0 ---内存中排序比率

Physical Reads : 504 ---物理读次数

Physical Writes : 162 ---物理写次数

DB size : 57 MB ---DB大小

Total Physical writes : 162 ---总物理写次数

Active SQL count : 4 ---当前SQL执行数

Session count : 5 ---Session数量

执行详细性能检查。

[omm@ecs-e1b3 ~]$ **gs\_checkperf --detail**

Cluster statistics information:

Host CPU usage rate:

Host total CPU time : 45719980.000 Jiffies

Host CPU busy time : 761060.000 Jiffies

Host CPU iowait time : 6640.000 Jiffies

Host CPU busy time ratio : 1.66 %

Host CPU iowait time ratio : .01 %

MPPDB CPU usage rate:

MPPDB CPU time % in busy time : 5.12 %

MPPDB CPU time % in total time : .09 %

Shared buffer hit rate:

Shared Buffer Reads : 1057

Shared Buffer Hits : 139798

Shared Buffer Hit ratio : 99.25 %

In memory sort rate:

In-memory sort count : 0

In-disk sort count : 0

In-memory sort ratio : 0

I/O usage:

Number of files : 106

Physical Reads : 584

Physical Writes : 362

Read Time : 5794 ms

Write Time : 4046 ms

Disk usage:

DB size : 57 MB

Total Physical writes : 362

Average Physical write : 89471.08

Maximum Physical write : 362

Activity statistics:

Active SQL count : 4

Session count : 5

Node statistics information:

dn\_6001:

MPPDB CPU Time : 38960 Jiffies

Host CPU Busy Time : 761060 Jiffies

Host CPU Total Time : 45719980 Jiffies

MPPDB CPU Time % in Busy Time : 5.12 %

MPPDB CPU Time % in Total Time : .09 %

Physical memory : 7144341504 Bytes

DB Memory usage : 14922285056 Bytes

Shared buffer size : 1073741824 Bytes

Shared buffer hit ratio : 99.25 %

Sorts in memory : 0

Sorts in disk : 0

In-memory sort ratio : 0

Number of files : 106

Physical Reads : 584

Physical Writes : 362

Read Time : 5794

Write Time : 4046

Session statistics information(Top 10):

Session CPU statistics:

1 dn\_6001-postgres-omm:

Session CPU time : 2

Database CPU time : 39020

Session CPU time % : .01 %

……………

Session Memory statistics:

1 dn\_6001-postgres-omm:

Buffer Reads : 1309

Shared Buffer Hit ratio : 93.03

In Memory sorts : 0

In Disk sorts : 0

In Memory sorts ratio : 0

Total Memory Size : 7433136

Used Memory Size : 6443268

………………..

Session IO statistics:

1 dn\_6001-postgres-omm:

Physical Reads : 98

Read Time : 1069

2 dn\_6001-postgres-omm:

Physical Reads : 13

Read Time : 173

……….........

[omm@ecs-e1b3 ~]$

gs\_checkperf 检查实验结束。

## 通过EXPLAIN进行SQL语句优化

说明：

* 使用explain能显示SQL语句的执行计划；
* 执行计划将显示SQL语句所引用的表会采用什么样的扫描方式，如：简单的顺序扫描、索引扫描等，如果引用了多个表，执行计划还会显示用到的JOIN算法；
* 执行计划的最关键的部分是语句的预计执行开销，这是计划生成器估算执行该语句将花费多长的时间；
* 若指定了ANALYZE选项，则该语句模拟执行并形成最优的执行计划（并非真正执行），然后根据实际的运行结果显示统计数据，包括每个计划节点内时间总开销（毫秒为单位）和实际返回的总行数。这对于判断计划生成器的估计是否接近现实非常有用。

（如未登录则进行操作，连续实验可忽略此步骤）用root用户登录装有openGauss数据库服务的操作系统然后用 su – omm命令切换至OMM用户环境，登录后信息如下。

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Tue Jul 21 09:21:11 CST 2021

System load: 0.01

Processes: 109

Memory used: 6.7%

Swap used: 0.0%

Usage On: 15%

IP address: 192.168.0.96

Users online: 1

[root@ecs-e1b3 ~]# **su - omm**

Last login: Fri Jul 10 19:05:39 CST 2021 on pts/0

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Tue Jul 21 09:21:25 CST 2021

System load: 0.01

Processes: 111

Memory used: 7.0%

Swap used: 0.0%

Usage On: 15%

IP address: 192.168.0.96

Users online: 1

[omm@ecs-e1b3 ~]$

启动、连接数据库。

启动数据库服务。

[omm@ecs-e1b3 ~]$ **gs\_om -t start;**

Starting cluster.

=========================================

=========================================

Successfully started.

然后连接postgres数据库。

[omm@ecs-e1b3 ~]$ **gsql -d postgres -p 26000 -r**

gsql ((openGauss 2.0.0 build 38a9312a) compiled at 2021-05-27 14:56:08 commit 472 last mr 549 )

Non-SSL connection (SSL connection is recommended when requiring high-security)

Type ＂help＂ for help.

postgres=#

创建student表。

postgres=# **CREATE TABLE student**

**( std\_id INT NOT NULL,**

**std\_name VARCHAR(20) NOT NULL,**

**std\_sex VARCHAR(6),**

**std\_birth DATE,**

**std\_in DATE NOT NULL,**

**std\_address VARCHAR(100)**

**);**

CREATE TABLE

插入表数据。

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (1,'张一','男','1993-01-01','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (2,'张二','男','1993-01-02','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (3,'张三','男','1993-01-03','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (4,'张四','男','1993-01-04','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (5,'张五','男','1993-01-05','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (6,'张六','男','1993-01-06','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (7,'张七','男','1993-01-07','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (8,'张八','男','1993-01-08','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (9,'张九','男','1993-01-09','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (10,'李一','男','1993-01-10','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (11,'李二','男','1993-01-11','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (12,'李三','男','1993-01-12','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (13,'李四','男','1993-01-13','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (14,'李五','男','1993-01-14','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (15,'李六','男','1993-01-15','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (16,'李七','男','1993-01-16','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (17,'李八','男','1993-01-17','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (18,'李九','男','1993-01-18','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (19,'王一','男','1993-01-19','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (20,'王二','男','1993-01-20','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (21,'王三','男','1993-01-21','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (22,'王四','男','1993-01-22','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (23,'王五','男','1993-01-23','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (24,'王六','男','1993-01-24','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (25,'王七','男','1993-01-25','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (26,'王八','男','1993-01-26','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (27,'王九','男','1993-01-27','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (28,'钱一','男','1993-01-28','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (29,'钱二','男','1993-01-29','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (30,'钱三','男','1993-01-30','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (31,'钱四','男','1993-02-01','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (32,'钱五','男','1993-02-02','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (33,'钱六','男','1993-02-03','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (34,'钱七','男','1993-02-04','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (35,'钱八','男','1993-02-05','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (36,'钱九','男','1993-02-06','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (37,'吴一','男','1993-02-07','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (38,'吴二','男','1993-02-08','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (39,'吴三','男','1993-02-09','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (40,'吴四','男','1993-02-10','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (41,'吴五','男','1993-02-11','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (42,'吴六','男','1993-02-12','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (43,'吴七','男','1993-02-13','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (44,'吴八','男','1993-02-14','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (45,'吴九','男','1993-02-15','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (46,'柳一','男','1993-02-16','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (47,'柳二','男','1993-02-17','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (48,'柳三','男','1993-02-18','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (49,'柳四','男','1993-02-19','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (50,'柳五','男','1993-02-20','2011-09-01','江苏省南京市雨花台区');

查询统计数据。

postgres=# **select count(\*) from student;**

count

-------

50

(1 row)

postgres=# **select \* from student order by std\_id;**

std\_id | std\_name | std\_sex | std\_birth | std\_in | std\_address

--------+----------+---------+---------------------+---------------------+----------------------

1 | 张一 | 男 | 1993-01-01 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

2 | 张二 | 男 | 1993-01-02 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

3 | 张三 | 男 | 1993-01-03 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

4 | 张四 | 男 | 1993-01-04 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

……………..

查看表信息。

postgres=# **\d student**

Table ＂public.student＂

Column | Type | Modifiers

-------------+--------------------------------+-----------

std\_id | integer | not null

std\_name | character varying(20) | not null

std\_sex | character varying(6) |

std\_birth | timestamp(0) without time zone |

std\_in | timestamp(0) without time zone | not null

std\_address | character varying(100) |

收集表的统计信息。

postgres=# **ANALYZE VERBOSE student;**

INFO: analyzing ＂public.student＂(dn\_6001 pid=48036)

INFO: ANALYZE INFO : ＂student＂: scanned 1 of 1 pages, containing 50 live rows and 0 dead rows; 50 rows in sample, 50 estimated total rows(dn\_6001 pid=48036)

ANALYZE

使用ANALYZE VERBOSE语句更新统计信息，会同时输出表的相关信息。

查看语句的执行计划。（查询结果截图）

postgres=# **explain select \* from student where std\_id=30;**

QUERY PLAN

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

Seq Scan on student (cost=0.00..1.62 rows=1 width=62)

Filter: (std\_id = 30)

(2 rows)

Seq Scan on student 表示使用的是全表扫描。

给表添加主键。

postgres=# **alter table student add primary key (std\_id);**

NOTICE: ALTER TABLE / ADD PRIMARY KEY will create implicit index ＂student\_pkey＂ for table ＂student＂

ALTER TABLE

再次查看表信息。

确定主键是否建好。

postgres=# **\d student**

Table ＂public.student＂

Column | Type | Modifiers

-------------+--------------------------------+-----------

std\_id | integer | not null

std\_name | character varying(20) | not null

std\_sex | character varying(6) |

std\_birth | timestamp(0) without time zone |

std\_in | timestamp(0) without time zone | not null

std\_address | character varying(100) |

Indexes:

**＂student\_pkey＂ PRIMARY KEY, btree (std\_id) TABLESPACE pg\_default**

student\_pkey 为主键名称。

通过hint来优化语句扫描方式。

通过加hint来使查询语句进行索引扫描。（查询结果截图）

postgres=# **explain select /\*+indexscan(student student\_pkey)\*/ \* from student where std\_id=30;**

QUERY PLAN

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

[Bypass]

Index Scan using student\_pkey on student (cost=0.00..8.27 rows=1 width=62)

Index Cond: (std\_id = 30)

(3 rows)

postgres=#

Index Scan using student\_pkey on student 表示语句通过student表上的主键索引student\_pkey进行了索引扫描。

退出数据库。

postgres=# **\q**

EXPLAIN进行SQL优化实验结束。

思考：两种扫描方式的区别是什么？具体是如何实现的？比较两种扫描方式的cost（提供查询结果截图）,为什么全表扫描比索引扫描cost更小？在什么情况下通过主键进行查找会比全表扫描更节省时间？

# 最大连接数设置

## 实验介绍

### 关于本实验

当应用程序与数据库的连接数超过最大值，则新的连接无法建立。建议对连接数进行监控，及时释放空闲的连接或者增加最大连接数。

本实验主要是讲如何来设置数据库最大连接个数。

### 实验目的

掌握对数据库最大连接数的设置方法。

## 场景设置及操作步骤

用ROOT用户登录装有openGauss数据库服务的操作系统然后用 su – omm命令切换至OMM用户环境，登录后信息如下。

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Mon Jul 27 11:22:46 CST 2021

System load: 0.03

Processes: 154

---------

IP address: 192.168.0.12

Users online: 3

[root@ecs-e1b3 ~]# **su - omm**

Last login: Mon Jul 27 09:23:44 CST 2021 on pts/0

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Mon Jul 27 11:23:37 CST 2021

System load: 0.01

Processes: 156

Memory used: 2.4%

Swap used: 0.0%

Usage On: 14%

IP address: 192.168.0.12

Users online: 3

确认openGauss数据库服务是否启动。

[omm@ecs-e1b3 ~]$ **gs\_om -t status;**

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

cluster\_name : dbCluster

cluster\_state : Normal

redistributing : No

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

cluster\_state : Normal 表示已启动，可以正常使用。如果状态为非Normal表示不可用

为了实验场景设置，如果数据库服务没有启动，请执行步gs\_om -t start 命令启动服务。

登录数据库。

使用gsql客户端以管理员用户身份连接postgres数据库，假设端口号为26000。

[omm@ecs-e1b3 ~]$ **gsql -d postgres -p 26000 -r**

gsql ((openGauss 2.0.0 build 38a9312a) compiled at 2021-05-27 14:57:08 commit 472 last mr 549 )

Non-SSL connection (SSL connection is recommended when requiring high-security)

Type ＂help＂ for help.

postgres=#

查看当前数据库已使用的连接数。

postgres=# **select count(1) from pg\_stat\_activity;**

count

-------

10

(1 row)

10表示当前有10个应用已连接到数据库。

查看数据库设置的最大连接数。

postgres=# **SHOW max\_connections;**

max\_connections

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

5000

(1 row)

5000 表示数据库设置的最大连接个数为5000。如果当前数据库已使用的连接数快接近于最大连接数时，运维人员先要果断的增加最大连接数以防系统新的连接无法建立。

调整最大连接数参数。

用alter system set 语句来设置此参数，如下。

[omm@ecs-e1b3 ~]$ **gsql -d postgres -p 26000 -r**

gsql ((openGauss 2.0.0 build 38a9312a) compiled at 2021-05-27 14:57:08 commit 472 last mr 549 )

Non-SSL connection (SSL connection is recommended when requiring high-security)

Type ＂help＂ for help.

postgres=# **alter system set max\_connections=6000;**

NOTICE: please restart the database for the POSTMASTER level parameter to take effect.

ALTER SYSTEM SET

postgres=#**\q**

重启数据库。

gs\_om -t stop先关闭数据库,然后用gs\_om -t start再启动数据库。

[omm@ecs-e1b3 ~]$ **gs\_om -t stop;**

Stopping cluster.

=========================================

Successfully stopped cluster.

=========================================

End stop cluster.

[omm@ecs-e1b3 ~]$ **gs\_om -t start;**

Starting cluster.

=========================================

=========================================

Successfully started.

验证参数设置是否成功。（截图验证）

使用gsql客户端以管理员用户身份连接postgres数据库，然后查看参数值。

[omm@ecs-e1b3 ~]$ **gsql -d postgres -p 26000 -r**

gsql ((openGauss 2.0.0 build 38a9312a) compiled at 2021-05-27 14:57:08 commit 472 last mr 549 )

Non-SSL connection (SSL connection is recommended when requiring high-security)

Type ＂help＂ for help.

postgres=# **SHOW max\_connections;**

max\_connections

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

6000

(1 row)

这里显示max\_connections 为 6000，说明前面参数的修改已经生效。

退出数据库。

postgres=#**\q**

最大连接数设置实验结束。

# 例行表、索引的维护

## 实验介绍

### 关于本实验

为了保证数据库的有效运行，数据库必须在插入/删除操作后，基于客户场景，定期做VACUUM FULL和ANALYZE，更新统计信息，以便获得更优的性能。

* VACUUM FULL可回收已更新或已删除的数据所占据的磁盘空间，同时将小数据文件合并；
* VACUUM对每个表维护了一个可视化映射来跟踪包含对别的活动事务可见的数组的页。一个普通的索引扫描首先通过可视化映射来获取对应的数组，来检查是否对当前事务可见。若无法获取，再通过堆数组抓取的方式来检查。因此更新表的可视化映射，可加速唯一索引扫描；
* ANALYZE可收集与数据库中表内容相关的统计信息。统计结果存储在系统表PG\_STATISTIC中。查询优化器会使用这些统计数据，生成最有效的执行计划。
* 数据库经过多次删除操作后，索引页面上的索引键将被删除，造成索引膨胀。例行重建索引，可有效的提高查询效率。

本实验主要是通过使用VACUUM、VACUUM FULL FULL来收缩表，用ANALYZE来收集表的统计信息以及对表上的索引进行重建。

### 实验目的

* 掌握VACUUM、VACUUM FULL FULL、ANALYZE基本的使用及如何重建索引；

## 场景设置及操作步骤

用ROOT用户登录装有openGauss数据库服务的操作系统然后用 su – omm命令切换至OMM用户环境，登录后信息如下。（连续实验可忽略此步骤）

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Tue Jul 27 16:21:11 CST 2021

System load: 0.01

Processes: 109

Memory used: 6.7%

Swap used: 0.0%

Usage On: 15%

IP address: 192.168.0.96

Users online: 1

[root@ecs-e1b3 ~]# **su - omm**

Last login: Fri Jul 27 16:22:11 CST 2021 on pts/0

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Tue Jul 27 16:21:11 CST 2021

System load: 0.01

Processes: 111

Memory used: 7.0%

Swap used: 0.0%

Usage On: 15%

IP address: 192.168.0.96

Users online: 1

[omm@ecs-e1b3 ~]$

启动服务器后，然后使用gsql客户端以管理员用户身份连接postgres数据库，假设端口号为26000。

启动数据库服务。

[omm@ecs-e1b3 ~]$ **gs\_om -t start;**

Starting cluster.

=========================================

=========================================

Successfully started.

连接postgres数据库。

[omm@ecs-e1b3 ~]$ **gsql -d postgres -p 26000 -r**

gsql ((openGauss 2.0.0 build 38a9312a) compiled at 2021-05-27 14:56:08 commit 472 last mr 549 )

Non-SSL connection (SSL connection is recommended when requiring high-security)

Type ＂help＂ for help.

postgres=#

创建student表。（连续实验可忽略步骤三、步骤四，因为student表已经在实验2中创建并插入数据）

postgres=# **drop table student;**

postgres=# **CREATE TABLE student**

**( std\_id INT NOT NULL,**

**std\_name VARCHAR(20) NOT NULL,**

**std\_sex VARCHAR(6),**

**std\_birth DATE,**

**std\_in DATE NOT NULL,**

**std\_address VARCHAR(100)**

**);**

CREATE TABLE

插入表数据。

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (1,'张一','男','1993-01-01','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (2,'张二','男','1993-01-02','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (3,'张三','男','1993-01-03','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (4,'张四','男','1993-01-04','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (5,'张五','男','1993-01-05','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (6,'张六','男','1993-01-06','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (7,'张七','男','1993-01-07','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (8,'张八','男','1993-01-08','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (9,'张九','男','1993-01-09','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (10,'李一','男','1993-01-10','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (11,'李二','男','1993-01-11','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (12,'李三','男','1993-01-12','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (13,'李四','男','1993-01-13','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (14,'李五','男','1993-01-14','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (15,'李六','男','1993-01-15','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (16,'李七','男','1993-01-16','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (17,'李八','男','1993-01-17','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (18,'李九','男','1993-01-18','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (19,'王一','男','1993-01-19','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (20,'王二','男','1993-01-20','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (21,'王三','男','1993-01-21','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (22,'王四','男','1993-01-22','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (23,'王五','男','1993-01-23','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (24,'王六','男','1993-01-24','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (25,'王七','男','1993-01-25','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (26,'王八','男','1993-01-26','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (27,'王九','男','1993-01-27','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (28,'钱一','男','1993-01-28','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (29,'钱二','男','1993-01-29','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (30,'钱三','男','1993-01-30','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (31,'钱四','男','1993-02-01','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (32,'钱五','男','1993-02-02','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (33,'钱六','男','1993-02-03','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (34,'钱七','男','1993-02-04','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (35,'钱八','男','1993-02-05','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (36,'钱九','男','1993-02-06','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (37,'吴一','男','1993-02-07','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (38,'吴二','男','1993-02-08','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (39,'吴三','男','1993-02-09','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (40,'吴四','男','1993-02-10','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (41,'吴五','男','1993-02-11','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (42,'吴六','男','1993-02-12','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (43,'吴七','男','1993-02-13','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (44,'吴八','男','1993-02-14','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (45,'吴九','男','1993-02-15','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (46,'柳一','男','1993-02-16','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (47,'柳二','男','1993-02-17','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (48,'柳三','男','1993-02-18','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (49,'柳四','男','1993-02-19','2011-09-01','江苏省南京市雨花台区');

INSERT INTO student(std\_id,std\_name,std\_sex,std\_birth,std\_in,std\_address) VALUES (50,'柳五','男','1993-02-20','2011-09-01','江苏省南京市雨花台区');

查询统计数据。（查看全表内容时，按回车键可查看更多内容）

postgres=# **select count(\*) from student;**

count

-------

50

(1 row)

postgres=# **select \* from student order by std\_id;**

std\_id | std\_name | std\_sex | std\_birth | std\_in | std\_address

--------+----------+---------+---------------------+---------------------+----------------------

1 | 张一 | 男 | 1993-01-01 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

2 | 张二 | 男 | 1993-01-02 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

3 | 张三 | 男 | 1993-01-03 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

4 | 张四 | 男 | 1993-01-04 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

5 | 张五 | 男 | 1993-01-05 00:00:00 | 2011-09-01 00:00:00 | 江苏省南京市雨花台区

……………..

查看表信息。

postgres=# **\d student**

Table ＂public.student＂

Column | Type | Modifiers

-------------+--------------------------------+-----------

std\_id | integer | not null

std\_name | character varying(20) | not null

std\_sex | character varying(6) |

std\_birth | timestamp(0) without time zone |

std\_in | timestamp(0) without time zone | not null

std\_address | character varying(100) |

使用VACUUM命令，进行磁盘空间回收。

postgres=# **vacuum student;**

VACUUM

删除表中数据。

postgres=# **delete from student where std\_id>30;**

DELETE 20

使用VACUUM FULL命令，进行磁盘空间回收。

postgres=# **vacuum full student;**

VACUUM

使用ANALYZE语句更新统计信息。

postgres=# **analyze student;**

ANALYZE

使用ANALYZE VERBOSE语句更新统计信息，并输出表的相关信息。（该步骤截图）

postgres=# **analyze verbose student;**

INFO: analyzing ＂public.student＂(dn\_6001 pid=37195)

INFO: ANALYZE INFO : ＂student＂: scanned 1 of 1 pages, containing 30 live rows and 20 dead rows; 30 rows in sample, 30 estimated total rows(dn\_6001 pid=37195)

ANALYZE

执行VACUUM ANALYZE命令进行查询优化。

postgres=# **vacuum analyze student;**

VACUUM

查看特定表的统计信息。

postgres=# **select relname,n\_tup\_ins,n\_tup\_upd,n\_tup\_del,last\_analyze,vacuum\_count from PG\_STAT\_ALL\_TABLES where relname='student';**

relname | n\_tup\_ins | n\_tup\_upd | n\_tup\_del | last\_analyze | vacuum\_count

---------+-----------+-----------+-----------+------------------------------+--------------

student | 50 | 0 | 20 | 2021-07-27 17:07:19.17167+08 | 3

(1 row)

postgres=#

PG\_STAT\_ALL\_TABLES视图将包含当前数据库中每个表的一行统计信息，以上查询结果中各列分别表示：

Relname 表名

n\_tup\_ins 插入行数

n\_tup\_upd 更新行数

n\_tup\_del 删除行数

last\_analyze 上次手动分析该表的时间

vacuum\_count 这个表被手动清理的次数

索引维护。

说明：

* 如果数据发生大量删除后，索引页面上的索引键将被删除，导致索引页面数量的减少，造成索引膨胀。重建索引可回收浪费的空间。
* 新建的索引中逻辑结构相邻的页面，通常在物理结构中也是相邻的，所以一个新建的索引比更新了多次的索引访问速度要快。
* 重建索引有以下两种方式：

1. 使用REINDEX语句重建索引；

2、先删除索引（DROP INDEX），再创建索引（CREATE INDEX）。

先在student表的std\_name列上创建一个索引，如下。

postgres=# **create index inx\_stu01 on student(std\_name);**

CREATE INDEX

postgres=#

方式1：使用REINDEX语句重建索引，具体如下。

postgres=# **reindex table student;**

REINDEX

postgres=#

方式2：先删除索引（DROP INDEX），再创建索引（CREATE INDEX），具体如下。

postgres=# **drop index inx\_stu01;**

DROP INDEX

postgres=# **create index inx\_stu01 on student(std\_name);**

CREATE INDEX

postgres=#

查看表结构信息，具体如下。

postgres=# **\d student;**

Table ＂public.student＂

Column | Type | Modifiers

-------------+--------------------------------+-----------

std\_id | integer | not null

std\_name | character varying(20) | not null

std\_sex | character varying(6) |

std\_birth | timestamp(0) without time zone |

std\_in | timestamp(0) without time zone | not null

std\_address | character varying(100) |

Indexes:

＂inx\_stu01＂ btree (std\_name) TABLESPACE pg\_default

退出数据库。

postgres=#**\q**

例行表、索引的维护实验结束。

思考：什么时候需要重建索引？为什么？openGauss中重建索引的方式有哪些？

# 附录一：Linux操作系统相关命令

Linux中的命令格式为：command [options] [arguments] 中括号表示可选的，即有些命令不需要选项也不需要参数，但有的命令在运行时需要多个选项或参数。

* options(选项)：选项是调整命令执行行为的开关，选项的不同决定了命令的显示结果不同。
* agruments(参数)：参数是指命令的作用对象。

## vi/vim

文本编辑器，若文件存在则是编辑，若不存在则是创建并编辑文本。

命令语法：

vim [参数]

参数说明：可编辑的文件名。

命令示例：

* 编辑名为clusterconfig的xml文本。

vim clusterconfig.xml

注：

vim编辑器有以下三种模式：

* 正常模式：其它模式下按Esc或Ctrl+[进入，左下角显示文件名或为空。
* 插入模式：正常模式下按i键进入，左下角显示--INSERT--。
* 可视模式：正常模式下按v键进入，左下角显示--VISUAL--。

退出命令（正常模式下）：

* :wq 保存并退出。
* :q! 强制退出并忽略所有更改。
* :e! 放弃所有修改，并打开原有文件。

## cd

显示当前目录的名称，或切换当前的目录（打开指定目录）。

命令语法：

cd [参数]

参数说明：

* 无参数：切换用户当前目录。
* . ：表示当前目录。
* .. ：表示上一级目录。
* ~ ：表示home目录。
* / ：表示根目录。

命令示例：

* 切换到usr目录下的bin目录中。

cd /usr/bin

* 切换到用户home目录。

cd

* 切换到当前目录(cd后面接一个.)。

cd .

* 切换到当前目录上一级目录(cd后面接两个.)。

cd ..

* 切换到用户home目录。

cd ~

* 切换到根目录下。

cd /

注：切换目录需要理解绝对路径和相对路径这两个概念。

* 绝对路径：在Linux中，绝对路径是从/（即根目录）开始的，例如 /opt/software、/etc/profile, 如果目录以 / 就是绝对目录。
* 相对路径：是以 . 或 .. 开始的目录。 . 表示用户当前操作所在的位置，而 .. 表示上级目录。例如 ./gs\_om 表示当前目录下的文件或者目录。

## mv

文件或目录改名(move (rename) files)或将文件或目录移入其它位置，经常用来备份文件或者目录。

命令语法：

mv [选项] 参数1 参数2

常用选项：

* -b：若需覆盖文件，则覆盖前先行备份。

参数说明：

* 参数1：源文件或目录。
* 参数2：目标文件或目录。

命令示例：

* 将文件python重命名为python.bak。

mv python python.bak

* 将/physical/backup目录下的所有文件和目录移到/data/dbn1目录下。

mv /physical/backup/\* /data/dbn1

## curl

在Linux中curl是一个利用URL规则在命令行下工作的文件传输工具。支持文件的上传和下载，是综合传输工具。

命令语法：

curl [选项] [URL]

常用选项：

* -A/--user-agent <string>：设置用户代理发送给服务器；
* -C/--continue-at <offset>：断点续转；
* -D/--dump-header <file>：把header信息写入到该文件中；
* -e/--referer：来源网址；
* -o/--output：把输出写到该文件中；
* -O/--remote-name：把输出写到该文件中，保留远程文件的文件名；
* -s/--silent：静默模式。不输出任何东西；
* -T/--upload-file <file>：上传文件；
* -u/--user <user[:password]>：设置服务器的用户和密码；
* -x/--proxy <host[:port]>：在给定的端口上使用HTTP代理；
* -#/--progress-bar：进度条显示当前的传送状态。

参数说明：

* URL：指定的文件传输URL地址。

命令示例：

* 将url(<https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo)的内容保存到/etc/yum.repos.d/CentOS-Base.repo>文件中。

curl -o /etc/yum.repos.d/CentOS-Base.repo <https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo>

* 如果在传输过程中掉线，可以使用-C的方式进行续传。

curl -C -O <https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo>

## yum

Shell 前端软件包管理器。基于 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载和安装。

命令语法：

yum [options] [command] [package ...]

常用选项：

* -h：查看帮助；
* -y：当安装过程提示选择全部为 "yes"；
* -q：不显示安装的过程。

参数说明：

* command：要进行的操作；
* package：安装的包名。

命令示例：

* 列出所有可更新的软件清单命令。

yum check-update

* 更新所有软件命令。

yum update

* 列出所有可安装的软件清单命令。

yum list

* 安装指定的软件。

yum install -y libaio-devel flex bison ncurses-devel glibc.devel patch lsb\_release wget python3

## wget

wget是Linux下下载文件的最常用命令。wget支持HTTP，HTTPS和FTP协议，支持自动下载,即可以在用户退出系统后在后台执行,直到下载结束。

命令语法：

wget [选项] [URL]

常用选项:

* -c：接着下载没下载完的文件；
* -b：启动后转入后台执行；
* -P：指定下载目录；
* -O：变更下载文件名；
* --ftp-user --ftp-password：使用FTP用户认证下载。

参数说明：

* 指定的文件下载URL地址。

命令示例：

* 下载openGauss数据库安装文件到当前文件夹。

wget [https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.1.0/x86/openGauss-1.1.0-CentOS-64bit.tar.gz](https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/x86/openGauss-1.0.0-CentOS-64bit.tar.gz)

* 使用wget断点续传。

wget –c [https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.1.0/x86/openGauss-1.1.0-CentOS-64bit.tar.gz](https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/x86/openGauss-1.0.0-CentOS-64bit.tar.gz)

## ln

为某一个文件在另外一个位置建立一个同步的链接（软硬链接，不带选项为硬链接）。

当需要在不同的目录，用到相同的文件时，就不需要在每一个需要的目录下都放一个必须相同的文件，我们只要在某个固定的目录，放上该文件，然后在其它的目录下用ln命令链接（link）它就可以，不必重复的占用磁盘空间。

命令语法：

ln [选项] 参数1 参数2

常用选项：

* -b --删除，覆盖以前建立的链接；
* -d --允许超级用户制作目录的硬链接；
* -s --软链接(符号链接)。

参数说明：

* 参数1：源文件或目录。
* 参数2：被链接的文件或目录。

命令示例：

* 为python3文件创建软链接/usr/bin/python，如果python3丢失，/usr/bin/python将失效。

ln -s python3 /usr/bin/python

* 为python3创建硬链接/usr/bin/python，python3与/usr/bin/python的各项属性相同。

ln python3 /usr/bin/python

## mkdir

创建指定的名称的目录，要求创建目录的用户在当前目录中具有写权限，并且指定的目录名不能是当前目录中已有的目录。

命令语法：

mkdir [选项] [参数]

常用选项：

* -p --可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此选项后,系统将自动建立好那些尚不存在的目录,即一次可以建立多个目录（递归）；
* -v --每次创建新目录都显示信息；
* -m --设定权限<模式> (类似chmod)，而不是rwxrwxrwx减umask。

参数说明：

* 需要创建的目录。

命令示例：

* 创建一个空目录。

mkdir test

* 递归创建多个目录。

mkdir -p /opt/software/openGauss

* 创建权限为777的目录（目录的权限为rwxrwxrwx）：

mkdir –m 777 test

## chmod

更改文件权限。

命令语法：

chmod [选项] <mode> <file...>

常用选项：

* -R, --以递归的方式对目前目录下的所有文件与子目录进行相同的权限变更。

参数说明：

* mode：权限设定字串，详细格式如下 。

[ugoa...][[+-=][rwxX]...][,...]，

其中，[ugoa...]：u 表示该档案的拥有者，g 表示与该档案的拥有者属于同一个群体(group)者，o 表示其他以外的人，a 表示所有（包含上面三者）；[+-=]：+ 表示增加权限，- 表示取消权限，= 表示唯一设定权限；[rwxX]：r 表示可读取，w 表示可写入，x 表示可执行，X表示只有当该档案是个子目录或者该档案已经被设定过为可执行。

* file：文件列表（单个或者多个文件、文件夹）。

命令示例：

* 设置所有用户可读取文件 cluterconfig.xml。

chmod ugo+r cluterconfig.xml

或

chmod a+r cluterconfig.xml

* 设置当前目录下的所有档案与子目录皆设为任何人可读写。

chmod -R a+rw \*

数字权限使用格式：

* 这种使用方式中，规定数字4、2和1表示读、写、执行权限，即r=4,w=2,x=1。
* 例：rwx = 7（4+2+1）；rw = 6（4+2）；r-x = 5 （4+0+1）；r-- = 4（4+0+0）；--x = 1（0+0+1）；

每个文件都可以针对三个粒度，设置不同的rwx(读写执行)权限。即我们可以用三个8进制数字分别表示 拥有者 、群组 、其它组( u、 g 、o)的权限详情，并用chmod直接加三个8进制数字的方式直接改变文件权限。语法格式为。

chmod <abc> file...

其中，a，b，c各为一个数字，分别代表User、Group、及Other的权限，相当于简化版的chmod u=权限,g=权限,o=权限 file...，而此处的权限将用8进制的数字来表示User、Group、及Other的读、写、执行权限。

命令示例：

* 赋予cluterconfig.xml文件可读可写可执行权限（所有权限）。

chmod 777 cluterconfig.xml

* 赋予/opt/software/openGauss目录下所有文件及其子目录 用户所有权限组可读可执行权限，其他用户可读可执行权限。

chmod R 755 /opt/software/openGauss

## chown

利用 chown 将指定文件的拥有者改为指定的用户或组，用户可以是用户名或者用户ID；组可以是组名或者组ID；文件是以空格分开的要改变权限的文件列表，支持通配符。只有系统管理者(root)才有这样的权限。使用权限 : **root**。

命令语法：

chown [选项] user[:group] file...

常用选项:

* -c : 显示更改的部分的信息；
* -f : 忽略错误信息；
* -R : 处理指定目录以及其子目录下的所有文件。

参数说明

* user : 新的文件拥有者的使用者 ID。
* group : 新的文件拥有者的使用者组(group)。
* flie：文件。

命令示例：

* 将文件 file1.txt 的拥有者设为omm，群体的使用者dbgrp。

chown omm:dbgrp /opt/software/openGauss/clusterconfig.xml

* 将目前目录下的所有文件与子目录的拥有者皆设为omm，群体的使用者dbgrp。

chown -R omm:dbgrp \*

## ls

列出文件和目录的内容。

命令语法：

ls [选项] [参数]

常用选项：

* -l --以长格式显示，列出文件的详细信息，如创建者，创建时间，文件的读写权限列表等等；
* -a --列出文件下所有的文件，包括以"."和".."开头的隐藏文件 （Linux下文件隐藏文件是以 .开头的，如果存在 .. 代表存在着父目录）；
* -d --列出目录本身而非目录内的文件，通常要与-l一起使用；
* -R --同时列出所有子目录层，与-l相似，只是不显示出文件的所有者，相当于编程中的＂递归＂实现；
* -t --按照时间进行文件的排序，Time（时间）；
* -s --在每个文件的后面打印出文件的大小，size（大小）；
* -S --以文件的大小进行排序。

参数说明：

* 目录或文件。

命令示例：

* 以长格式列出当前目录中的文件及目录。

ls -l

## cp

复制文件或者目录。

命令语法：

cp [选项] 参数1 参数2

常用选项：

* -f --如果目标文件无法打开则将其移除并重试(当 -n 选项存在时则不需再选此项)；
* -n --不要覆盖已存在的文件(使前面的 -i 选项失效)；
* -I --覆盖前询问(使前面的 -n 选项失效)；
* -p --保持指定的属性(默认：模式，所有权，时间戳)，如果可能保持附加属性：环境、链接、xattr 等；
* -R,-r --复制目录及目录内的所有项目。

参数说明：

* 参数1：源文件。
* 参数2：目标文件。

命令示例：

* 将home目录中的abc文件复制到opt目录下。

cp /home/abc /opt

注：目标文件存在时，会询问是否覆盖。这是因为cp是cp -i的别名。目标文件存在时，即使加了-f标志，也还会询问是否覆盖。

## rm

删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件，只是删除了链接，原有文件均保持不变。

rm是一个危险的命令，使用的时候要特别当心，否则整个系统就会毁在这个命令（比如在/（根目录）下执行rm \* rf）。所以，我们在执行rm之前最好先确认一下在哪个目录，到底要删除什么东西，操作时保持高度清醒的头脑。

命令语法：

rm [选项] 文件

常用选项：

* -f --忽略不存在的文件，从不给出提示；
* -r --指示rm将参数中列出的全部目录和子目录均递归地删除。

参数说明：

* 需要删除的文件或目录。

命令示例：

* 删除文件。

rm qwe

注：输入rm qwe命令后，系统会询问是否删除，输入y后就会删除文件，不想删除文件则输入n。

* 强制删除某个文件。

rm-rf clusterconfig.log

## cat

连接文件并在标准输出上输出。这个命令常用来显示文件内容，或者将几个文件连接起来显示，或者从标准输入读取内容并显示，它常与重定向符号配合使用。

命令语法：

cat [选项] [参数]

常用选项：

* -E --在每行结束显示$；
* -n –由1开始对给所有输出行编号；
* -b 或 --number-nonblank：和 -n 相似，只不过对于空白行不编号；
* -v --使用 ^ 和 M- 符号，除了 LFD 和 TAB 之外。

参数说明：

* 可操作的文件名。

命令示例:

* 显示testfile文件的内容。

cat textfile

* 把 textfile1 和 textfile2 的文档内容加上行号（空白行不加）之后将内容追加到 textfile3 文档里。

cat -b textfile1 textfile2 >> textfile3

* 向/etc/profile中追加内容（输入EOF表示结束追加）。

cat >>/etc/profile<<EOF

>export LD\_LIBRARY\_PATH=$packagePath/script/gspylib/clib:$LD\_LIBRARY\_PATH

>EOF

注：

* EOF是end of file的缩写，表示＂文字流＂（stream）的结尾。＂文字流＂可以是文件（file），也可以是标准输入（stdin）。在Linux系统之中，EOF是当系统读取到文件结尾，所返回的一个信号值（也就是-1）。

# 附录二：openGauss数据库基本操作

## 查看数据库对象

* 查看帮助信息。

postgres=# \?

* 切换数据库。

postgres=# \c dbname

* 列举数据库。

使用\l元命令查看数据库系统的数据库列表。

postgres=# \l

使用如下命令通过系统表pg\_database查询数据库列表。

postgres=# SELECT datname FROM pg\_database;

* 列举表。

postgres=# \dt

* 列举所有表、视图和索引。

postgres=# \d+

使用gsql的\d+命令查询表的属性。

postgres=# \d+ tablename

* 查看表结构。

*p*ostgres=# \d tablename

* 列举schema。

postgres=# \dn

* 查看索引。

postgres=# \di

* 查询表空间。

使用gsql程序的元命令查询表空间。

postgres=# \db

检查pg\_tablespace系统表。如下命令可查到系统和用户定义的全部表空间。

postgres=# SELECT spcname FROM pg\_tablespace;

* 查看数据库用户列表。

postgres=# SELECT \* FROM pg\_user;

* 查看用户属性。

postgres=# SELECT \* FROM pg\_authid;

* 查看所有角色。

postgres=# SELECT \* FROM PG\_ROLES;

## 其他操作

* 查看openGauss支持的所有SQL语句。

postgres=#\h

* 切换用户。

postgres=# \c – username

* 退出数据库。

postgres=# \q