

《数据库系统原理》课程实验指导

openGauss 数据库使用



2022年9月

目录

前 言.....	3
实验环境说明.....	3
1 gsql 客户端工具.....	4
1.1 实验介绍.....	4
1.2 gsql 客户端工具简介	4
1.3 gsql 客户端工具使用	4
1.3.1 查询数据库实例	5
1.3.2 查询端口号.....	5
1.3.3 启动数据库服务	6
1.3.4 本地连接数据库.....	6
1.3.5 执行单条字符串命令.....	7
1.3.6 使用文件作为命令源执行命令	7
1.3.7 将输出重定向到文件.....	9
1.3.8 发送会话日志到文件	10
1.3.9 列出数据库.....	11
1.3.10 编辑模式.....	11
1.3.11 安静模式	12
1.3.12 单行运行模式	13
1.3.13 打印 gsql 版本信息	13
1.3.14 自动断开连接	15
1.3.15 关闭数据库服务	15
2 openGauss 数据库基本操作	17
2.1 实验介绍.....	17
2.2 postgresSQL 简介	17
2.3 元命令	17
2.3.1 \d.....	17
2.3.2 其它	20
2.4 SQL 语句	21
2.4.1 创建管理用户.....	21
2.4.2 创建管理数据库.....	22
2.4.3 管理表.....	23
2.4.4 创建管理 schema	25
2.4.5 创建管理视图	26
2.4.6 创建管理序列	27

2.4.7 创建管理存储过程	28
2.4.8 创建管理全局临时表	29

前 言

实验环境说明

本实验环境为 virtualBOX 虚拟机 openEuler20.03 系统上的 openGauss1.1.0 数据库，实验数据采用 TD-LTE 网络配置数据库的十四张表，实验过程会用到 WinSCP，Putty 软件。

1 gsql 客户端工具

1.1 实验介绍

关于本实验

本实验主要描述 openGauss 数据库的 gsql 客户端工具的使用以及连接数据库的方法

实验目的

- 1.掌握 gsql 数据库开发调试工具的基本使用方法。
- 2.学会用 gsql 连接 openGauss 数据库。

1.2 gsql 客户端工具简介

gsql 是 openGauss 自带的客户端工具，用于数据库开发调试，安装 openGauss 时便以包含。gsql 提供在命令行下运行的数据库连接以及操作和维护。使用 gsql 连接数据库，可以交互式地输入、编辑、执行 SQL 语句。除了具备这些操作数据库的基本功能，gsql 还提供了若干高级特性，便于用户使用。

1.3 gsql 客户端工具使用

切换到 omm 用户，以操作系统用户 omm 登录数据库主节点。之后的操作都是在 omm 用户下进行的。

```
su - omm
```

```
[root@db1 ~]# su - omm
Last login: Tue Mar 16 15:13:31 CST 2021 on pts/0

Welcome to 4.19.90-2003.4.0.0036.oel.x86_64

System information as of time:  2021年 03月 17日 星期三 15:54:33 CST

System load:      0.00
Processes:        104
Memory used:      8.0%
Swap used:        0.0%
Usage On:         81%
IP address:       192.168.56.102
Users online:     2

[omm@db1 ~]$
```

1.3.1 查询数据库实例

使用“gs_om -t status --detail”命令查询 openGauss 各实例情况。（-t 指明 om 命令的类型，--detail 显示详细信息）

```
gs_om -t status --detail
```

结果如下：

```
[omm@db1 ~]$ gs_om -t status --detail
[ Cluster State ]

cluster_state      : Unavailable
redistributing     : No
current_az         : AZ_ALL

[ Datanode State ]

node  node_ip      instance          state
-----
1    db1 10.0.3.15    6001 /gaussdb/data/db1 P Primary Manually stopped
```

如上部署了数据库主节点实例的服务器 IP 地址为 10.0.3.15。数据库主节点数据路径为“/gaussdb/data/db1”。

1.3.2 查询端口号

在数据库主节点数据路径下的 postgresql.conf 文件中查看端口号信息。

```
cat /gaussdb/data/db1/postgresql.conf | grep port
```

结果如下：

```

[omm@db1 ~]$ cat /gaussdb/data/db1/postgresql.conf | grep port
port = 26000
# (change requires restart)
# amount of data between renegotiations, no longer supported
# Assigned by installation (change requires restart)
# Assigned by the operating system
# The heartbeat thread will not start if not set localheartbeatport and remoteheartbeatport.
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.3 remoteport=12212 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.4 remoteport=12312 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.6 remoteport=12312 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.7 remoteport=12312 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.8 remoteport=12312 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.9 remoteport=12312 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes
# e.g. 'localhost=10.145.130.2 localport=12311 localheartbeatport=12214 remotehost=10.145.130.9 remoteport=12312 remoteheartbeatport=12215, localhost=10.145.133.2 localport=12313 remotes

# *r = remote host and port
alarm_report_interval = 10
# Port of GTM
gsm_port = 6666
# Port1 of GTM
gsm_port1 = 6665
# Port2 of GTM
gsm_port2 = 6664
# Port3 of GTM
gsm_port3 = 6663
# Port4 of GTM
gsm_port4 = 6662
# Port5 of GTM
gsm_port5 = 6661
# Port6 of GTM
gsm_port6 = 6660
# Port7 of GTM
gsm_port7 = 6659
# Port the streaming router listens on, please keep the value (streaming_router_port = port) not change.
# Default port is 5438, setting by kernel, range 0 ~ 45898.
# value 0 means use default value (port + 4).
streaming_router_port = 5438

```

第一行显示的 port=26000, 便是数据库主节点的端口号。

注意：数据库主节点实例的服务器 IP 地址，数据路径和端口号，会因个人情况有所不同，在之后操作中请按照实际情况进行替换。

1.3.3 启动数据库服务

```
gs_om -t start
```

启动成功(下面的 gsql 命令除了 gsql --help 以外，都要在启动数据库服务后才能执行)：

```

[omm@db1 ~]$ gs_om -t start
Starting cluster.

=====

Successfully started.

```

1.3.4 本地连接数据库

启动数据库服务后，执行如下命令连接数据库。

```
gsql -d postgres -p 26000 -r
```

其中 postgres 为需要连接的数据库名称，26000 为数据库主节点的端口号。（-d 指定连接的数据库名称，postgres 为 openGauss 安装完成后默认生成的数据库，-p 指定连接的数据库服务器端口号，-r 使用 libedit 库连接，进入编辑模式）

连接成功后，系统显示类似如下信息：

```

[omm@db1 ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=#

```

omm 用户是管理员用户，因此系统显示“DBNAME=#”。若使用普通用户身份登录和连接数据库，系统显示“DBNAME=>”。

“Non-SSL connection”表示未使用 SSL 方式连接数据库。如果需要高安全性时，请用 SSL 进行安全的 TCP/IP 连接。

在 omm 用户下，若要使用其他用户连接该数据库（假设 jack 用户已经创建了，并且密码为 Jack1234，用户创建的介绍在实验 2.4 中）

```
[omm@db1 ~]$ gsql -d db_tpcc -p 26000 -U jack -W Jack1234 -r
```

其中，-U 表示连接数据库的用户名，-W 表示该用户名的密码

连接成功后便可以输入 SQL 语句或者 postgresSQL 元命令（交互式输入）来进行数据库操作了。

1.3.5 执行单条字符串命令

gsql 命令直接执行一条 SQL 语句命令或者元命令（-c 执行单条命令）

比如：显示版权信息的字符串元命令，\copyright

```
gsql -d postgres -p 26000 -c "\copyright"
```

```
[omm@db1 ~]$ gsql -d postgres -p 26000 -c "\copyright"
GaussDB Kernel Database Management System
Copyright (c) Huawei Technologies Co., Ltd. 2018. All rights reserved.
```

使用 SQL 语句命令查询 tbccll 表数据（之前已经导入过的, 详情请见实验指导书 02-2）

```
gsql -d postgres -p 26000 -c "select * from tbccll"
```

```
[omm@db1 ~]$ gsql -d postgres -p 26000 -c "select * from tbccll"
city | sector_id | sector_name | wnodebid | enodeb_name | earfcn | pci | psc | ssc | tac | vendor | longitude | latitude | style | azimuth | height | electtilt | mechttilt |
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
bankia | 124672-0 | A池刘果-HLHF-1 | 124672 | A池刘果-HLHF | 38400 | 32 | 2 | 10 | 14531 | 华为 | 112.77068 | 33.810396 | 基站 | 30 | 43 | 6 | 2 |
bankia | 124672-1 | A池刘果-HLHF-2 | 124672 | A池刘果-HLHF | 38400 | 30 | 0 | 10 | 14531 | 华为 | 112.77068 | 33.810396 | 基站 | 150 | 43 | 6 | 2 |
bankia | 124672-2 | A池刘果-HLHF-3 | 124672 | A池刘果-HLHF | 38400 | 31 | 1 | 10 | 14531 | 华为 | 112.77068 | 33.810396 | 基站 | 240 | 43 | 6 | 2 |
bankia | 124673-0 | A池张沟村-HLHF-1 | 124673 | A池张沟村-HLHF | 38400 | 200 | 2 | 66 | 14531 | 华为 | 112.821401 | 33.778498 | 基站 | 40 | 43 | 3 | 0 |
bankia | 124673-1 | A池张沟村-HLHF-2 | 124673 | A池张沟村-HLHF | 38400 | 180 | 0 | 66 | 14531 | 华为 | 112.821401 | 33.778498 | 基站 | 110 | 43 | 3 | 1 |
bankia | 124673-2 | A池张沟村-HLHF-3 | 124673 | A池张沟村-HLHF | 38400 | 199 | 1 | 66 | 14531 | 华为 | 112.821401 | 33.778498 | 基站 | 220 | 43 | 3 | 1 |
bankia | 124674-0 | A池苏门-HLHF-1 | 124674 | A池苏门-HLHF | 38400 | 327 | 0 | 109 | 14531 | 华为 | 112.7852 | 33.807536 | 基站 | 30 | 43 | 3 | 0 |
bankia | 124674-1 | A池苏门-HLHF-2 | 124674 | A池苏门-HLHF | 38400 | 329 | 2 | 109 | 14531 | 华为 | 112.7852 | 33.807536 | 基站 | 180 | 30 | 6 | 3 |
bankia | 124674-2 | A池苏门-HLHF-3 | 124674 | A池苏门-HLHF | 38400 | 320 | 1 | 109 | 14531 | 华为 | 112.7852 | 33.807536 | 基站 | 290 | 30 | 6 | 3 |
bankia | 124675-0 | A池南河-HLHF-1 | 124675 | A池南河-HLHF | 38400 | 94 | 1 | 31 | 14563 | 华为 | 112.822746 | 33.691016 | 基站 | 60 | 43 | 3 | 2 |
bankia | 124675-1 | A池南河-HLHF-2 | 124675 | A池南河-HLHF | 38400 | 93 | 0 | 31 | 14563 | 华为 | 112.822746 | 33.691016 | 基站 | 150 | 30 | 6 | 3 |
bankia | 124675-2 | A池南河-HLHF-3 | 124675 | A池南河-HLHF | 38400 | 95 | 2 | 31 | 14563 | 华为 | 112.822746 | 33.691016 | 基站 | 310 | 43 | 3 | 1 |
bankia | 124676-0 | A池峪源-HLHF-1 | 124676 | A池峪源-HLHF | 38400 | 74 | 2 | 24 | 15833 | 华为 | 112.742883 | 33.694259 | 基站 | 0 | 43 | 3 | 0 |
```

1.3.6 使用文件作为命令源执行命令

创建文件夹存放相关文档。（也可以通过 WinSCP 创建）

```
mkdir /home/omm/openGauss
```

创建文件，例如文件名为“command.sql”，。

```
vi /home/omm/openGauss/command.sql
```

文件打开后输入 i，进入 INSERT 模式，输入” select * from tbccll;”。然后点击 ESC，输入“: wq”保存文档并退出。


```

select * from tbcell;

```

执行如下命令使用文件作为命令源(-f 表明从文件中执行命令)。

```
gsq1 -d postgres -p 26000 -f /home/omm/openGauss/command.sql
```

```

omm@db1 ~]$ mkdir /home/omm/openGauss
omm@db1 ~]$ vi /home/omm/openGauss/command.sql
omm@db1 ~]$ gsql -d postgres -p 26000 -f /home/omm/openGauss/command.sql

```

city	sector_id	sector_name	enodeb_id	enodeb_name	earfcn	pci	psc	ssc	tac	vendor	longitude	latitude	style	azimuth	height	electtilt	mechtlt
nanxia	124672-0	A地刘果-HLRF-1	124672	A地刘果-HLRF	38400	32	2	10	14531	华为	112.77068	33.810396	宏站	30	43	6	2
nanxia	124672-1	A地刘果-HLRF-2	124672	A地刘果-HLRF	38400	30	0	10	14531	华为	112.77068	33.810396	宏站	150	43	6	2
nanxia	124672-2	A地刘果-HLRF-3	124672	A地刘果-HLRF	38400	31	1	10	14531	华为	112.77068	33.810396	宏站	240	43	6	2
nanxia	124673-0	A地张沟村-HLRF-1	124673	A地张沟村-HLRF	38400	200	2	66	14531	华为	112.821401	33.778498	宏站	40	43	3	0
nanxia	124673-1	A地张沟村-HLRF-2	124673	A地张沟村-HLRF	38400	198	0	66	14531	华为	112.821401	33.778498	宏站	110	43	3	1
nanxia	124673-2	A地张沟村-HLRF-3	124673	A地张沟村-HLRF	38400	199	1	66	14531	华为	112.821401	33.778498	宏站	220	43	3	1
nanxia	124674-0	A地苏门-HLRF-1	124674	A地苏门-HLRF	38400	327	0	109	14531	华为	112.7852	33.807536	宏站	30	43	3	0
nanxia	124674-1	A地苏门-HLRF-2	124674	A地苏门-HLRF	38400	329	2	109	14531	华为	112.7852	33.807536	宏站	180	30	6	3

如果 FILENAME 是-（连字符），则从标准输入读取

```
gsq1 -d postgres -p 26000 -f -
```

```

omm@db1 ~]$ gsql -d postgres -p 26000 -f -
postgres=# select * from tbcell;

```

city	sector_id	sector_name	enodeb_id	enodeb_name	earfcn	pci	psc	ssc	tac	vendor	longitude	latitude	style	azimuth	height	electtilt	mechtlt
nanxia	124672-0	A地刘果-HLRF-1	124672	A地刘果-HLRF	38400	32	2	10	14531	华为	112.77068	33.810396	宏站	30	43	6	2
nanxia	124672-1	A地刘果-HLRF-2	124672	A地刘果-HLRF	38400	30	0	10	14531	华为	112.77068	33.810396	宏站	150	43	6	2
nanxia	124672-2	A地刘果-HLRF-3	124672	A地刘果-HLRF	38400	31	1	10	14531	华为	112.77068	33.810396	宏站	240	43	6	2
nanxia	124673-0	A地张沟村-HLRF-1	124673	A地张沟村-HLRF	38400	200	2	66	14531	华为	112.821401	33.778498	宏站	40	43	3	0
nanxia	124673-1	A地张沟村-HLRF-2	124673	A地张沟村-HLRF	38400	198	0	66	14531	华为	112.821401	33.778498	宏站	110	43	3	1
nanxia	124673-2	A地张沟村-HLRF-3	124673	A地张沟村-HLRF	38400	199	1	66	14531	华为	112.821401	33.778498	宏站	220	43	3	1
nanxia	124674-0	A地苏门-HLRF-1	124674	A地苏门-HLRF	38400	327	0	109	14531	华为	112.7852	33.807536	宏站	30	43	3	0
nanxia	124674-1	A地苏门-HLRF-2	124674	A地苏门-HLRF	38400	329	2	109	14531	华为	112.7852	33.807536	宏站	180	30	6	3
nanxia	124674-2	A地苏门-HLRF-3	124674	A地苏门-HLRF	38400	328	1	109	14531	华为	112.7852	33.807536	宏站	290	30	6	3

执行完后使用元命令\q退出

```

postgres=# \q
total time: 63828 ms

```

1.3.7 将输出重定向到文件

创建文件，例如文件名为“outputOnly.txt”。

```
touch /home/omm/openGauss/outputOnly.txt
```

执行如下命令（-o 发送查询结果到指定文件）。

```
gsql -d postgres -p 26000 -o /home/omm/openGauss/outputOnly.txt
```

进入 gsql 环境，输入以下语句：

```
postgres=# select * from tbcell;
```

操作结果并没有显示。

```
omm@db1 ~]# touch /home/omm/openGauss/outputOnly.txt
omm@db1 ~]# gsql -d postgres -p 26000 -o /home/omm/openGauss/outputOnly.txt
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
type "help" for help.

postgres=# select * from tbcell;
postgres=# \q
omm@db1 ~]#
```

查看“outputOnly.txt”文档中的内容如下：

```
cat /home/omm/openGauss/outputOnly.txt
```

查询结果在文档中显示（内容过多，只截图了结尾）：

```
bankia | 7420-129 | D县新寨村-HLHF-2 | 7420 | D县新寨村-HLHF | 38400 | 154 | 1 | 51 | 14387 | 华为 | 112.07011 | 33.68066 | 宏站 | 260 | 11 | 3 | 5 |
bankia | 7421-128 | D县王彦村-HLHF-1 | 7421 | D县王彦村-HLHF | 38400 | 117 | 0 | 39 | 14547 | 华为 | 112.65313 | 33.63333 | 宏站 | 100 | 28 | 3 | 3 |
bankia | 7421-129 | D县王彦村-HLHF-2 | 7421 | D县王彦村-HLHF | 38400 | 118 | 1 | 39 | 14547 | 华为 | 112.65313 | 33.63333 | 宏站 | 180 | 28 | 3 | 3 |
bankia | 7421-130 | D县王彦村-HLHF-3 | 7421 | D县王彦村-HLHF | 38400 | 119 | 2 | 39 | 14547 | 华为 | 112.65313 | 33.63333 | 宏站 | 280 | 28 | 3 | 3 |
bankia | 7422-128 | E宝阳村东村-HLHF-1 | 7422 | E宝阳村东村-HLHF | 38400 | 180 | 0 | 40 | 14832 | 华为 | 111.62265 | 33.36081 | 宏站 | 150 | 16 | 3 | 2 |
bankia | 7422-129 | E宝阳村东村-HLHF-2 | 7422 | E宝阳村东村-HLHF | 38400 | 181 | 1 | 40 | 14832 | 华为 | 111.62265 | 33.36081 | 宏站 | 310 | 16 | 3 | 2 |
bankia | 7423-128 | E宝阳村东村-HLHF-1 | 7423 | E宝阳村东村-HLHF | 38400 | 126 | 0 | 42 | 14355 | 华为 | 111.78915 | 33.51611 | 宏站 | 105 | 28 | 3 | 2 |
bankia | 7423-129 | E宝阳村东村-HLHF-2 | 7423 | E宝阳村东村-HLHF | 38400 | 127 | 1 | 42 | 14355 | 华为 | 111.78915 | 33.51611 | 宏站 | 255 | 28 | 3 | 2 |
(632 rows)
```

也可以通过 WinSCP 查看

/home/omm/openGauss/outputOnly.txt - root@192.168.56.102 - 编辑器 - WinSCP

city	sector_id	sector_name	enodebid	enodeb_name	earfcn	pci	pss	sss	
sanxia	124672-0	A池刘果-HLHF-1	124672	A池刘果-HLHF	38400	32	2	10	14
sanxia	124672-1	A池刘果-HLHF-2	124672	A池刘果-HLHF	38400	30	0	10	14
sanxia	124672-2	A池刘果-HLHF-3	124672	A池刘果-HLHF	38400	31	1	10	14
sanxia	124673-0	A池张沟村-HLHF-1	124673	A池张沟村-HLHF	38400	200	2	66	14
sanxia	124673-1	A池张沟村-HLHF-2	124673	A池张沟村-HLHF	38400	198	0	66	14
sanxia	124673-2	A池张沟村-HLHF-3	124673	A池张沟村-HLHF	38400	199	1	66	14
sanxia	124674-0	A池苏门-HLHF-1	124674	A池苏门-HLHF	38400	327	0	109	14
sanxia	124674-1	A池苏门-HLHF-2	124674	A池苏门-HLHF	38400	329	2	109	14
sanxia	124674-2	A池苏门-HLHF-3	124674	A池苏门-HLHF	38400	328	1	109	14
sanxia	124675-0	A池南涧-HLHF-1	124675	A池南涧-HLHF	38400	94	1	31	14
sanxia	124675-1	A池南涧-HLHF-2	124675	A池南涧-HLHF	38400	93	0	31	14
sanxia	124675-2	A池南涧-HLHF-3	124675	A池南涧-HLHF	38400	95	2	31	14
sanxia	124676-0	A池峪洞-HLHF-1	124676	A池峪洞-HLHF	38400	74	2	24	14
sanxia	124676-1	A池峪洞-HLHF-2	124676	A池峪洞-HLHF	38400	72	0	24	14
sanxia	124676-2	A池峪洞-HLHF-3	124676	A池峪洞-HLHF	38400	73	1	24	14
sanxia	124677-0	A池高岭-HLHF-1	124677	A池高岭-HLHF	38400	400	1	133	14
sanxia	124677-1	A池高岭-HLHF-2	124677	A池高岭-HLHF	38400	401	2	133	14
sanxia	124677-2	A池高岭-HLHF-3	124677	A池高岭-HLHF	38400	399	0	133	14
sanxia	124678-0	A池东坡头-HLHF-1	124678	A池东坡头-HLHF	38400	96	0	32	14
sanxia	124678-1	A池东坡头-HLHF-2	124678	A池东坡头-HLHF	38400	98	2	32	14
sanxia	124678-2	A池东坡头-HLHF-3	124678	A池东坡头-HLHF	38400	97	1	32	14

行: 1/5456 列: 1 字符: 32 (0x20) 编码: UTF-8

1.3.8 发送会话日志到文件

创建文件，例如文件名为“output.txt”。

```
touch /home/omm/openGauss/output.txt
```

执行如下命令，除了正常的输出源之外，把所有查询输出记录到文件中（-L 发送会话日志到指定文件）。

```
gsql -d postgres -p 26000 -L /home/omm/openGauss/output.txt
```

进入 gsql 环境，输入以下语句：

```
postgres=# select * from tbcell;
```

结果正常显示

```
ommdb1 -10 touch /home/omm/openGauss/output.txt
ommdb1 -10 gsql -d postgres -p 26000 -L /home/omm/openGauss/output.txt
gsq1 ((openGauss 1.1.0 build 3920439) compiled at 2020-12-31 20:08:21 commit 0 last m2 )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.
```

city	sector_id	sector_name	enodebid	enodeb_name	earfcn	pci	pss	sss	tac	vendor	longitude	latitude	style	azimuth	height	electtilt	mechtilt
sanxia	124672-0	A池刘果-HLHF-1	124672	A池刘果-HLHF	38400	32	2	10	14531	华为	112.77068	33.810396	基站	30	43	6	2
sanxia	124672-1	A池刘果-HLHF-2	124672	A池刘果-HLHF	38400	30	0	10	14531	华为	112.77068	33.810396	基站	150	43	6	2
sanxia	124672-2	A池刘果-HLHF-3	124672	A池刘果-HLHF	38400	31	1	10	14531	华为	112.77068	33.810396	基站	240	43	6	2
sanxia	124673-0	A池张沟村-HLHF-1	124673	A池张沟村-HLHF	38400	200	2	66	14531	华为	112.821401	33.778498	基站	40	43	3	0
sanxia	124673-1	A池张沟村-HLHF-2	124673	A池张沟村-HLHF	38400	198	0	66	14531	华为	112.821401	33.778498	基站	110	43	3	1
sanxia	124673-2	A池张沟村-HLHF-3	124673	A池张沟村-HLHF	38400	199	1	66	14531	华为	112.821401	33.778498	基站	220	43	3	1

输入\q退出后，查看“output.txt”文档中的内容如下（内容过多，只截图了结尾）：

```
cat /home/omm/openGauss/output.txt
```

sanxia	7420-129	D县新寨西-HLHF-2	7420	D县新寨西-HLHF	38400	154	1	51	14387	华为	112.07011	33.68066	基站	260	11	3	5
sanxia	7421-128	D县王彦村-HLHF-1	7421	D县王彦村-HLHF	38400	117	0	39	14547	华为	112.45313	33.63333	基站	100	28	3	3
sanxia	7421-129	D县王彦村-HLHF-2	7421	D县王彦村-HLHF	38400	118	1	39	14547	华为	112.45313	33.63333	基站	150	28	3	3
sanxia	7421-130	D县王彦村-HLHF-3	7421	D县王彦村-HLHF	38400	119	2	39	14547	华为	112.45313	33.63333	基站	280	28	3	3
sanxia	7422-128	E宝阳村东村-HLHF-1	7422	E宝阳村东村-HLHF	38400	180	0	60	18832	华为	111.62265	33.36081	基站	150	16	3	2
sanxia	7422-129	E宝阳村东村-HLHF-2	7422	E宝阳村东村-HLHF	38400	181	1	60	18832	华为	111.62265	33.36081	基站	310	16	3	2
sanxia	7423-128	E宝集村东村-HLHF-1	7423	E宝集村东村-HLHF	38400	126	0	42	14355	华为	111.78915	33.51611	基站	105	28	3	2
sanxia	7423-129	E宝集村东村-HLHF-2	7423	E宝集村东村-HLHF	38400	127	1	42	14355	华为	111.78915	33.51611	基站	255	28	3	2

(452 rows)

也可以通过 WinSCP 查看

```

/home/omm/openGauss/output.txt - root@192.168.56.102 - 编辑器 - WinSCP
***** QUERY *****
select * from tbccl;
*****

  city | sector_id | sector_name | enodebid | enodeb_name | earfcn | pci | pss | sss |
-----+-----+-----+-----+-----+-----+-----+-----+-----+
sanxia | 124672-0 | A池刘果-HLHF-1 | 124672 | A池刘果-HLHF | 38400 | 32 | 2 | 10 | 1
sanxia | 124672-1 | A池刘果-HLHF-2 | 124672 | A池刘果-HLHF | 38400 | 30 | 0 | 10 | 1
sanxia | 124672-2 | A池刘果-HLHF-3 | 124672 | A池刘果-HLHF | 38400 | 31 | 1 | 10 | 1
sanxia | 124673-0 | A池张沟村-HLHF-1 | 124673 | A池张沟村-HLHF | 38400 | 200 | 2 | 66 | 14
sanxia | 124673-1 | A池张沟村-HLHF-2 | 124673 | A池张沟村-HLHF | 38400 | 198 | 0 | 66 | 14
sanxia | 124673-2 | A池张沟村-HLHF-3 | 124673 | A池张沟村-HLHF | 38400 | 199 | 1 | 66 | 14
sanxia | 124674-0 | A池苏门-HLHF-1 | 124674 | A池苏门-HLHF | 38400 | 327 | 0 | 109 | 14
sanxia | 124674-1 | A池苏门-HLHF-2 | 124674 | A池苏门-HLHF | 38400 | 329 | 2 | 109 | 14
sanxia | 124674-2 | A池苏门-HLHF-3 | 124674 | A池苏门-HLHF | 38400 | 328 | 1 | 109 | 14
sanxia | 124675-0 | A池南涧-HLHF-1 | 124675 | A池南涧-HLHF | 38400 | 94 | 1 | 31 | 14
sanxia | 124675-1 | A池南涧-HLHF-2 | 124675 | A池南涧-HLHF | 38400 | 93 | 0 | 31 | 14

```

-o 与 -l 主要区别：

-o 将操作的结果不显示在 gsql 环境下，而是显示在指定的文件里，病区文件里只显示操作结果，不显示操作命令。

-l 是将会话日志输出到指定的文件里，操作结果仍然正常显示在 gsql 环境下，并且文件里即显示操作结果，也显示操作命令。

1.3.9 列出数据库

```
gsql -d postgres -p 26000 -l
```

显示如下（-l 指--list, 列出所有可有的数据库）：

```

[omm@db1 ~]$ gsql -d postgres -p 26000 -l
List of databases
  Name      | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
postgres   | omm   | UTF8     | C       | C     |
template0  | omm   | UTF8     | C       | C     | =c/omm          +
           |       |          |         |       | omm=CTc/omm
templatel  | omm   | UTF8     | C       | C     | =c/omm          +
           |       |          |         |       | omm=CTc/omm
(3 rows)

```

1.3.10 编辑模式

如下命令连接数据库，开启在客户端操作中可以编辑的模式。

```
gsql -d postgres -p 26000 -r
```

进入 gsql 环境，输入以下语句：

```
postgres=# select * from t_test;
```

写完后不要按回车，光标在最后闪烁。

```
postgres=# select * from t_test;
```

按“向左”键将光标移动到“*”，将“*”修改为“firstcol”。

```
postgres=# select firstcol from t_test;
 firstcol
-----
      200
(1 row)

postgres=#
```

编辑模式“上下左右键”，并且按下“向上”、“向下”键可以切换输入过的命令（其他模式的上下左右键为输入相应的字符）。

1.3.11 安静模式

进入安静模式（-q 只显示查询结果，不显示其他额外信息）

```
gsql -d postgres -p 26000 -q
```

进入 gsql 环境，输入以下语句：

```
postgres=# create table t_test (firstcol int);
postgres=# select * from t_test;
postgres=# drop table t_test;
```

-r 编辑模式下：

```
[omm@db1 ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# create table t_test (firstcol int);
CREATE TABLE
postgres=# select * from t_test;
 firstcol
-----
(0 rows)

postgres=# drop table t_test;
DROP TABLE
```

-q 安静模式下：

```
[omm@db1 ~]$ gsql -d postgres -p 26000 -q
postgres=# create table t_test (firstcol int);
postgres=# select * from t_test;
 firstcol
-----
(0 rows)

postgres=# drop table t_test;
```

1.3.12 单行运行模式

进入单行运行模式（-S 每个命令都将由换行符结束，像分号那样）

```
gsql -d postgres -p 26000 -S
```

进入 gsql 环境，输入以下语句，其中每条语句只用按回车结束即可执行，无需输入分号；

```
postgres=# create table t_test (firstcol int)
postgres=# select * from t_test
postgres=# drop table t_test
```

-r 编辑模式下：

```
postgres=# gsql -d postgres -p 26000 -r
postgres=# \q
[omm@db1 ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# create table t_test (firstcol int)
postgres=# select * from t_test
postgres=# drop table t_test
postgres=#
```

-S 单行运行模式下：

```
[omm@db1 ~]$ gsql -d postgres -p 26000 -S
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres^# create table t_test (firstcol int)
CREATE TABLE
postgres^# select * from t_test
 firstcol
-----
(0 rows)

postgres^# drop table t_test
DROP TABLE
```

1.3.13 打印 gsql 版本信息

-V 输出版本信息

```
gsql -V
```

结果如下，并且 gsql 将在显示后结束：

```
[omm@db1 ~]$ gsql -V
gsql (openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr
```

1.3.14 帮助信息

更多的 gsql 命令可以通过帮助信息获取

```
gsql --help 或者 gsql -?
```

```
[omm@db1 ~]$ gsql -?
gsql is the FusionInsight LibrA interactive terminal.

Usage:
  gsql [OPTION]... [DBNAME [USERNAME]]

General options:
  -c, --command=COMMAND      run only single command (SQL or internal) and exit
  -d, --dbname=DBNAME        database name to connect to (default: "omm")
  -f, --file=FILENAME        execute commands from file, then exit
  -l, --list                  list available databases, then exit
  -v, --set=, --variable=NAME=VALUE
                              set gsql variable NAME to VALUE
  -V, --version               output version information, then exit
  -X, --no-gsqlrc             do not read startup file (~/.gsqlrc)
  -l ("one"), --single-transaction
                              execute command file as a single transaction
  -?, --help                  show this help, then exit

Input and output options:
  -a, --echo-all             echo all input from script
  -e, --echo-queries          echo commands sent to server
  -E, --echo-hidden           display queries that internal commands generate
  -k, --with-key=KEY          the key for decrypting the encrypted file
  -L, --log-file=FILENAME     send session log to file
  -m, --maintenance          can connect to cluster during 2-pc transaction recovery
  -n, --no-libedit            disable enhanced command line editing (libedit)
  -o, --output=FILENAME       send query results to file (or |pipe)
  -q, --quiet                 run quietly (no messages, only query output)
  -C, --enable-client-encryption
                              enable client encryption feature
  -s, --single-step           single-step mode (confirm each query)
  -S, --single-line           single-line mode (end of line terminates SQL command)
```

更多的 `gs_om` 命令可以通过帮助信息获取

```
gs_om --help 或者 gs_om -?
```

```
[omm@db1 ~]$ gs_om --help

gs_om is a utility to manage a cluster.

Usage:
  gs_om -? | --help
  gs_om -V | --version
OLAP scene:
  gs_om -t start [-h HOSTNAME] [-D dataDir] [--time-out=SECS]
                  [--security-mode=MODE] [-l LOGFILE]
  gs_om -t stop [-h HOSTNAME] [-D dataDir] [--time-out=SECS] [-m MODE]
                  [-l LOGFILE]
  gs_om -t restart [-h HOSTNAME] [-D dataDir] [--time-out=SECS]
                  [--security-mode=MODE] [-l LOGFILE] [-m MODE]
  gs_om -t status [-h HOSTNAME] [-o OUTPUT] [--detail] [--all] [-l LOGFILE]
  gs_om -t generateconf -X XMLFILE [--distribute] [-l LOGFILE]
  gs_om -t cert [--cert-file=CERTFILE | --rollback] [-L] [-l LOGFILE]
  gs_om -t kerberos -m [install|uninstall] -U USER [-l LOGFILE]
                  [--krb-server|--krb-client]
  gs_om -t view [-o OUTPUT]
  gs_om -t query [-o OUTPUT]
  gs_om -t refreshconf

General options:
  -t                  Type of the OM command.
  -l                  Path of log file.
  -?, --help          Show help information for this utility,
and exit the command line mode.
  -V, --version       Show version information.
```


1.3.14 自动断开连接

缺省情况下，客户端连接数据库后处于空闲状态时会根据参数 `session_timeout` 的默认值自动断开连接。如果要关闭超时设置，设置参数 `session_timeout` 为 0 即可。

首先，连接数据库并查看 `session_timeout` 值

```
show session_timeout;
```

表示空闲状态达 10 分钟时会自动断开连接

```
postgres=# show session_timeout;
 session_timeout
-----
      10min
(1 row)
```

输入 \q 退出后，更改参数 `session_timeout`

```
gs_guc reload -N all -I all -c "session_timeout=0"
```

更改成功，这样数据库不会因为空闲状态超时而自动断开连接了

```
[omm@dml ~]$ gs_guc reload -N all -I all -c "session_timeout=0"
NOTICE: GaussDB Kernel gsql client has an automatic reconnection mechanism, when the timeout, the gsql will be reconnection after disconnection.
Begin to perform the total nodes: 1.
Popen count is 1, Popen success count is 1, Popen failure count is 0.
Begin to perform gs_guc for datanodes.
Command count is 1, Command success count is 1, Command failure count is 0.

Total instances: 1. Failed instances: 0.
ALL: Success to perform gs_guc!
```

再连接数据库，查看其值

```
postgres=# show session_timeout;
 session_timeout
-----
      0
(1 row)
```

注意：在超时自动断开连接后再输入命令，就会显示

```
postgres=# select * from tbcell;
WARNING: Session unused timeout.
FATAL: terminating connection due to administrator command
could not send data to server: Broken pipe
The connection to the server was lost. Attempting reset: Succeeded.
postgres=#
```

此时需要 \q 退出后，再重新连接数据库

```
\q
```

```
gsql -d postgres -p 26000 -r
```

1.3.15 关闭数据库服务

```
gs_om -t stop
```

关闭成功


```
[omm@db1 ~]$ gs_om -t stop
Stopping cluster.
=====
Successfully stopped cluster.
=====
End stop cluster.
```

2 openGauss 数据库基本操作

2.1 实验介绍

关于本实验

本实验主要描述在连接 openGauss 数据库进入编辑模式下的操作命令

实验目的

- 1.了解 postgresQL 的元命令。
- 2.掌握 postgres 支持的 SQL 语句。

2.2 postgresQL 简介

PostgreSQL 是一个功能非常强大的、源代码开放的客户/服务器关系型数据库管理系统,它支持大部分的 SQL 标准并且提供了很多其他现代特性,如复杂查询、外键、触发器、视图、事务完整性、多版本并发控制等。同样,PostgreSQL 也可以用许多方法扩展,例如通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等。另外,因为许可证的灵活,任何人都可以以任何目的免费使用、修改和分发 PostgreSQL。而 openGauss 便是基于 PostgreSQL9.2.4 的内核开发的,接下来将从其支持的 SQL 语言和内部的元命令两方面来介绍数据库的基本操作。

2.3 元命令

postgresql 中的元命令是指以 \ (反斜线)开头的命令,通过使用这些丰富的元命令,能够便捷地管理数据库。注意:元命令结尾不用加分号; ,直接按回车便可以执行。

先登录 omm 用户,连接数据库进入编辑模式

```
su - omm
gs_om -t start
gsql -d postgres -p 26000 -r
```

2.3.1 \d

- 1、不加分任何参数表示查看当前数据库的所有表,视图,序列等。
- 2、\d tablename 后面跟一个表名,表示显示这个表的结构定义
- 3、\d indexname 后面跟一个索引名,也可以显示这个索引的信息
- 4、\dt 只显示匹配的表
- 5、\di 只显示索引
- 6、\ds 只显示序列
- 7、\dv 只显示视图
- 8、\df 只显示函数

9、\dn 列出所有的 schema

10、\du 或 \dg 列出所有的数据库用户和角色

11、\db 显示所有的表空间，表空间其实是一个目录，放在这个表空间的表，就是把表的数据文件发到这个表空间下。

12、\dp 或 \z 显示表的权限分配情况

13、在以上命令后面多一个+，将显示比原来更详细的信息，比如：\d+,\di+,\d+ tablename 等

```
postgres=# \d
          List of relations
Schema |      Name      | Type | Owner |      Storage
-----+-----+-----+-----+-----
public | tbadjcell       | table | omm    | {orientation=row,compression=no}
public | tbatuc2i        | table | omm    | {orientation=row,compression=no}
public | tbatudata       | table | omm    | {orientation=row,compression=no}
public | tbatuhandover   | table | omm    | {orientation=row,compression=no}
public | tbc2i           | table | omm    | {orientation=row,compression=no}
public | tbcell          | table | omm    | {orientation=row,compression=no}
public | tbcellkpi       | table | omm    | {orientation=row,compression=no}
public | tbcelltraffic   | table | omm    | {orientation=row,compression=no}
public | tbhandover      | table | omm    | {orientation=row,compression=no}
public | tbmrodata       | table | omm    | {orientation=row,compression=no}
public | tboptcell       | table | omm    | {orientation=row,compression=no}
public | tbpciassignment | table | omm    | {orientation=row,compression=no}
public | tbprb           | table | omm    | {orientation=row,compression=no}
public | tbsecadjcell    | table | omm    | {orientation=row,compression=no}
(14 rows)
```

```
postgres=# \d tbcell
```

Table "public.tbcell"		
Column	Type	Modifiers
city	character varying(255)	default NULL::character varying
sector_id	character varying(50)	not null
sector_name	character varying(255)	not null
enodebid	integer	not null
enodeb_name	character varying(255)	not null
earfcn	integer	not null
pci	integer	
pss	integer	
sss	integer	
tac	integer	
vendor	character varying(255)	
longitude	double precision	not null
latitude	double precision	not null
style	character varying(255)	
azimuth	double precision	not null
height	double precision	
electtilt	double precision	
mechtilt	double precision	
totletilt	double precision	not null

Indexes:

```
"tbcell_pkey" PRIMARY KEY, btree (sector_id) TABLESPACE pg_default
```

Check constraints:

```
"tbcell_check" CHECK (pci = (3 * sss + pss))
```

```
"tbcell_check1" CHECK (totletilt = (electtilt + mechtilt))
```

```
"tbcell_pci_check" CHECK (pci >= 0 AND pci <= 503)
```

```
"tbcell_pss_check" CHECK (pss >= 0 AND pss <= 2)
```

```
"tbcell_sss_check" CHECK (sss >= 0 AND sss <= 167)
```

```
postgres=# \di
```

List of relations					
Schema	Name	Type	Owner	Table	Storage
public	tbadjcell_pkey	index	omm	tbadjcell	
public	tbatuc2i_pkey	index	omm	tbatuc2i	
public	tbatudata_pkey	index	omm	tbatudata	
public	tbc2i_pkey	index	omm	tbc2i	
public	tbcell_pkey	index	omm	tbcell	
public	tbcellkpi_pkey	index	omm	tbcellkpi	
public	tbcelltraffic_pkey	index	omm	tbcelltraffic	
public	tbhandover_pkey	index	omm	tbhandover	
public	tboptcell_pkey	index	omm	tboptcell	
public	tbpciassignment_pkey	index	omm	tbpciassignment	
public	tbprb_pkey	index	omm	tbprb	
public	tbsecadjcell_pkey	index	omm	tbsecadjcell	

(12 rows)

```
postgres=# \d tbcell_pkey
          Index "public.tbcell_pkey"
   Column |          Type          | Definition
-----+-----+-----
 sector_id | character varying(50) | sector_id
primary key, btree, for table "public.tbcell"
```

2.3.2 其它

- 1、\c dbname 切换数据库
- 2、\c - username 切换用户
- 3、\l 查看数据库系统的数据库列表
- 4、\copyright 查看版权信息
- 5、\copy from/to 从文件导入数据到关系表或从关系表导出数据到文件，详情请见实验指导书 02-2
- 6、\w filename 打印当前查询缓冲区到标准输出
- 7、\pset 设置输出的格式，\pset border 0 表示输出内容无边框。border 1 表示边框只在内部。border 2 表示内外都有边框
- 8、\x 把表中的每一行的每列数据都拆分为单行展示，与 MySQL 中的 “\G” 的功能类似。
- 9、\echo 用于输出一行信息，通常用于在 .sql 文件中输出一些提示信息。
- 10、\password 设置密码
- 11、\conninfo 列出当前数据库连接的信息
- 12、\dx 查看数据库中安装的扩展 或 select * from pg_extension;
- 13、\encoding 指定客户端的字符编码，如 \encoding UTF8;
- 14、\h 或者 \help 查看 openGauss 支持的所有 SQL 语句, 若在后面加上具体的 SQL 语句，可以查看具体 SQL 语句的帮助信息，比如：\h SELECT
- 15、\? 查看帮助信息，可以再次查看更多的元命令
- 16、\q 退出 gsql

实例：

\w:

创建 “outputSQL.txt” 文件。

```
touch /home/omm/openGauss/outputSQL.txt
```

连接数据库。

```
gsql -d postgres -p 26000 -r
```

输入以下查询语句。

```
postgres=# select * from tbcell;
```

将查询结果输出到 “outputSQL.txt” 文件

```
postgres=# \w /home/omm/openGauss/outputSQL.txt
```

\copy:

创建目标表 i。

```
postgres=# CREATE TABLE i(i int);
```

导入数据，从 stdin 拷贝数据到目标表 a。

```
postgres=# \copy i from stdin;
```

出现>>符号提示时，输入数据，输入\.时结束。

```
postgres=# create table i(i int);
CREATE TABLE
postgres=# \copy i from stdin
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself.
>> 1
>> 2
>> \.
postgres=# select * from i;
 i
---
 1
 2
(2 rows)
```

2.4 SQL 语句

openGauss 支持 SQL2003 标准语法，能够执行大部分常用的 SQL 语句。注意：在编辑模式下，SQL 语句需要在结尾加上分号；，再按回车键才能执行。

2.4.1 创建管理用户

通过 CREATE USER 创建的用户，默认具有 LOGIN 权限；

通过 CREATE USER 创建用户的同时系统会在执行该命令的数据库中，为该用户创建一个同名的 SCHEMA；其他数据库中，则不自动创建同名的 SCHEMA；用户可使用 CREATE SCHEMA 命令，分别在其他数据库中，为该用户创建同名 SCHEMA。

系统管理员在普通用户同名 schema 下创建的对象，所有者为 schema 的同名用户（非系统管理员）。

注意：CREATE ROLE 就不会创建同名 SCHEMA

创建用户 jack，登录密码为 Jack1234(密码长度至少为 8, 类型至少为 3 类)。

```
postgres=# CREATE USER jack PASSWORD 'Jack1234';
CREATE ROLE
```

同样的下面语句也可以创建用户。

```
postgres=# CREATE USER jack IDENTIFIED BY 'Jack1234';
CREATE ROLE
```

如果创建有“创建数据库”权限的用户，则需要加 CREATEDB 关键字。

```
postgres=# CREATE USER jack CREATEDB PASSWORD 'Jack1234';
CREATE ROLE
```

将用户 jack 的登录密码由 Jack1234 修改为 Jack12345。注意：当安装完 openGauss 数据库，第一次用用户 omm 连接数据库后，需要先修改密码，否则无法执行 SQL 语句。

```
postgres=# ALTER USER jack IDENTIFIED BY 'Jack12345' REPLACE 'Jack1234';
ALTER ROLE
```

为用户 jack 追加 CREATEROLE 权限。

```
postgres=# ALTER USER jack CREATEROLE;
ALTER ROLE
```

将 enable_seqscan 的值设置为 on，设置成功后，在下一会话中生效。

```
postgres=# ALTER USER jack SET enable_seqscan TO on;  
ALTER ROLE
```

锁定 jack 帐户。

```
postgres=# ALTER USER jack ACCOUNT LOCK;  
ALTER ROLE
```

解锁 jack 用户。

```
postgres=# ALTER USER jack ACCOUNT UNLOCK;  
ALTER ROLE
```

查看数据库用户列表

```
postgres=# SELECT * FROM pg_user;
```

要查看用户属性

```
postgres=# SELECT * FROM pg_authid;
```

查看所有角色

```
postgres=# SELECT * FROM PG_ROLES;
```

删除用户 (如果该用户上拥有数据库，需先删除该数据库或者把该数据库转移给其他用户后，才能删除该用户)。因为创建用户时，创建了一个同名的 schema，所以删除时需加上 CASCADE

```
postgres=# DROP USER jack CASCADE;
```

以上语句中的 USER 都可以换成 ROLE，但是由于 ROLE 没有创建同名的 schema，则删除时不需加上 CASCADE

2.4.2 创建管理数据库

使用如下命令创建一个新的数据库 db_new。可以用 OWNER 指定数据库的拥有者，如果不指定，则默认创建该数据库的当前用户为其拥有者。创建数据库时也可以指定数据库的表空间等属性，可以通过 \h CREATE DATABASE 查看创建数据库时的更多设置。

```
postgres=# CREATE DATABASE db_new OWNER omm;  
CREATE DATABASE
```

使用如下命令通过系统表 pg_database 查询数据库列表 (SELECT 后面加上 oid，可以查看对应的 oid。)

```
postgres=# SELECT datname FROM pg_database;
```

用户可以使用如下命令修改数据库属性 (比如：owner、名称和默认的配置属性)。

使用以下命令为数据库更改拥有者。

```
postgres=# ALTER DATABASE db_new OWNER TO jack;  
ALTER DATABASE
```

使用如下命令为数据库重新命名。

```
postgres=# ALTER DATABASE db_new RENAME TO db_0;  
ALTER DATABASE
```

用户可以使用 DROP DATABASE 命令删除数据库。此命令删除了数据库中的系统目录，并且删除了带有数据的磁盘上的数据库目录。用户必须是数据库的 owner 或者系统管理员才能删除数据库。当有人连接数据库时，删除操作会失败。删除数据库时请先连接到其他的数据库。

使用如下命令删除数据库：

```
postgres=# DROP DATABASE db_0;
DROP DATABASE
```

2.4.3 管理表

表是建立在数据库中的，在不同的数据库中 can 存放相同的表。甚至可以通过使用模式在同一个数据库中创建相同名称的表。

执行如下命令创建表。

```
postgres=# CREATE TABLE customer_t1
(
c_customer_sk integer default null,
c_customer_id varchar(50) default null,
c_first_name varchar(50) default null,
c_last_name varchar(50) default null
);
```

当结果显示为如下信息，则表示创建成功。

```
CREATE TABLE
```

其中 `c_customer_sk`、`c_customer_id`、`c_first_name` 和 `c_last_name` 是表的字段名（字段名可以有中文，英文，也可以有下划线和数字，但是不能其它符号，并且数字不能作为开头），`integer` 和 `varchar(50)` 分别是这四字段名称的类型（数据类型的介绍在实验指导书 02-2 中）。

向表中插入数据：

数据值是按照这些字段在表中出现的顺序列出的，并且用逗号分隔。通常数据值是文本（常量），但也允许使用标量表达式。

```
postgres=# INSERT INTO customer_t1(c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', 'Grace');
```

如果用户已经知道表中字段的顺序，也可无需列出表中的字段。例如以下命令与上面的命令效果相同。

```
postgres=# INSERT INTO customer_t1 VALUES (3769, 'hello', 'Grace');
```

如果用户不知道所有字段的数值，可以忽略其中的一些。没有数值的字段将被填充为字段的缺省值。例如：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_first_name) VALUES (3769, 'Grace');
```

用户也可以对独立的字段或者整个行明确缺省值：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', DEFAULT);
```

或

```
postgres=# INSERT INTO customer_t1 DEFAULT VALUES;
```

向表中插入多行数据，命令如下：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES
(9527, 'maps', 'Joes'),
(4321, 'tpcds', 'Lily'),
(9527, 'world', 'James');
```

如果需要向表中插入多条数据，除此命令外，也可以多次执行插入一行数据命令实现。但是建议使用此命令可以提升效率。

如果从指定表插入数据到当前表，例如在数据库中创建了一个表 `customer_t1` 的备份表 `customer_t2`，现在需要将表 `customer_t1` 中的数据插入到表 `customer_t2` 中，则可以执行如下命令。

```
postgres=# CREATE TABLE customer_t2
```



```
(  
c_customer_sk integer default null,  
c_customer_id varchar(50) default null,  
c_first_name varchar(50) default null,  
c_last_name varchar(50) default null  
);
```

插入数据:

```
INSERT INTO customer_t2 SELECT * FROM customer_t1;
```

更新表中数据:

修改已经存储在数据库中数据的行为叫做更新。用户可以更新单独一行，所有行或者指定的部分行。还可以独立更新每个字段，而其他字段则不受影响。

需要将表 customer_t1 中 c_customer_sk 为 9527 的字段重新定义为 9876:

```
postgres=# UPDATE customer_t1 SET c_customer_sk = 9876 WHERE c_customer_sk = 9527;
```

这里的表名称也可以使用模式名修饰，否则会从默认的模式路径找到这个表。SET 后面紧跟字段和新的字段值。新的字段值不仅可以是常量，也可以是变量表达式。

比如，把所有 c_customer_sk 的值增加 100:

```
postgres=# UPDATE customer_t1 SET c_customer_sk = c_customer_sk + 100;
```

用户可以在一个 UPDATE 命令中更新更多的字段，方法是在 SET 子句中列出更多赋值，比如:

```
postgres=# UPDATE customer_t1 SET c_customer_id = 'Admin', c_first_name = 'Local' WHERE c_customer_sk = 4321;
```

查看表中数据:

执行如下命令查询表 customer_t1 的数据量。

```
postgres=# SELECT count(*) FROM customer_t1;
```

执行如下命令查询表 customer_t1 的所有数据。

```
postgres=# SELECT * FROM customer_t1;
```

执行如下命令只查询字段 c_customer_sk 的数据。

```
postgres=# SELECT c_customer_sk FROM customer_t1;
```

执行如下命令过滤字段 c_customer_sk 的重复数据。

```
postgres=# SELECT DISTINCT( c_customer_sk ) FROM customer_t1;
```

执行如下命令查询字段 c_customer_sk 为 3869 的所有数据。

```
postgres=# SELECT * FROM customer_t1 WHERE c_customer_sk = 3769;
```

执行如下命令按照字段 c_customer_sk 进行排序。

```
postgres=# SELECT * FROM customer_t1 ORDER BY c_customer_sk;
```

执行如下命令查询 ROWNUM 伪列。

```
postgres=# SELECT rownum, c_customer_sk, c_customer_id FROM customer_t1;
```

执行如下命令使用别名进行查询(CNB、CSK、CID 为列别名，T 为表别名)。

```
postgres=# SELECT rownum CNB, T.c_customer_sk CSK, T.c_customer_id CID FROM customer_t1 T;
```

删除表中数据:

在使用表的过程中，可能会需要删除已过期的数据，删除数据必须从表中整行的删除。

使用 DELETE 命令删除行，如果删除表 customer_t1 中所有 c_customer_sk 为 3869 的记录：

```
postgres=# DELETE FROM customer_t1 WHERE c_customer_sk = 3869;
```

如果执行如下命令之一，会删除表中所有的行。

```
postgres=# DELETE FROM customer_t1;
```

或：

```
postgres=# TRUNCATE TABLE customer_t1;
```

全表删除的场景下，建议使用 truncate，不建议使用 delete。

删除创建的表：

```
postgres=# DROP TABLE customer_t1;
postgres=# DROP TABLE customer_t2;
```

2.4.4 创建管理 schema

执行如下命令来创建一个 schema。

```
postgres=# CREATE SCHEMA myschema;
```

当结果显示为如下信息，则表示成功创建一个名为 myschema 的 schema。

```
CREATE SCHEMA
```

执行如下命令在创建 schema 时指定 owner。

```
postgres=# CREATE SCHEMA myschema AUTHORIZATION omm;
```

当结果显示为如下信息，则表示成功创建一个属于 omm 用户，名为 myschema 的 schema。

```
CREATE SCHEMA
```

在特定 schema 下创建对象或者访问特定 schema 下的对象，需要使用有 schema 修饰的对象名。该名称包含 schema 名以及对象名，他们之间用 “.” 号分开。

执行如下命令在 myschema 下创建 mytable 表。

```
postgres=# CREATE TABLE myschema.mytable(id int, name varchar(20));
CREATE TABLE
```

执行如下命令查询 myschema 下 mytable 表的所有数据。

```
postgres=# SELECT * FROM myschema.mytable;
```

可以设置 search_path 配置参数指定寻找对象可用 schema 的顺序。在搜索路径列出的第一个 schema 会变成默认的 schema。如果在创建对象时不指定 schema，则会创建在默认的 schema 中。

执行如下命令查看搜索路径。

```
postgres=# SHOW SEARCH_PATH;
search_path
-----
"$user",public
(1 row)
```

执行如下命令将搜索路径设置为 myschema、public，首先搜索 myschema。

```
postgres=# SET SEARCH_PATH TO myschema,public;
SET
```

默认情况下，用户只能访问属于自己的 schema 中的数据库对象。如果需要访问其他 schema 的对象，则该 schema 的所有者应该赋予他对该 schema 的 usage 权限。

通过将模式的 CREATE 权限授予某用户，被授权用户就可以在此模式中创建对象。注意默认情况下，所有角色都拥有在 public 模式上的 USAGE 权限，但是普通用户没有在 public 模式上的 CREATE 权限。普通用户能够连接到一个指定数据库并在它的 public 模式中创建对象是不安全的，如果普通用户具有在 public 模式上的 CREATE 权限，则建议通过如下语句撤销该权限。

假如 Jack 拥有在 public 模式上的 CREATE 权限，则撤销 Jack 在 public 模式下创建对象的权限

```
postgres=# REVOKE CREATE ON SCHEMA public FROM Jack;  
REVOKE
```

使用以下命令查看现有的 schema:

```
postgres=# SELECT current_schema();
```

将 myschema 的 usage 权限赋给用户 jack。

```
postgres=# GRANT USAGE ON schema myschema TO jack;  
GRANT
```

将用户 jack 对于 myschema 的 usage 权限收回。

```
postgres=# REVOKE USAGE ON schema myschema FROM jack;  
REVOKE
```

当 schema 为空时，即该 schema 下没有数据库对象，使用 DROP SCHEMA 命令进行删除。例如删除名为 nullschema 的空 schema。

```
postgres=# DROP SCHEMA IF EXISTS nullschema;
```

当 schema 非空时，如果要删除一个 schema 及其包含的所有对象，需要使用 CASCADE 关键字。例如删除 myschema 及该 schema 下的所有对象。

```
postgres=# DROP SCHEMA myschema CASCADE;  
DROP SCHEMA
```

2.4.5 创建管理视图

执行如下命令创建普通视图 MyView。

```
postgres=# CREATE OR REPLACE VIEW MyView AS SELECT * FROM tbcell WHERE PSS=1;
```

执行如下命令创建物化视图 MV_MyView。

```
postgres=# CREATE MATERIALIZED VIEW MV_MyView AS SELECT * FROM tbcell WHERE PSS=1;
```

物化视图使用场景：报表统计、大表统计等，定期固化数据快照，避免对多表重复跑相同的查询。

物化视图使用注意事项：

不可在临时表或全局临时表上创建。

当基表数据发生变化时，需要使用刷新命令保持物化视图与基表同步。

可以执行如下命令刷新物化视图 MV_MyView。

```
postgres=# REFRESH MATERIALIZED VIEW MV_MyView;
```

执行如下命令查询 MyView 视图。

```
postgres=# SELECT * FROM MyView;
```

执行如下命令查询 MyView 视图的详细信息。

```
postgres=# \d+ Myview
```

执行如下命令查询 MV_MyView 视图。

```
postgres=# SELECT * FROM MV_MyView;
```

执行如下命令查询 MyView 视图的详细信息。

```
postgres=# \d+ MV_MyView
```

执行如下命令删除视图。

```
postgres=# DROP VIEW MyView;
DROP VIEW
postgres=# DROP MATERIALIZED VIEW MV_MyView;
DROP MATERIALIZED VIEW
```

2.4.6 创建管理序列

用法一：声明字段类型为序列整型(serial)来定义标识符字段

```
postgres=# CREATE TABLE T1
(id serial,
name text
);
```

当结果显示为如下信息，则表示创建成功。

```
CREATE TABLE
```

用法二：创建序列，并通过 nextval('sequence_name') 函数指定为某一字段的默认值

创建序列。

```
postgres=# CREATE SEQUENCE seq1 cache 100;
```

结果显示为如下信息，则表示创建成功。

```
CREATE SEQUENCE
```

指定为某一字段的默认值，使该字段具有唯一标识属性。

```
postgres=# CREATE TABLE T2
(
id int not null default nextval('seq1'),
name text
);
```

当结果显示为如下信息，则表示默认值指定成功。

```
CREATE TABLE
```

指定序列与列的归属关系

将序列和一个表的指定字段进行关联。删除此字段或其所在表的时候会自动删除已关联的序列。

```
postgres=# ALTER SEQUENCE seq1 OWNED BY T2.id;
```

当结果显示为如下信息，则表示指定成功。

```
ALTER SEQUENCE
```

删除序列

```
DROP SEQUENCE seq1 CASCADE;
```

2.4.7 创建管理存储过程

创建表 t_test。

```
postgres=# create table t_test(c1 int, c2 int);
```

创建存储过程 insert_data。

```
postgres=# create or replace procedure insert_data
is
a int;
b int;
begin
a=1;
b=2;
insert into t_test values(a,b);
insert into t_test values(b,a);
end;
/
```

调用存储过程。

```
call insert_data();
```

查询表内容。

```
postgres=# select * from t_test;
 c1 | c2
----+----
  1 |  2
  2 |  1
(2 rows)
```

管理存储过程，命令如下：

```
postgres=# \sf insert_data
```

结果如下：

```
postgres=# \sf insert_data
CREATE OR REPLACE PROCEDURE public.insert_data()
AS DECLARE
a int;
b int;
begin
a=1;
b=2;
insert into t_test values(a,b);
insert into t_test values(b,a);
end;
/
```

删除存储过程，命令如下：

```
drop procedure insert_data;
```

2.4.8 创建管理全局临时表

会话级全局临时表

数据会话级可见，其他会话看不到数据，但表结构可见。

创建临时表 t_test1。

建表语句，使用 ON COMMIT PRESERVE ROWS

```
postgres=# CREATE GLOBAL TEMPORARY TABLE t_test1(
id integer,
lbl text
) ON COMMIT PRESERVE ROWS;
```

成功返回如下：

```
CREATE TABLE
```

在当前会话插入数据并查询。

```
postgres=# insert into t_test1 values(1,'data1');
INSERT 0 1
postgres=# insert into t_test1 values(2,'data2');
INSERT 0 1
postgres=# select * from t_test1;
 id | lbl
----+----
  1 | data1
  2 | data2
(2 rows)
```

退出会话再查看。

```
postgres=# \q
[omm@ecs-32de ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 13b34b53) compiled at 2021-03-17 12:01:33 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# \dt t_test1

                                List of relations
 Schema | Name   | Type  | Owner | Storage
-----+-----+-----+-----+-----
 public | t_test1 | table | omm   | {orientation=row,compression=no,on_commit_delete_rows=false}
(1 row)
```

查询表内容。

```
postgres=# select * from t_test1;
 id | lbl
----+----
(0 rows)
```

此时可以发现，在其它会话中表结构可以看到，但是表数据看不到。

删除临时表。

```
postgres=# drop table t_test1;  
DROP TABLE
```

事务级全局临时表

数据事务级可见，事务提交后数据删除。

创建临时表 t_test2。

建表语句，使用 ON COMMIT DELETE ROWS

```
postgres=# CREATE GLOBAL TEMPORARY TABLE t_test2(  
id integer,  
lbl text  
) ON COMMIT DELETE ROWS;  
CREATE TABLE
```

插入数据并查询。

先用 begin 开始一个事务，接着给表插入数据，此时再对表进行查询，可以查出相应数据。

```
postgres=# begin;  
BEGIN  
postgres=# insert into t_test2 values(1,'data1');  
INSERT 0 1  
postgres=# select * from t_test2;  
 id | lbl  
----+-----  
  1 | data1  
(1 row)
```

结束事务再查询。

先用 commit 提交来结束事务，此时再对表进行查询，可以发现已经查询不出数据了。

```
postgres=# commit;  
COMMIT  
postgres=# select * from t_test2;  
 id | lbl  
----+-----  
(0 rows) (1 row)
```

删除临时表。

```
postgres=# drop table t_test2;  
DROP TABLE
```

此外，还有表空间，分区表和索引的创建与管理，它们的详细介绍在实验指导书 06-2 上。