1. 官方文档网站。

   Brief Tutorial (opengauss.org)

2. 修改数据库参数（三种方式）。
   - 直接修改 postgres.conf 文件，修改后需重启数据库实例，如涉及主备，主备都需修改。（如果查找不到参数名，可以通过数据字典表 `pg_setting` 查询）
   - 在数据库中使用 `ALTER SYSTEM SET` 修改。
   - gs_guc工具，例：

     ```
     $ gs_guc set -N all -I all -D /gaussdb/data/dbnode -c
     "password_encryption_type = 1"
     $ gs_om -t restart
     ```

3. 连接认证文件 pg_hba.conf（两种方式）。
   - 直接编辑修改文件，修改后重启数据库生效（或通过 `gs_ctl reload` 重新加载）。
   - gs_guc工具，例：

     ```
     $ gs_guc reload -h "host postgres user01 192.168.56.1/32 sha256"
     $ gs_om -t restart
     ```

4. 数据字典表。

名称	说明
pg_setting	可用于查询参数。
pg_proc	可用于查询函数。
pg_tables	可用于查询表名。
gs_recyclebin	可用于查询回收站对象。
gs_txn_snapshot	可用于查询 csn。

5. gsql 命令帮助的使用。

命令	说明	举例
\h	查询 sql 帮助。	\h CREATE SCHEMA
\?	查询命令帮助。	\?

6. 数据导入导出。
   - 使用元命令。
   - 使用工具 gs_dump，gs_restore

```
# 步骤一：创建测试数据。
$ gsql -p 15432 -d postgres -c "create table mytb01(id number);insert
into mytb01 values(1);"
$ gsql -p 15432 -d postgres -c "create table mytb02(id number);insert
into mytb02 values(1),(2);"
$ gsql -p 15432 -d postgres -c "create table mytb03(id number);insert
into mytb03 values(1),(2),(3);"
# 步骤二：导出数据。
$ gs_dump -p 15432 -n public -f /home/omm/schema_public.sql -F c
postgres
# 步骤三：删除数据。
$ gsql -p 15432 -d postgres -c "drop table mytb01;drop table
mytb02;\dt;"
# 步骤四：导入数据。
$ gs_restore -d postgres -p 15432 -t mytb02 -F c
/home/omm/schema_public.sql
$ gsql -p 15432 -d postgres -c "\d mytb02;select * from mytb02;"
```

7. 备份恢复（使用工具 pg_proback）。

```
# 步骤一：设置参数
$ gs_guc set -N all -I all -c "enable_cbm_tracking = on" -D
/data/opengauss3.0/data/db/
$ gs_om -t restart
# 步骤一：初始化备份目录。
$ gs_probackup init -B /home/omm/probackup
# 步骤二：新建备份实例。
$ gs_probackup add-instance -B /home/omm/probackup/ -D
/data/opengauss3.0/data/db/ --instance=db（自定义实例名称）
# 步骤三：创建测试数据，进行全量备份。
$ gsql -p 15432 -d postgres -c "create table mytb04(id number);insert into
mytb04 values(1),(2);"
$ gs_probackup backup -B /home/omm/probackup/ --instance=db -d postgres -b
FULL -p 15432
# 步骤四：创建测试数据，进行增量备份。
$ gsql -p 15432 -d postgres -c "create table mytb05(id number);insert into
mytb05 values(1),(2),(3);"
$ gs_probackup backup -B /home/omm/probackup/ --instance=db -d postgres -b
PTRACK -p 15432
# 步骤五：查询备份信息。
$ gs_probackup show -B /home/omm/probackup/
# 步骤六：删除数据库，进行恢复。
$ rm -rf /data/opengauss3.0/data/db/*
$ gs_probackup restore  -B /home/omm/probackup/ --instance=db -p 15432 -d
postgres
$ gs_ctl build -D /data/opengauss3.0/data/db/
$ gs_ctl restart -D /data/opengauss3.0/data/db/
# 步骤七：将增量备份合并到全量备份中。
$ gs_probackup show -B /home/omm/probackup/ --instance=db
$ gs_probackup merge -B /home/omm/probackup/ --instance=db -i RRR8CI(备份ID)
# 步骤八：删除备份。
$ gs_probackup delete -B /home/omm/probackup/ --instance=db -i RRR76M
$ gs_probackup show -B /home/omm/probackup/ --instance=db
```

8. 闪回技术。

```
# 步骤一：修改参数。
$ gs_guc set -N all -I all -c "undo_zone_count=16384"
$ gs_guc set -N all -I all -c "enable_recyclebin=on"
$ gs_guc set -N all -I all -c "recyclebin_retention_time=30min"
$ gs_guc set -N all -I all -c "enable_default_ustore_table=on"
$ gs_guc set -N all -I all -c "version_retention_age=10000"
$ gs_guc set -N all -I all -c "undo_retention_time=1800000"
$ gs_om -t restart
# 步骤二：创建测试数据。
$ gsql -p 15432 -d postgres -c "create table mytb06(id number(1));insert
into mytb06 values(1),(2),(3);select * from mytb06;"
# 步骤三：删除表，查询回收站，闪回到删除之前。
$ gsql -p 15432 -d postgres -c "DROP TABLE mytb06;SELECT * FROM
gs_recyclebin;TIMECAPSULE TABLE mytb06 to BEFORE DROP;\dt;select * from
mytb06;"
# 步骤四：清空表，查询回收站，闪回到清空之前。
gsql -p 15432 -d postgres -c "TRUNCATE mytb06;SELECT * FROM
gs_recyclebin;TIMECAPSULE TABLE mytb06 to BEFORE TRUNCATE;\dt;select * from
mytb06;"
# 步骤五：修改或删除数据，查询时间戳。
gsql -r -p 15432 -d postgres -c "select * from mytb06;SELECT
current_timestamp;"
gsql -r -p 15432 -d postgres -c "update mytb06 SET id=4 where id=1;SELECT
current_timestamp;"
gsql -r -p 15432 -d postgres -c "delete from mytb06 where id=2;SELECT
current_timestamp;select * from mytb06;"
# 步骤六：闪回查询。
## 使用 TIMESTAMP 闪回查询。
gsql -r -p 15432 -d postgres -c "SELECT * FROM mytb06 TIMECAPSULE TIMESTAMP
'2023-03-19 16:18:48.719653+08';"
## 使用 CSN 闪回查询。
gsql -r -p 15432 -d postgres -c "SELECT snptime,snpcsn FROM gs_txn_snapshot
WHERE snptime BETWEEN '2023-03-19 16:18:00' AND '2023-03-19 16:19:00';"
gsql -r -p 15432 -d postgres -c "SELECT * FROM mytb06 TIMECAPSULE CSN 2330;"
# 步骤七：闪回表。
## 使用 TIMESTAMP 闪回。
gsql -r -p 15432 -d postgres -c "TIMECAPSULE TABLE mytb06 TO TIMESTAMP
'2023-03-19 16:19:16.205218+08';SELECT * FROM mytb06;"
## 使用 CSN 闪回。
gsql -r -p 15432 -d postgres -c "SELECT snptime,snpcsn FROM gs_txn_snapshot
WHERE snptime BETWEEN '2023-03-19 16:19:00' AND '2023-03-19 16:19:30';"
gsql -r -p 15432 -d postgres -c "TIMECAPSULE TABLE mytb06 TOTIMECAPSULE CSN
2373;"
```

9. 防篡改数据库。

```
# 步骤一：创建防篡改模式。
$ gsql -r -p 15432 -d postgres -c "CREATE SCHEMA schema01 WITH
BLOCKCHAIN;\dn;"
# 步骤二：在防篡改模式下创建表（防篡改表的最高列数为1600-1=1599）,插入数据。
$ gsql -r -p 15432 -d postgres -c "create table schema01.tb01(id
number);insert into schema01.tb01 values(1),(2);select *, hash from
schema01.tb01”
# 查询全局区块表记录。
$ gsql -r -p 15432 -d postgres -c "SELECT * FROM gs_global_chain;”
# 查询历史表记录（<>里面填写相应内容）。
$ gsql -r -p 15432 -d postgres -c "SELECT * FROM blockchain.
<schemaname>_<tablename>_hist;”
# 查询防篡改相关函数。
$ gsql -r -p 15432 -d postgres -c "select proname from pg_proc where proname
like 'le%';"
```

10. 全密态等值查询。

```
# 步骤一：配置环境变量 LOCALKMS_FILE_PATH。(客户端主密钥存储路径)
$ export LOCALKMS_FILE_PATH=/data/cmk
$ mkdir $LOCALKMS_FILE_PATH
# 步骤二：创建CMK。
$ gsql -r -p 15432 -d postgres -C -c 'CREATE CLIENT MASTER KEY cmk01 WITH
(KEY_STORE = localkms, KEY_PATH = "cmk01", ALGORITHM = RSA_2048);'
# 步骤三：创建CEK。
$ gsql -r -p 15432 -d postgres -C -c "CREATE COLUMN ENCRYPTION KEY cek01
WITH VALUES (CLIENT_MASTER_KEY = cmk01, ALGORITHM  =
AEAD_AES_256_CBC_HMAC_SHA256);"
# 步骤四：创建加密表。
$ gsql -r -p 15432 -d postgres -C -c "CREATE TABLE mytb08(id number
ENCRYPTED WITH ( COLUMN_ENCRYPTION_KEY = cek01, ENCRYPTION_TYPE =
DETERMINISTIC));";
```