基于 openGauss 的场景化综合实验实验报告

2112514 辛浩然

一、实验环境

基于 openGauss

原因：之前通过一些实验熟悉了 openGauss，积累过一些基于 openGauss 进行数据库操作的经验与方法。

二、实验步骤及截图（包括 sql 查询语句与关系代数表达式）

1.1.3 创建数据表

步骤一：创建金融数据库 finance

1.登录 omm 用户：

```
su - omm
```

2.启动数据库服务：

```
gs_om -t start;
```

3.使用 gsql 工具登陆数据库：

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

4.创建数据库 finance，使用 UTF8 编码方式：

```
CREATE DATABASE finance ENCODING 'UTF8' template = template0;
```

5.连接 finance 数据库：

```
\connect finance
```

6.创建名为 finance 的模式：

```
CREATE SCHEMA finance;
SET search_path TO finance;
```

```
[root@opengauss01 ~]# su - omm
Last login: Sun Apr 16 13:23:58 CST 2023 on pts/0


Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time:  Mon May  8 14:02:42 CST 2023

System load:    0.04
Processes:      149
Memory used:    10.2%
Swap used:      0.0%
Usage On:       14%
IP address:     192.168.0.146
Users online:   1



[omm@opengauss01 ~]$ gs_om -t start;
Starting cluster.
=========================================
[SUCCESS] opengauss01
2023-05-08 14:02:55.269 6458908f.1 [unknown] 281473738407952 [unknown] 0 dn_6001 01000  0 [BACKEND] WARNING:  could not create any HA TCP/IP sockets
2023-05-08 14:02:55.273 6458908f.1 [unknown] 281473738407952 [unknown] 0 dn_6001 01000  0 [BACKEND] WARNING:  Failed to initialize the memory protect for g_instan
ce.attr.attr_storage.cstore_buffers (16 Mbytes) or shared memory (2363 Mbytes) is larger.
=========================================
Successfully started.
[omm@opengauss01 ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 21:03:52 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# CREATE DATABASE finance ENCODING 'UTF8' template = template0;
CREATE DATABASE
postgres=# \connect finance
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "finance" as user "omm".
finance=# CREATE SCHEMA finance;
CREATE SCHEMA
finance=# SET search_path TO finance;
SET
finance=#
```

步骤二：创建各个表：

```
finance=# CREATE TABLE insurance
finance-# (
finance(#         i_name VARCHAR(100) NOT NULL,
finance(#         i_id INT PRIMARY KEY,
finance(#         i_amount INT,
finance(#         i_person CHAR(20),
finance(#         i_year INT,
finance(#         i_project VARCHAR(200)
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "insurance_pkey" for table "insurance"
CREATE TABLE
finance=# CREATE TABLE fund
finance-# (
finance(#         f_name VARCHAR(100) NOT NULL,
finance(#         f_id INT PRIMARY KEY,
finance(#         f_type CHAR(20),
finance(#         f_amount INT,
finance(#         risk_level CHAR(20) NOT NULL,
finance(#         f_manager INT NOT NULL
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "fund_pkey" for table "fund"
CREATE TABLE
finance=# CREATE TABLE property
finance-# (
finance(#         pro_c_id INT NOT NULL,
finance(#         pro_id INT PRIMARY KEY,
finance(#         pro_status CHAR(20),
finance(#         pro_quantity INT,
finance(#         pro_income INT,
finance(#         pro_purchase_time DATE
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "property_pkey" for table "property"
CREATE TABLE
```

```
finance=# CREATE TABLE client
finance-# (
finance(#         c_id INT PRIMARY KEY,
finance(#         c_name VARCHAR(100) NOT NULL,
finance(#         c_mail CHAR(30) UNIQUE,
finance(#         c_id_card CHAR(20) UNIQUE NOT NULL,
finance(#         c_phone CHAR(20) UNIQUE NOT NULL,
finance(#         c_password CHAR(20) NOT NULL
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "client_pkey" for table "client"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "client_c_mail_key" for table "client"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "client_c_id_card_key" for table "client"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "client_c_phone_key" for table "client"
CREATE TABLE
finance=# CREATE TABLE bank_card
finance-# (
finance(#         b_number CHAR(30) PRIMARY KEY,
finance(#         b_type CHAR(20),
finance(#         b_c_id INT NOT NULL
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "bank_card_pkey" for table "bank_card"
CREATE TABLE
finance=#
finance=# CREATE TABLE finances_product
finance-# (
finance(#         p_name VARCHAR(100) NOT NULL,
finance(#         p_id INT PRIMARY KEY,
finance(#         p_description VARCHAR(4000),
finance(#         p_amount INT,
finance(#         p_year INT
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "finances_product_pkey" for table "finances_product"
CREATE TABLE
```

1.1.4 插入表数据

插入数据后查询：

```
finance=# select count(*) from client;
 count
-------
    30
(1 row)
```

```
finance=# select count(*) from bank_card;
 count
-------
    20
(1 row)
```

```
finance=# select count(*) from finances_product;
 count
-------
     4
(1 row)
```

```
finance=# select count(*) from insurance;
 count
-------
     5
(1 row)
```

```
finance=# select count(*) from fund;
 count
-------
     4
(1 row)
```

```
finance=# select count(*) from property;
 count
-------
     4
(1 row)
```

1.1.5 手工插入一条数据

属性冲突的情况：c_id_card 和 c_phone 为唯一且非空，插入的数据与已有数据冲突，插入失败。

```
finance=# INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (31,
','lili@huawei.com','340211199301010005','18815650005','gaussdb_005');
ERROR:  duplicate key value violates unique constraint "client_c_id_card_key"
DETAIL:  Key (c_id_card)=(340211199301010005  ) already exists.
finance=#
```

插入成功的情况：

```
finance=# INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (31,'李丽
','lili@huawei.com','340211199301010031','18815650031','gaussdb_031');
INSERT 0 1
finance=#
```

1.1.6 添加约束

为 finances_product 表的 p_amount 列添加大于等于 0 的约束。尝试手工插入一条金额小于 0 的记录，插入失败。

```
finance=# alter table finances_product add constraint c_p_mount check (p_amount>=0);
ALTER TABLE
finance=# INSERT INTO finances_product(p_name,p_id,p_description,p_amount,p_year) VALUES ('信
贷资产',10,'一般指银行作为委托人将通过发行理财产品募集资金委托给信托公司，信托公司作为受托人成
立信托计划，将信托资产购买理财产品发售银行或第三方信贷资产。',-10,6);
ERROR:  new row for relation "finances_product" violates check constraint "c_p_mount"
DETAIL:  Failing row contains (信贷资产, 10, 一般指银行作为委托人将通过发行理财产品募集..., -1
0, 6).
```

为 fund 表的 f_amount 列添加大于等于 0 的约束；为 insurance 表的 i_amount 列添加大于等于 0 的约束。

```
finance=# alter table fund add constraint c_f_mount check (f_amount>=0);
ALTER TABLE
finance=# alter  table insurance add constraint c_i_mount check (i_amount >=0);
ALTER TABLE
```

1.1.7 查询数据（SQL 语句与关系代数表达式）

步骤一 单表查询

查询银行卡信息表。

select b_number,b_type from bank_card;

$$\pi_{(b\_number,b\_type,b\_c\_id)}(bank\_card)$$

```
finance=# select  b_number,b_type from bank_card;
           b_number            |      b_type
-------------------------------+------------------
 6222021302020000001           | 信用卡
 6222021302020000002           | 信用卡
 6222021302020000003           | 信用卡
 6222021302020000004           | 信用卡
 6222021302020000005           | 信用卡
 6222021302020000006           | 信用卡
 6222021302020000007           | 信用卡
 6222021302020000008           | 信用卡
 6222021302020000009           | 信用卡
 6222021302020000010           | 信用卡
 6222021302020000011           | 储蓄卡
 6222021302020000012           | 储蓄卡
 6222021302020000013           | 储蓄卡
 6222021302020000014           | 储蓄卡
 6222021302020000015           | 储蓄卡
 6222021302020000016           | 储蓄卡
 6222021302020000017           | 储蓄卡
 6222021302020000018           | 储蓄卡
 6222021302020000019           | 储蓄卡
 6222021302020000020           | 储蓄卡
(20 rows)
```

步骤二 条件查询

查询资产信息中可用的资产数据。

select *from property where pro_status='可用';

$$\pi_{pro\_c\_id,pro\_id,pro\_status,pro\_quantity,pro\_income,pro\_purchase\_time}\left(\sigma_{pro\_status='可用'}(property)\right)$$

```
finance=# select *from property where pro_status='可用';
 pro_c_id | pro_id |     pro_status     | pro_quantity | pro_income |  pro_purchase_time
----------+--------+--------------------+--------------+------------+---------------------
        5 |      1 | 可用               |            4 |       8000 | 2018-07-01 00:00:00
       10 |      2 | 可用               |            4 |       8000 | 2018-07-01 00:00:00
       15 |      3 | 可用               |            4 |       8000 | 2018-07-01 00:00:00
(3 rows)
```

步骤三 聚合查询

查询用户表中有多少个用户。

select count(*) from client;

$$COUNT(c\_id)$$

```
finance=# select count(*) from client;
 count
-------
    31
(1 row)
```

查询银行卡信息表中，储蓄卡和信用卡的个数。

select b_type,count(*) from bank_card group by b_type;

$$\pi_{b\_byte,COUNT\left(\gamma_{b\_type}(bank\_card)\right)}(bank\_card)$$

```
finance=# select b_type,count(*) from bank_card group by b_type;
      b_type       | count
-------------------+-------
 储蓄卡            |    10
 信用卡            |    10
(2 rows)
```

查询保险信息表中，保险金额的平均值。

select avg(i_amount) from insurance;

$$AVG(i\_amount)$$

```
finance=# select avg(i_amount) from insurance;
          avg
-----------------------
 2700.0000000000000000
(1 row)
```

查询保险信息表中保险金额的最大值和最小值所对应的险种和金额。

select i_name,i_amount from insurance where i_amount >= all (select i_amount from insurance);

select i_name,i_amount from insurance where i_amount <= all (select i_amount from insurance);

$$\pi_{i\_name,i\_amount}\left(\sigma_{i\_amount=MAX(i\_amount)}(insurance)\right)$$

$$\pi_{i\_name,i\_amount}\left(\sigma_{i\_amount=MIN(i\_amount)}(insurance)\right)$$

```
finance=# Select i_name,i_amount from insurance where i_amount >= all (select i_amount from in
surance);
  i_name  | i_amount
----------+----------
 意外保险 |     5000
(1 row)

finance=# Select i_name,i_amount from insurance where i_amount <= all (select i_amount from in
surance);
    i_name    | i_amount
--------------+----------
 财产损失保险 |     1500
(1 row)
```

步骤四 连接查询

查询用户编号在银行卡表中出现的用户的编号，用户姓名和身份证。

select c_id,c_name,c_id_card from client where exists (select * from bank_card where bank_card.b_c_id=client.c_id);

$$\pi_{c\_id,c\_name,c\_id\_card}(client \bowtie_{bank\_card.b\_c\_id=client.c_{id}} bank\_card)$$

```
finance=# Select c_id,c_name,c_id_card from client where exists (select * f
rom bank_card where bank_card.b_c_id=client.c_id);
 c_id | c_name |      c_id_card
------+--------+---------------------
    1 | 张一   | 340211199301010001
    3 | 张三   | 340211199301010003
    5 | 张五   | 340211199301010005
    7 | 张七   | 340211199301010007
    9 | 张九   | 340211199301010009
   10 | 李一   | 340211199301010010
   12 | 李三   | 340211199301010012
   14 | 李五   | 340211199301010014
   16 | 李七   | 340211199301010016
   18 | 李九   | 340211199301010018
   19 | 王一   | 340211199301010019
   21 | 王三   | 340211199301010021
   23 | 王五   | 340211199301010023
   24 | 王六   | 340211199301010024
   26 | 王八   | 340211199301010026
   27 | 王九   | 340211199301010027
   29 | 钱二   | 340211199301010029
(17 rows)
```

查询银行卡号不是‘62220213020200000l*’（*表示未知）的用户的编号，姓名和身份证。

select c_id,c_name,c_id_card from client where c_id not in(select b_c_id from bank_card where b_number like '622202130202000001%');

不能使用关系代数表达式表示

```
finance=# select c_id,c_name,c_id_card from client where c_id not in(select
 b_c_id from bank_card where b_number like '622202130202000001%');
 c_id | c_name |      c_id_card
------+--------+---------------------
    1 | 张一   | 340211199301010001
    2 | 张二   | 340211199301010002
    4 | 张四   | 340211199301010004
    5 | 张五   | 340211199301010005
    6 | 张六   | 340211199301010006
    8 | 张八   | 340211199301010008
    9 | 张九   | 340211199301010009
   10 | 李一   | 340211199301010010
   11 | 李二   | 340211199301010011
   13 | 李四   | 340211199301010013
   14 | 李五   | 340211199301010014
   15 | 李六   | 340211199301010015
   16 | 李七   | 340211199301010016
   17 | 李八   | 340211199301010017
   20 | 王二   | 340211199301010020
   22 | 王四   | 340211199301010022
   25 | 王七   | 340211199301010025
   28 | 钱一   | 340211199301010028
   29 | 钱二   | 340211199301010029
   30 | 钱三   | 340211199301010030
   31 | 李丽   | 340211199301010031
(21 rows)
```

步骤五 子查询

通过子查询，查询保险产品中保险金额大于平均值的保险名称和适用人群。

select i.i_name,i.i_person from insurance i where i.i_amount>(select avg(i_amount) from insurance);

$$\pi_{i\_name,i\_person}\left(\sigma_{i\_amount>AVG(i\_amount)}(insurance)\right)$$

```
finance=# Select i.i_name,i.i_person from insurance i where i.i_amount>(sel
ect avg(i_amount) from insurance);
  i_name  |      i_person
----------+--------------------
 人寿保险 | 老人
 意外保险 | 所有人
(2 rows)
```

步骤六 ORDER BY 和 GROUP BY

按照降序查询保险编号大于 2 的保险名称，保额和适用人群。

select i_name,i_amount,i_person from insurance where i_id>2 order by i_amount desc;

关系代数无法实现降序排列。

查询：$\pi_{i\_name,i\_amount,i\_person}\left(\sigma_{i\_id>2}\ (insurance)\right)$

```
finance=# select i_name,i_amount,i_person from insurance where i_id>2 order
 by i_amount  desc;
    i_name    | i_amount |     i_person
--------------+----------+------------------
 意外保险     |     5000 | 所有人
 医疗保险     |     2000 | 所有人
 财产损失保险 |     1500 | 中年人
(3 rows)
```

查询各保险信息总数，按照 p_year 分组。

select p_year,count(*) from finances_product group by p_year;

$$\pi_{p\_year,COUNT(p\_id)}\left(\gamma_{p\_year}(finance\_product)\right)$$

```
finance=# select p_year,count(*) from finances_product group by p_year;
 p_year | count
--------+-------
      6 |     4
(1 row)
```

步骤七 HAVING 和 WITH AS

查询保险金额统计数量等于 2 的适用人群数。

select i_person,count(i_amount) from insurance group by i_person having count(i_amount)=2;

$$\pi_{i\_person,COUNT(i\_amount)}\left(\sigma_{COUNT(i\_amount)=2}(insurance)\right)$$

```
finance=# select i_person,count(i_amount) from insurance group by i_person
having
finance-# count(i_amount)=2;
      i_person       | count
---------------------+-------
 老人                |     2
 所有人              |     2
(2 rows)
```

使用 WITH AS 查询基金信息表。

with temp as (select* from fund)

select * from temp;

$$\pi_{f\_name,f\_id,f\_type,f\_amount,risk\_level,f_{manager}}(fund)$$

```
finance=# WITH temp AS (SELECT * FROM fund)
finance-# SELECT * FROM temp;
   f_name     | f_id |      f_type       | f_amount |     risk_level      | f_manager
--------------+------+-------------------+----------+---------------------+-----------
 股票         |    1 | 股票型            |    10000 | 高                  |         1
 投资         |    2 | 债券型            |    10000 | 中                  |         2
 国债         |    3 | 货币型            |    10000 | 低                  |         3
 沪深300指数  |    4 | 指数型            |    10000 | 中                  |         4
(4 rows)
```

1.1.8 视图

针对“查询用户编号在银行卡表中出现的用户的编号，用户姓名和身份证”的查询，创建视图。使用视图进行查询。

CREATE VIEW v_client as SELECT c_id,c_name,c_id_card FROM client WHERE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id);

```
finance=# CREATE VIEW v_client as SELECT c_id,c_name,c_id_card FROM client
WHERE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id)
;
CREATE VIEW
finance=# SELECT * FROM v_client;
 c_id | c_name |      c_id_card
------+--------+---------------------
    1 | 张一   | 340211199301010001
    3 | 张三   | 340211199301010003
    5 | 张五   | 340211199301010005
    7 | 张七   | 340211199301010007
    9 | 张九   | 340211199301010009
   10 | 李一   | 340211199301010010
   12 | 李三   | 340211199301010012
   14 | 李五   | 340211199301010014
   16 | 李七   | 340211199301010016
   18 | 李九   | 340211199301010018
   19 | 王一   | 340211199301010019
   21 | 王三   | 340211199301010021
   23 | 王五   | 340211199301010023
   24 | 王六   | 340211199301010024
   26 | 王八   | 340211199301010026
   27 | 王九   | 340211199301010027
   29 | 钱二   | 340211199301010029
(17 rows)
```

修改视图，在原有查询的基础上，过滤出信用卡用户。

CREATE OR REPLACE VIEW v_client as SELECT c_id,c_name,c_id_card FROM client WHERE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id and bank_card.b_type='信用卡');

```
finance=# CREATE OR REPLACE VIEW v_client as SELECT c_id,c_name,c_id_card F
ROM client WHERE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_c
ard.b_c_id and bank_card.b_type='信用卡');
CREATE VIEW
finance=# select * from v_client;
 c_id | c_name |      c_id_card
------+--------+---------------------
    1 | 张一   | 340211199301010001
    3 | 张三   | 340211199301010003
    5 | 张五   | 340211199301010005
    7 | 张七   | 340211199301010007
    9 | 张九   | 340211199301010009
   10 | 李一   | 340211199301010010
   12 | 李三   | 340211199301010012
   14 | 李五   | 340211199301010014
   16 | 李七   | 340211199301010016
   18 | 李九   | 340211199301010018
(10 rows)
```

修改视图名称

ALTER VIEW v_client RENAME TO v_client_new;

删除视图

DROP VIEW v_client_new;

```
finance=# ALTER VIEW v_client RENAME TO v_client_new;
ALTER VIEW
finance=# DROP VIEW v_client_new;
DROP VIEW
```

1.1.9 索引

在普通表 property 上创建索引。

CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_time);

在普通表 property 上重建及重命名索引。

重建索引：

DROP INDEX idx_property;

CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_time);

重命名索引：

ALTER INDEX idx_property RENAME TO idx_property_temp;

删除索引：

DROP INDEX idx_property_temp;

```
finance=# CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_time);
CREATE INDEX
finance=# DROP INDEX idx_property;
DROP INDEX
finance=# CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_time);
CREATE INDEX
finance=#
finance=# ALTER INDEX idx_property RENAME TO idx_property_temp;
ALTER INDEX
finance=# DROP INDEX idx_property_temp;
DROP INDEX
```

1.1.10 数据的修改和删除

步骤一 修改数据

修改/更新银行卡信息表中 b_c_id 小于 10 和客户信息表中 c_id 相同的记录的 b_type 字段。

```
finance=# UPDATE bank_card SET bank_card.b_type='借记卡' from client where bank_card.b_c_id =
client.c_id and bank_card.b_c_id<10;
UPDATE 7
finance=# SELECT * FROM bank_card ORDER BY b_c_id;
          b_number           |      b_type       | b_c_id
-----------------------------+-------------------+--------
 6222021302020000001         | 借记卡            |      1
 6222021302020000002         | 借记卡            |      3
 6222021302020000016         | 借记卡            |      3
 6222021302020000003         | 借记卡            |      5
 6222021302020000013         | 借记卡            |      7
 6222021302020000004         | 借记卡            |      7
 6222021302020000005         | 借记卡            |      9
 6222021302020000006         | 信用卡            |     10
 6222021302020000007         | 信用卡            |     12
 6222021302020000019         | 储蓄卡            |     12
 6222021302020000008         | 信用卡            |     14
 6222021302020000009         | 信用卡            |     16
 6222021302020000010         | 信用卡            |     18
 6222021302020000011         | 储蓄卡            |     19
 6222021302020000012         | 储蓄卡            |     21
 6222021302020000014         | 储蓄卡            |     23
 6222021302020000015         | 储蓄卡            |     24
 6222021302020000017         | 储蓄卡            |     26
 6222021302020000018         | 储蓄卡            |     27
 6222021302020000020         | 储蓄卡            |     29
(20 rows)
```

步骤二 删除数据

```
finance=# SELECT * FROM fund;
   f_name    | f_id |      f_type       | f_amount |     risk_level      | f_manager
-------------+------+-------------------+----------+---------------------+-----------
 股票        |    1 | 股票型            |    10000 | 高                  |         1
 投资        |    2 | 债券型            |    10000 | 中                  |         2
 国债        |    3 | 货币型            |    10000 | 低                  |         3
 沪深300指数 |    4 | 指数型            |    10000 | 中                  |         4
(4 rows)

finance=# DELETE FROM fund WHERE f_id<3;
DELETE 2
finance=# SELECT * FROM fund;
   f_name    | f_id |      f_type       | f_amount |     risk_level      | f_manager
-------------+------+-------------------+----------+---------------------+-----------
 国债        |    3 | 货币型            |    10000 | 低                  |         3
 沪深300指数 |    4 | 指数型            |    10000 | 中                  |         4
(2 rows)
```

1.1.11 新用户的创建和授权

创建新用户，授予 finance 数据库下 bank_card 表的查询和插入权限，以及 SCHEMA 的权限。

```
finance=# CREATE USER dbuser1 IDENTIFIED BY 'Aa111111';
CREATE ROLE
finance=# GRANT SELECT,INSERT ON finance.bank_card TO dbuser1;
GRANT
finance=# GRANT ALL ON SCHEMA finance to dbuser1;
GRANT
```

1.1.12 新用户连接数据库

访问 finance 数据库的表 bank_card。

```
[omm@opengauss01 ~]$ gsql -d finance -U dbuser1 -p 26000;
Password for user dbuser1:
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 21:03:52 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

finance=> select * from finance. bank_card where b_c_id<10;
           b_number            |      b_type       | b_c_id
-------------------------------+-------------------+--------
 6222021302020000001           | 借记卡            |      1
 6222021302020000002           | 借记卡            |      3
 6222021302020000003           | 借记卡            |      5
 6222021302020000004           | 借记卡            |      7
 6222021302020000005           | 借记卡            |      9
 6222021302020000013           | 借记卡            |      7
 6222021302020000016           | 借记卡            |      3
(7 rows)
```

1.1.13 删除 schema

使用操作系统 omm 用户使用 gsql，新建 session

使用“\dn”查看数据库下的 schema

```
[omm@opengauss01 ~]$ gsql -d finance -p 26000;
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 21:03:52 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

finance=# \dn
    List of schemas
    Name     |  Owner
-------------+---------
 cstore      | omm
 dbe_perf    | omm
 dbuser      | dbuser
 dbuser1     | dbuser1
 finance     | omm
 pkg_service | omm
 public      | omm
 snapshot    | omm
(8 rows)
```

设置默认查询为 finance，

使用“\dt”命令可以看到在 finance 中的对象

```
finance=# set search_path to finance;
SET
finance=# \dt
                        List of relations
 Schema  |       Name       | Type  | Owner |             Storage
---------+------------------+-------+-------+---------------------------------
 finance | bank_card        | table | omm   | {orientation=row,compression=no}
 finance | client           | table | omm   | {orientation=row,compression=no}
 finance | finances_product | table | omm   | {orientation=row,compression=no}
 finance | fund             | table | omm   | {orientation=row,compression=no}
 finance | insurance        | table | omm   | {orientation=row,compression=no}
 finance | property         | table | omm   | {orientation=row,compression=no}
(6 rows)
```

使用 DROP SCHEMA 命令删除 finance 有报错，因为 finance 下存在对象。

使用 DROP SCHEMA…..CASCADE 删除，会将 finance 连同下的对象一起删除。

使用“\dt”命令可以看到在 finance 和 public 中的对象，对象已删除。

```
finance=# DROP SCHEMA finance;
ERROR:  cannot drop schema finance because other objects depend on it
DETAIL:  table client depends on schema finance
table bank_card depends on schema finance
table finances_product depends on schema finance
table insurance depends on schema finance
table fund depends on schema finance
table property depends on schema finance
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
finance=# DROP SCHEMA finance CASCADE;
NOTICE:  drop cascades to 6 other objects
DETAIL:  drop cascades to table client
drop cascades to table bank_card
drop cascades to table finances_product
drop cascades to table insurance
drop cascades to table fund
drop cascades to table property
DROP SCHEMA
finance=# \dt
No relations found.
```

三、总结分析

实验时长：约 4 小时；

遇到的问题：不了解 with as 语句；

学习到的知识点：1.学习到 with as 语句的用法。可以定义一个 SQL 片断，该 SQL 片断会被整个 SQL 语句用到。可以使 SQL 语句的可读性更高。2.进一步查缺补漏，掌握了查询等操作的 SQL 语句书写，熟练了代数关系表达式的用法。