**《数据库系统实验》指导书**

**广东工业大学**

**计算机学院**

### 实验六 **openGauss场景化综合应用实验**

**实验项目名称：openGauss场景化综合应用实验**

**实验项目性质：验证型**

**所属课程名称：数据库系统**

**一、实验目的**

1、掌握openGauss数据库的基本操作和SQL语法；

2、运用openGauss数据库的实现场景化应用；

**二、实验内容和要求**

1、在openGauss中创建数据库、模式、用户、表，并熟悉openGauss的操作命令与方式；

2、利用openGauss数据库解决金融场景的实际应用；

3、利用JDBC连接创建的数据库，实现数据库的初步应用；

**三、实验主要仪器设备和材料**

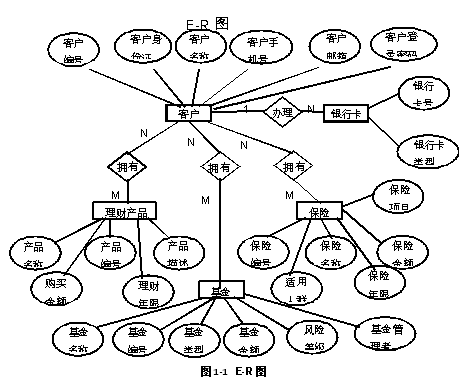
1．计算机及操作系统：PC机，Windows系统/Centos系统；

2．数据库管理系统：openGauss；

**四、实验背景**

设A市C银行为了方便对银行数据的管理和操作，引入了华为openGauss数据库。针对C银行的业务，本实验主要将对象分为客户、银行卡、理财产品、保险、基金。因此，针对这些数据库对象，本实验假设C银行的金融数据库存在着以下关系：客户可以办理银行卡，同时客户可以购买不同的银行产品，如理财产品，基金和保险。那么，根据C银行的对象关系，本实验给出了相应的关系模式和ER图，并对其进行较为复杂的数据库操作。

**1 实验E-R图**



**2关系模式**

对于C银行中的5个对象，分别建立属于每个对象的属性集合，具体属性描述如下：

* 客户（客户编号、客户名称、客户邮箱，客户身份证，客户手机号，客户登录密码）
* 银行卡（银行卡号，银行卡类型）
* 理财产品（产品名称，产品编号，产品描述，购买金额，理财年限）
* 保险（保险名称，保险编号，保险金额，适用人群，保险年限，保障项目）
* 基金（基金名称，基金编号，基金类型，基金金额，风险等级，基金管理者）

**3.对象之间的关系**

一个客户可以办理多张银行卡

一个客户可以购买多个理财产品，同一类理财产品可由多个客户购买

一个客户可以购买多个基金，同一类基金可由多个客户购买

一个客户可以购买多个保险，同一类保险可由多个客户购买

说明：

由于一个客户可以办理多张银行卡，所以银行卡表引用客户表的客户编号作为外键。

由于一个客户可以购买多个理财产品，同一类理财产品可由多个客户购买。所以生成关系表——**资产表**。资产表引用客户表的商品编号作为外键，引用理财产品表的产品编号作为外键，并且添加商品状态、商品数量、商品收益和购买时间等属性。

客户和保险、客户和基金的关系同理，所以资产表同样作为生成的关系表，修改资产表的理财产品编号为商品编号，商品编号引用自理财产品表、保险和基金表的编号。

**4.物理模型**

对象及属性对应的编号为：

* Client(c\_id，c\_name，c\_mail，c\_id\_card，c\_phone，c\_password)
* bank\_card(b\_number，b\_type，**b\_c\_id**)
* finances\_product(p\_name，p\_id，p\_description，p\_amount，p\_year)
* insurance(i\_name，i\_id，i\_amount，i\_person，i\_year，i\_project)
* fund(f\_name，f\_id，f\_type，f\_amount，risk\_level，f\_manager)
* property(**pro\_id**,pro\_c\_id，pro\_pif\_id，**pro\_type**，pro\_status，pro\_quantity，pro\_income，pro\_purchase\_time)

说明：

资产表（property）中由于商品编号(pro\_pif\_id)字段引用自理财产品表、保险和基金表的id字段，为了防止三商品的id字段相互冲突，添加商品类型(pro\_type)字段区分三种商品。并且资产表添加资产编号(pro\_id)字段作为主键。

Client（客户）表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **约束** | **说明** |
| c\_id | INTEGER | PRIMARY KEY | 客户编码 |
| c\_name | VARCHAR(100) | NOT NULL | 客户名称 |
| c\_mail | CHAR(30) | UNIQUE | 客户邮箱 |
| c\_id\_card | CHAR(20) | UNIQUE NOT NULL | 客户身份证 |
| c\_phone | CHAR(20) | UNIQUE NOT NULL | 客户手机号 |
| c\_password | CHAR(20) | NOT NULL | 客户登录密码 |

bank\_card（银行卡）表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **约束** | **说明** |
| b\_number | CHAR(30) | PRIMARY KEY | 银行卡号 |
| b\_type | CHAR(20) |  | 银行卡类型 |
| b\_c\_id | INTEGER | NOT NULL **FOREIGN KEY** | 所属客户编号  注：本字段引用自client表的c\_id字段。 |

finances\_product（理财产品）信息表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **约束** | **说明** |
| p\_name | VARCHAR(100) | NOT NULL | 产品名称 |
| p\_id | INTEGER | PRIMARY KEY | 产品编号 |
| p\_description | VARCHAR(4000) |  | 产品描述 |
| p\_amount | INTEGER |  | 购买金额 |
| p\_year | INTEGER |  | 理财年限 |

insurance（保险）表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **约束** | **说明** |
| i\_name | VARCHAR(100) | NOT NULL | 保险名称 |
| i\_id | INTEGER | PRIMARY KEY | 保险编号 |
| i\_amount | INTEGER |  | 保险金额 |
| i\_person | CHAR(20) |  | 适用人群 |
| i\_year | INTEGER |  | 保险年限 |
| i\_project | VARCHAR(200) |  | 保障项目 |

fund（基金）表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **约束** | **说明** |
| f\_name | VARCHAR(100) | NOT NULL | 基金名称 |
| f\_id | INTEGER | PRIMARY KEY | 基金编号 |
| f\_type | CHAR(20) |  | 基金类型 |
| f\_amount | INTEGER |  | 基金金额 |
| risk\_level | CHAR(20) | NOT NULL | 风险等级 |
| f\_manager | INTEGER | NOT NULL | 基金管理者  说明：本字段引用自银行员工表的id字段，由于简化系统没有在本实验展示员工表。 |

property（资本）表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **约束** | **说明** |
| pro\_id | INTEGER | PRIMARY KEY | 资产编号 |
| pro\_c\_id | VARCHAR(100) | NOT NULL **FOREIGN KEY** | 客户编号  说明：本字段引用自client表的c\_id字段。 |
| pro\_pif\_id | INTEGER | NOT NULL **FOREIGN KEY** | 商品编号  说明：本字段引用自finances\_product表、insurance表和fund表三个表的id字段。 |
| pro\_type | INTEGER | NOT NULL | 商品类型  说明：1表示理财产品；2表示保险；3表示基金。 |
| pro\_status | CHAR(20) |  | 商品状态 |
| pro\_quantity | INTEGER |  | 商品数量 |
| pro\_income | INTEGER |  | 商品收益 |
| pro\_purchase\_time | DATE |  | 购买时间 |

**五、实验方法、步骤及结果测试**

**1创建数据表**

根据C银行的场景描述，本实验分别针对客户(client)，银行卡(bank\_card)，理财产品(finances\_product)，保险(insurance)，基金(fund)和资产(property)创建相应的表。具体的实验步骤如下所示：

**（1）创建金融数据库finance。**

切换到omm用户，以操作系统用户omm登录数据库主节点。

**su - omm**

启动服务命令：

gs\_om -t start

使用gsql工具登陆数据库。

gsql -d postgres -p 26000 -r

创建数据库f311900xxxx。（**注意，每位同学根据自己学号修改的后四位）**

CREATE DATABASE f311900xxxx ENCODING 'UTF8' template = template0 ;

连接f311900xxxx数据库。

\connect f311900xxxx

创建名为f311900xxxx的schema，并设置f311900xxxx为当前的schema。

CREATE SCHEMA f311900xxxx;

将默认搜索路径设为f311900xxxx。

SET search\_path TO f311900xxxx;

**（2）客户信息表的创建。**

在SQL编辑框中输入如下语句，创建客户信息表client。

如果存在client则先删除表client。

|  |  |
| --- | --- |
| SQL语句  DROP TABLE IF EXISTS client; | SQL语句及执行结果截图显示 |

创建表client。

|  |  |
| --- | --- |
| SQL语句  create table client  (  c\_id varchar(100) primary key,  cname varchar(100) not null,  c\_mail char(30) unique,  c\_id\_card char(20) not null unique,  c\_phone char(20) not null unique,  c\_password char(20) not null,  ); | SQL语句及执行结果截图显示 |

**（3）银行卡信息表的创建。**

在SQL编辑框中输入如下语句，创建银行卡信息表bank\_card。

如果存在则删除表bank\_card。

|  |  |
| --- | --- |
| SQL语句  drop table if exists bank\_card; | SQL语句及执行结果截图显示 |

创建表bank\_card。

|  |  |
| --- | --- |
| SQL语句  create table bank\_card  (  b\_number char(30) primary key,  b\_type char(20),  b\_c\_id varchar(100) not null  ); | SQL语句及执行结果截图显示 |

**（4）理财产品信息表的创建。**

创建理财产品信息表finances\_product。

如果存在则删除表finances\_product。

|  |  |
| --- | --- |
| SQL语句  drop table if exists finances\_product; | SQL语句及执行结果截图显示 |

创建表finances\_product。

|  |  |
| --- | --- |
| SQL语句  create table finances\_product  (  p\_name varchar(100) not null,  p\_id varchar(100) primary key,  p\_description varchar(4000),  p\_amount integer,  p\_year integer  ); | SQL语句及执行结果截图显示 |

**（5）保险信息表的创建。**

在SQL编辑框中输入如下语句，创建保险信息表insurance。

如果存在则删除表insurance。

|  |  |
| --- | --- |
| SQL语句  drop table if exists insurance; | SQL语句及执行结果截图显示 |

创建表insurance。

|  |  |
| --- | --- |
| SQL语句  create table insurance  (  i\_name varchar(100) not null,  i\_id varchar(100) primary key,  i\_amount integer,  i\_person char(20),  i\_year integer,  i\_project varchar(200)  ); | SQL语句及执行结果截图显示 |

**（6）基金信息表的创建。**

在SQL编辑框中输入如下语句，创建保险信息表fund。

如果存在则删除表fund。

|  |  |
| --- | --- |
| SQL语句  drop table if exists fund; | SQL语句及执行结果截图显示 |

创建表fund。

|  |  |
| --- | --- |
| SQL语句  create table fund  (  f\_name varchar(100) not null,  f\_id varchar(100) primary key,  f\_type char(20),  f\_amount integer,  risk\_level char(20) not null,  f\_manager integer not null  ); | SQL语句及执行结果截图显示 |

**（7）资产信息表的创建。**

在SQL编辑框中输入如下语句，创建资产信息表property。

如果存在则删除表property。

|  |  |
| --- | --- |
| SQL语句  drop table if exists property; | SQL语句及执行结果截图显示 |

创建表property。

|  |  |
| --- | --- |
| SQL语句  create table property  (  pro\_id varchar(100) primary key,  pro\_c\_id varchar(100) not null,  pro\_pif\_id varchar(100) not null,  pro\_type integer not null,  pro\_status char(20),  pro\_quantity integer,  pro\_income integer,  pro\_purchase\_time date  ); | SQL语句及执行结果截图显示 |

**2.插入表数据**

为了实现对表数据的相关操作，本实验需要以执行SQL语句的方式对金融数据库的相关表插入部分数据。

**（1）对client表进行数据初始化。**

执行insert操作。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

查询插入结果。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示      注意用带个人学号的截图替换 |

**（2）对bank\_card表进行数据初始化。**

执行insert操作。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

查询插入结果。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

**（3）对finances\_product表进行数据初始化。**

执行insert操作。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

查询插入结果。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

**（4）对insurance表进行数据初始化。**

执行insert操作。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

查询插入结果。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

**（5）对fund表进行数据初始化。**

执行insert操作。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

查询插入结果。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

**（6）对property表进行数据初始化。**

执行insert操作。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

查询插入结果。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

**3.手工插入一条数据**

当C银行有新的信息需要加入数据库时，系统需要在对应的数据表中手动插入一条新的数据。因此，针对主键属性定义的场景，介绍如何手动插入一条数据。

**（1）在金融数据库的客户信息表中添加一个客户的信息。（属性冲突的场景）**

c\_id\_card和c\_phone非唯一。

INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (31,'李丽','lili@huawei.com','340211199301010005','18815650005','gaussdb\_005');

错误信息如下：

duplicate key value violates unique constraint "client\_c\_id\_card\_key"

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

说明：由于在表的创建过程中，实验定义了c\_id\_card和c\_phone为唯一且非空（UNIQUE NOT NULL），所以当表中存在时，插入数据失败。

**（2）在金融数据库的客户信息表中添加一个客户的信息。（插入成功的场景）。**

插入成功的示例。

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');

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

**4.添加约束**

**（1）在理财产品表、保险信息表和基金信息表中，都存在金额这个属性，在现实生活中，金额不会存在负数。因此针对表中金额的属性，增加大于0的约束条件。**

为finances\_product表的p\_amount列添加大于等于0的约束。

ALTER table finances\_product ADD CONSTRAINT c\_p\_mount CHECK (p\_amount >=0);

**（2）尝试手工插入一条金额小于0的记录。**

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);

执行失败，失败原因：new row for relation "finances\_product" violates check constraint "c\_p\_mount"。

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示      注意用带个人学号的截图替换 |

**（3）向fund表添加约束。**

为fund表的f\_amount列添加大于等于0的约束。

ALTER table fund ADD CONSTRAINT c\_f\_mount CHECK (f\_amount >=0);

**（4）向insurance表添加约束。**

为insurance表的i\_amount列添加大于等于0的约束。

ALTER table insurance ADD CONSTRAINT c\_i\_mount CHECK (i\_amount >=0);

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

**5.查询数据**

在本章的金融数据库实验中，主要目的是为了让读者学习到更深一层的查询操作，让学习者能够更深入的去了解openGauss数据库的复杂操作。

**（1）单表查询。**

* 查询银行卡信息表。

|  |  |
| --- | --- |
| SQL语句  select \* from bank\_card; | SQL语句及执行结果截图显示 |

**（2）条件查询。**

* 查询资产信息中‘可用’的资产数据。

|  |  |
| --- | --- |
| SQL语句  select \* from property where pro\_status='可用'; | SQL语句及执行结果截图显示 |

**（3）聚合查询。**

查询用户表中有多少个用户。

|  |  |
| --- | --- |
| SQL语句  select conut(distinct c\_name) from client; | SQL语句及执行结果截图显示 |

查询银行卡信息表中，储蓄卡和信用卡的个数。

|  |  |
| --- | --- |
| SQL语句  select b\_type,count(\*) from bank\_card group by b\_type; | SQL语句及执行结果截图显示 |

查询保险信息表中，保险金额的平均值。

|  |  |
| --- | --- |
| SQL语句  select avg(i\_amount) from insurance; | SQL语句及执行结果截图显示 |

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。

|  |  |
| --- | --- |
| SQL语句  select i\_name,i\_amount from insurance where i\_amount=(select max(i\_amount) from insurance) or i\_amount=(select min(i\_amount) from insurance); | SQL语句及执行结果截图显示 |

**（4）连接查询（半连接）**

* 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。

|  |  |
| --- | --- |
| SQL语句  select b\_c\_id,c\_name,c\_id\_card from bank\_card,client where b\_c\_id=c\_id; | SQL语句及执行结果截图显示 |

备注：半连接是一种特殊的连接类型，在SQL中没有指定的关键字，通过在WHERE后面使用IN或EXISTS子查询实现。当IN/EXISTS右侧的多行满足子查询的条件时，主查询也只返回一行与EXISTS子查询匹配的行，而不是复制左侧的行。

**（5）连接查询（反连接）。**

* 查询银行卡号不是‘622202130202000001\*’（\*表示未知）的用户的编号，姓名和身份证。

|  |  |
| --- | --- |
| SQL语句  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%'); | SQL语句及执行结果截图显示 |

备注：反连接是一种特殊的连接类型，在SQL中没有指定的关键字，通过在WHERE后面使用 NOT IN或NOT EXISTS子查询实现。返回所有不满足条件的行。这个关系的概念跟半连接相反。

**（6）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。

|  |  |
| --- | --- |
| SQL语句  select i\_name, i\_person from insurance  where i\_amount > (select avg(i\_amount) from insurance); | SQL语句及执行结果截图显示 |

**（7）ORDER BY子句。**

* 按照保额降序查询保险编号大于2的保险名称，保额和适用人群。

|  |  |
| --- | --- |
| SQL语句  select i\_name, i\_amount, i\_person from insurance  where i\_id > 2 order by (i\_amount) desc; | SQL语句及执行结果截图显示 |

**（8）GROUP BY子句。**

* 查询各理财产品信息总数，按照p\_year分组。

|  |  |
| --- | --- |
| SQL语句  select p\_year, count(\*) from finances\_product group by (p\_year); | SQL语句及执行结果截图显示 |

**（9）HAVING子句。**

* 查询保险金额统计数量等于2的适用人群数。

|  |  |
| --- | --- |
| SQL语句  select count(distinct(i\_person)) as 适用人群数, count(i\_amount) as total  from insurance group by (i\_amount) having (total = 2); | SQL语句及执行结果截图显示 |

备注：HAVING子句依附于GROUP BY子句而存在。

**（10）WITH AS子句。**

* 使用WITH AS查询基金信息表。

|  |  |
| --- | --- |
| SQL语句  with temp as (select f\_name, ln(f\_amount) from fund order by f\_manager desc)  select \* from temp; | SQL语句及执行结果截图显示 |

备注：该语句为定义一个SQL片段，该SQL片段会被整个SQL语句用到。

可以使SQL语句的可读性更高。存储SQL片段的表与基本表不同，是一个虚表。数据库不存放对应的定义和数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从存储SQL片段的表中查询出的数据也随之改变。

**6视图**

视图是一个**虚拟表**，是sql的查询结果，其内容由查询定义。对于来自多张关联表的复杂查询，就不得不使用十分复杂的SQL语句进行查询，造成极差的体验感。使用视图之后，可以极大的简化操作，使用视图不需要关心相应表的结构、关联条件等。

**（1）创建视图。**

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。

|  |  |
| --- | --- |
| SQL语句  create view client\_in\_bank as select c\_id, c\_name, c\_id\_card from client where c\_id in (select b\_c\_id from bank\_card); | SQL语句及执行结果截图显示 |

使用视图进行查询。

|  |  |
| --- | --- |
| SQL语句  select \* from client\_in\_bank; | SQL语句及执行结果截图显示 |

**（2）修改视图内容**

修改视图，在原有查询的基础上，过滤出信用卡用户。

|  |  |
| --- | --- |
| SQL语句  create or replace view client\_in\_bank as  select c\_id, c\_name, c\_id\_card from client  where c\_id in (select b\_c\_id from bank\_card where b\_type = '信用卡'); | SQL语句及执行结果截图显示 |

使用视图进行查询。

|  |  |
| --- | --- |
| SQL语句  select \* from client\_in\_bank; | SQL语句及执行结果截图显示 |

**（3）修改视图名称。**

|  |  |
| --- | --- |
| SQL语句  alter view client\_in\_bank rename to v\_client; | SQL语句及执行结果截图显示 |

**（4）删除视图。**

将v\_client视图删除，删除视图不影响基表。

|  |  |
| --- | --- |
| SQL语句  drop view v\_client; | SQL语句及执行结果截图显示 |

**7索引**

**（1）创建索引。**

* 在普通表property上创建索引。

|  |  |
| --- | --- |
| SQL语句  create index time\_index on property (pro\_purchase\_time); | SQL语句及执行结果截图显示 |

**（2）重命名索引。**

* 在普通表property上重建及重命名索引。

重建索引。

|  |  |
| --- | --- |
| SQL语句  reindex index time\_index; | SQL语句及执行结果截图显示 |

重命名索引。

|  |  |
| --- | --- |
| SQL语句  alter index time\_index rename to idx\_property\_temp; | SQL语句及执行结果截图显示 |

**删除索引。**

* 删除索引idx\_property\_temp。

|  |  |
| --- | --- |
| SQL语句  drop index idx\_property\_temp; | SQL语句及执行结果截图显示 |

**8数据的修改和删除**

**（1）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。

查看表数据。

|  |  |
| --- | --- |
| SQL语句  select \* from bank\_card; | SQL语句及执行结果截图显示 |

开始更新数据：

|  |  |
| --- | --- |
| SQL语句  update bank\_card set b\_type = '已修改' where b\_c\_id < 10 and b\_c\_id in (select c\_id from client); | SQL语句及执行结果截图显示 |

重新查询数据情况。

|  |  |
| --- | --- |
| SQL语句  select \* from bank\_card; | SQL语句及执行结果截图显示 |

**（2）删除指定数据。**

删除基金信息表中编号小于3的行。

删除前查询结果。

|  |  |
| --- | --- |
| SQL语句  select \* from fund; | SQL语句及执行结果截图 |

开始删除数据：

|  |  |
| --- | --- |
| SQL语句  delete from fund where f\_id<3; | SQL语句及执行结果截图显示 |

查询删除结果。

|  |  |
| --- | --- |
| SQL语句  select \* from fund; | SQL语句及执行结果截图显示 |

**9新用户的创建和授权**

在本章中，假设C银行的某新员工想要在自己的用户下去访问sys用户下的金融数据库，则该员工需要向sys申请添加相关权限，具体操作如下：

**（1）连接数据库后，进入SQL命令界面。创建用户user311900xxxx（注意，每位同学根据自己学号修改的后四位），密码为Gauss#3demo。**

CREATE USER user311900xxxx IDENTIFIED BY 'Gauss#3demo';

**（2）给用户**user311900xxxx**授予f311900xxxx数据库下bank\_card表的查询和插入权限，并将SCHEMA的权限也授予**user311900xxxx**用户。**

GRANT SELECT,INSERT ON f311900xxxx.bank\_card TO user311900xxxx;

GRANT ALL ON SCHEMA f311900xxxx to user311900xxxx;

**（3）退出数据库：**

F311900xxxx=#**\q**

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

**10新用户连接数据库**

**（1）在gsql登录数据库，使用新用户连接。**

使用操作系统omm用户在新的窗口登陆并执行以下命令，并输入对应的密码。

gsql -d f311900xxxx -U user311900xxxx -p 26000 -r

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示    注意用带个人学号的截图替换 |

**（2）访问**f311900xxxx**数据库的表bank\_card。**

|  |  |
| --- | --- |
| SQL语句  select \* from bank\_card; | SQL语句及执行结果截图显示 |

**退出数据库：**

f311900xxxx=#**\q**

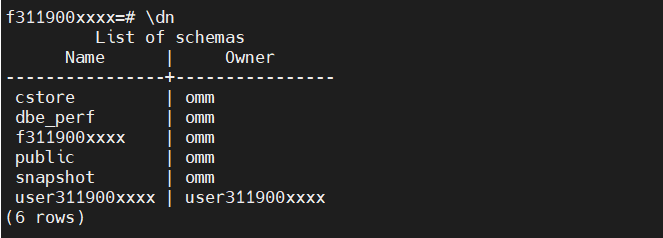
**11删除Schema**

**（1）使用管理员用户登陆**f311900xxxx**数据库。**

使用操作系统omm用户使用gsql，新建session。

gsql -d f311900xxxx -p 26000

1. **使用“\dn”查看数据库下的schema。**



**（3）设置默认查询为**f311900xxxx**。**

set search\_path to f311900xxxx;

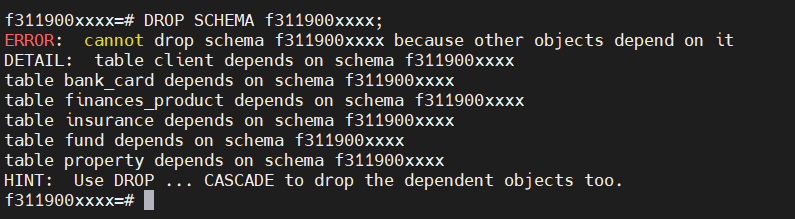
**（4）使用“\dt”命令可以看到在**f311900xxxx**中的对象。**

|  |  |
| --- | --- |
| SQL语句 | SQL语句及执行结果截图显示 |

**（5）使用DROP SCHEMA 命令删除**f311900xxxx**会有报错，因为**f311900xxxx**下存在对象。**

DROP SCHEMA f311900xxxx;

报错如下：



**（6）使用DROP SCHEMA…..CASCADE删除，会将**f311900xxxx**连同下的对象一起删除。**

DROP SCHEMA f311900xxxx CASCADE; /\*注意，此操作先不要做，否则会影响后面的数据库连接\*/

结果如下：

NOTICE: drop cascades to 6 other objects

DETAIL: drop cascades to table client

drop cascades to table bank\_card

drop cascades to table insurance

drop cascades to table fund

drop cascades to table property

drop cascades to table finances\_product

DROP SCHEMA

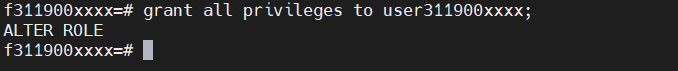
**（7）使用“\dt”命令可以看到在**f311900xxxx**和public中的对象，对象已删除。**

\dt

No relations found.

**12使用Java程序连接数据库并进行查询**

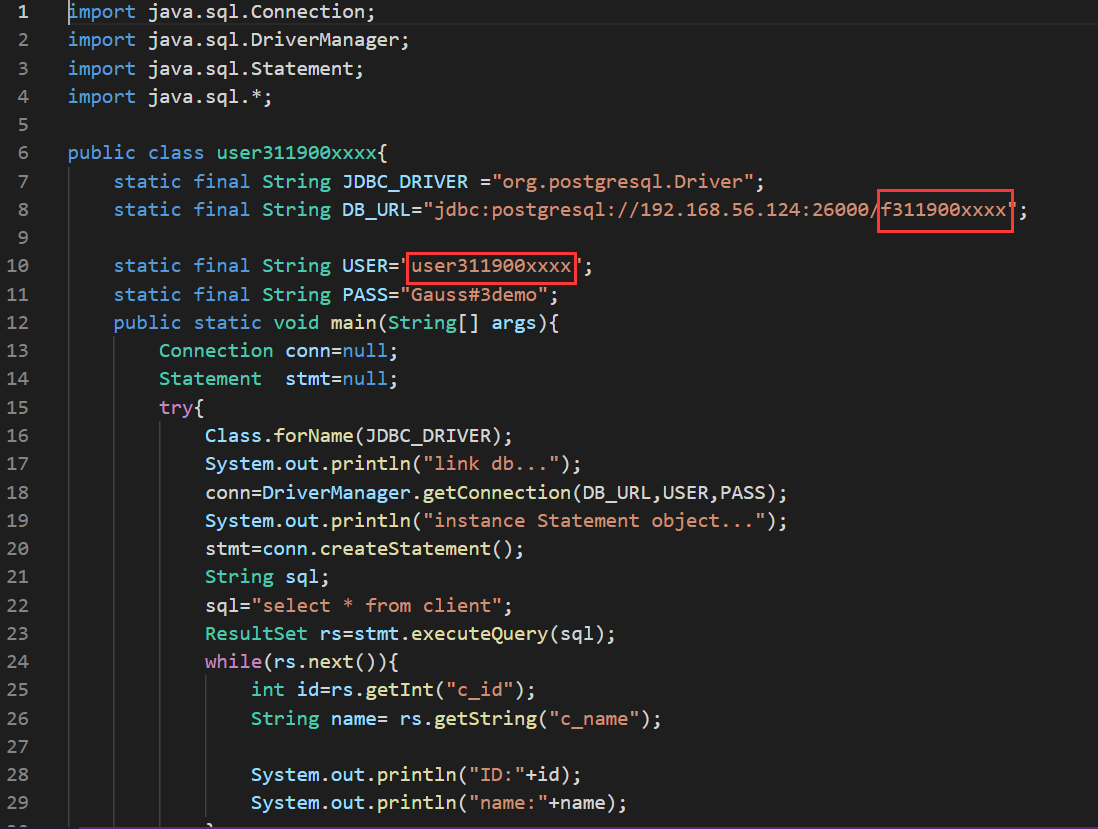
**（1）给用户user**311900xxxx**授予数据库操作权限。**



1. **下载JDBC的驱动包postgresql.jar。**

下载地址：https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.1/x86\_openEuler/openGauss-1.0.1-JDBC.tar.gz，解压缩后postgresql.jar文件放到java项目文件夹下。

1. **创建java连接源代码文件user311900xxxx.java**





1. **编译运行**

|  |  |
| --- | --- |
| 执行命令 | 执行结果截图显示        注意用带个人学号的截图替换 |

**六、实验总结：**

**七、问题与思考：**

1. **openGauss数据库的特点；**
2. **如何应用openGauss数据库解决实际问题；**