**2024实验二：openGauss场景综合实验**

**一、系统背景**

某银行为了建设金融管理系统，需要对该系统的数据库进行设计，针对该系统，主要将对象分为客户、银行卡、理财产品、保险、基金和资产。假设该系统存在着以下关系：客户可以办理银行卡，同时客户可以购买不同的银行产品，如资产，理财产品，基金和保险。

**二、系统功能要求**

该系统主要经营各类金融产品（包括理财产品、保险和基金），通过程序实现简单的管理功能。

**功能包括**：客户信息的查询、理财产品的查询、符合人群类型的保险产品查询、指定基金产品的上线下线、购买金融理财产品等。

金融管理系统**关系模型**参考如下：

* 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，所属客户编号。
* 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。
* 资产：客户编号，商品编号，商品类型，商品状态，商品数量，商品收益，购买时间。

**三、实验要求**

1. 按照实验中给定的关系模型，画出E-R 图。
2. 编写建表的SQL语句：
3. 创建数据库，执行刚才编写的建表语句，截图执行命令\d+输出的结果。
4. 对系统功能进行分析与设计，并给出其操作数据库涉及的基本增删改查SQL语句。
5. 请使用JDBC连接数据库实现金融管理系统所支持的功能。

**四、实验实现**

**4.1 E-R 图**

|  |
| --- |
| //仅供参考 |

**4.2 编写建表的SQL语句**

1. **创建金融数据库finance**

切换到omm用户，以omm用户登录数据库主节点。

|  |
| --- |
| Bash su - omm |

|  |
| --- |
| su omm 和su - omm区别在于是否改变当前工作环境，包括配置文件和工作目录 |

首先先启动数据库，启动命令如下：

|  |
| --- |
| SQL gs\_ctl start -D /home/omm/data -Z single\_node -l /home/omm/log/opengauss.log |

|  |
| --- |
| -D 表示用户自定义的数据路径，大家注意替换成自己的data路径 |

使用gsql工具登录连接数据库（注意端口号）

|  |
| --- |
| Bash gsql -d postgres -p 5432 -r |

|  |
| --- |
| 如何查看自己数据库的端口号呢？  找到与opengauss安装目录同级下的/data文件夹，打开其子文件postgresql.conf即可查看port |

创建金融数据库finance

|  |
| --- |
| SQL create database finance encoding 'utf8' template=template0; |

连接数据库finance

|  |
| --- |
| SQL \connect finance |

创建名为finance的schema，并设置finance为当前的schema

将默认搜索路径设为finance

|  |
| --- |
| SQL create schema finance; set search\_path to finance; |

2. **创建客户表**

* 删除表client

|  |
| --- |
| SQL drop table if exists client; |

* 创建表client

|  |
| --- |
| SQL create table client(  c\_id int primary key,  c\_name varchar(100) not null,  c\_mail char(30) unique,  c\_id\_card char(20) unique not null,  c\_phone char(20) unique not null,  c\_password char(20) not null ); |

3. **创建银行卡表**

* 删除银行卡表bank\_card

|  |
| --- |
| SQL drop table if exists bank\_card; |

* 创建银行卡表bank\_card

|  |
| --- |
| SQL create table bank\_card(  b\_number char(30) primary key,  b\_type char(20),  b\_c\_id int not null); |

4. **创建理财产品表**

* 删除理财产品表finances\_product

|  |
| --- |
| SQL drop table if exists finances\_product; |

* 创建理财产品表finances\_product

|  |
| --- |
| SQL create table finances\_product(  p\_name varchar(100) not null,  p\_id int primary key,  p\_description varchar(4000),  p\_amount int,  p\_year int ); |

5. **创建保险表**

* 删除保险表insurance

|  |
| --- |
| SQL drop table if exists insurance; |

* 创建保险表insurance

|  |
| --- |
| SQL create table insurance(  i\_name varchar(100) not null,  i\_id int primary key,  i\_amount int,  i\_person char(20),  i\_year int,  i\_project varchar(200) ); |

6. **创建基金表**

* 删除基金表fund

|  |
| --- |
| SQL drop table if exists fund; |

* 创建基金表fund

|  |
| --- |
| SQL create table fund(  f\_name varchar(100) not null,  f\_id int primary key,  f\_type char(20),  f\_amount int,  f\_risk\_level char(20) not null,  f\_manager int not null ); |

7. **创建资产表**

* 删除资产表property

|  |
| --- |
| SQL drop table if exists property; |

* 创建资产表property

|  |
| --- |
| SQL create table property(  prop\_id int primary key,  prop\_c\_id int not null,  prop\_pif\_id int not null,  prop\_type int not null,  prop\_status char(20),  prop\_quantity int,  prop\_income int,  prop\_purchase\_time date ); |

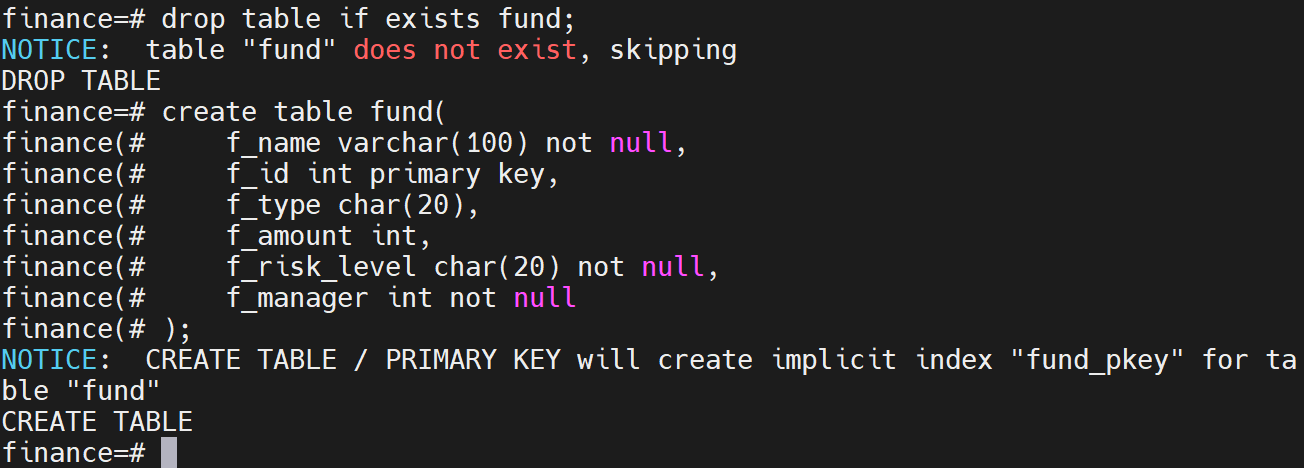
查看数据库中存在的所有表命令：

|  |
| --- |
| Bash \dt |

查看表结构

|  |
| --- |
| Bash \d tablename |

**4.3 执行建表语句截图结果**



**4.4 系统功能设计**

1. **插入表数据**

* 对client表进行数据初始化

插入10条数据

|  |
| --- |
| SQL INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (1,'张一','zhangyi@huawei.com','340211199301010001','18815650001','gaussdb\_001'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (2,'张二','zhanger@huawei.com','340211199301010002','18815650002','gaussdb\_002'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (3,'张三','zhangsan@huawei.com','340211199301010003','18815650003','gaussdb\_003'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (4,'张四','zhangsi@huawei.com','340211199301010004','18815650004','gaussdb\_004'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (5,'张五','zhangwu@huawei.com','340211199301010005','18815650005','gaussdb\_005'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (6,'张六','zhangliu@huawei.com','340211199301010006','18815650006','gaussdb\_006'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (7,'张七','zhangqi@huawei.com','340211199301010007','18815650007','gaussdb\_007'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (8,'张八','zhangba@huawei.com','340211199301010008','18815650008','gaussdb\_008'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (9,'张九','zhangjiu@huawei.com','340211199301010009','18815650009','gaussdb\_009'); INSERT INTO client(c\_id,c\_name,c\_mail,c\_id\_card,c\_phone,c\_password) VALUES (10,'李一','liyi@huawei.com','340211199301010010','18815650010','gaussdb\_010'); |

查询插入结果

|  |
| --- |
| SQL select count(\*) from client; |

* 对bank\_card表进行数据初始化

插10条数据

|  |
| --- |
| SQL INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000001','信用卡',1); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000002','信用卡',3); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000003','信用卡',5); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000004','信用卡',7); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000005','信用卡',9); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000006','信用卡',10); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000007','信用卡',12); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000008','信用卡',14); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000009','信用卡',16); INSERT INTO bank\_card(b\_number,b\_type,b\_c\_id) VALUES ('6222021302020000010','信用卡',18); |

查询插入结果

|  |
| --- |
| SQL select count(\*) from bank\_card; |

* 对finances\_product表进行数据初始化

插4条数据

|  |
| --- |
| SQL INSERT INTO finances\_product(p\_name,p\_id,p\_description,p\_amount,p\_year) VALUES ('债券',1,'以国债、金融债、央行票据、企业债为主要投资方向的银行理财产品。',50000,6); INSERT INTO finances\_product(p\_name,p\_id,p\_description,p\_amount,p\_year) VALUES ('信贷资产',2,'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。',50000,6); INSERT INTO finances\_product(p\_name,p\_id,p\_description,p\_amount,p\_year) VALUES ('股票',3,'与股票挂钩的理财产品。目前市场上主要以港股挂钩居多',50000,6); INSERT INTO finances\_product(p\_name,p\_id,p\_description,p\_amount,p\_year) VALUES ('大宗商品',4,'与大宗商品期货挂钩的理财产品。目前市场上主要以挂钩黄金、石油、农产品的理财产品居多。',50000,6); |

查询插入结果

|  |
| --- |
| SQL select count(\*) from finances\_product; |

* 对insurance表进行数据初始化

插3条数据

|  |
| --- |
| SQL INSERT INTO insurance(i\_name,i\_id,i\_amount,i\_person,i\_year,i\_project) VALUES ('健康保险',1,2000,'老人',30,'平安保险'); INSERT INTO insurance(i\_name,i\_id,i\_amount,i\_person,i\_year,i\_project) VALUES ('人寿保险',2,3000,'老人',30,'平安保险'); INSERT INTO insurance(i\_name,i\_id,i\_amount,i\_person,i\_year,i\_project) VALUES ('意外保险',3,5000,'所有人',30,'平安保险'); |

查询插入结果

|  |
| --- |
| SQL select count(\*) from insurance; |

* 对fund表进行数据初始化

插4条数据

|  |
| --- |
| SQL INSERT INTO fund(f\_name,f\_id,f\_type,f\_amount,f\_risk\_level,f\_manager) VALUES ('股票',1,'股票型',10000,'高',1); INSERT INTO fund(f\_name,f\_id,f\_type,f\_amount,f\_risk\_level,f\_manager) VALUES ('投资',2,'债券型',10000,'中',2); INSERT INTO fund(f\_name,f\_id,f\_type,f\_amount,f\_risk\_level,f\_manager) VALUES ('国债',3,'货币型',10000,'低',3); INSERT INTO fund(f\_name,f\_id,f\_type,f\_amount,f\_risk\_level,f\_manager) VALUES ('沪深300指数',4,'指数型',10000,'中',4); |

查询插入结果

|  |
| --- |
| SQL select count(\*) from fund; |

* 对property表进行数据初始化

插4条数据

|  |
| --- |
| SQL INSERT INTO property(prop\_id,prop\_c\_id,prop\_pif\_id,prop\_type,prop\_status,prop\_quantity,prop\_income,prop\_purchase\_time) VALUES (1,5,1,1,'可用',4,8000,'2018-07-01'); INSERT INTO property(prop\_id,prop\_c\_id,prop\_pif\_id,prop\_type,prop\_status,prop\_quantity,prop\_income,prop\_purchase\_time) VALUES (2,10,2,2,'可用',4,8000,'2018-07-01'); INSERT INTO property(prop\_id,prop\_c\_id,prop\_pif\_id,prop\_type,prop\_status,prop\_quantity,prop\_income,prop\_purchase\_time) VALUES (3,15,3,3,'可用',4,8000,'2018-07-01'); INSERT INTO property(prop\_id,prop\_c\_id,prop\_pif\_id,prop\_type,prop\_status,prop\_quantity,prop\_income,prop\_purchase\_time) VALUES (4,20,4,1,'冻结',4,8000,'2018-07-01'); |

查询插入结果

|  |
| --- |
| SQL select count(\*) from property; |

2. **添加约束**

* 针对表中金额的属性，增加大于0的约束条件

现实生活中，金额不会存在负数。因此，针对表中金额属性，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。

|  |
| --- |
| SQL alter table finances\_product add constraint c\_p\_mount check(p\_amount >= 0); |

* 插入一条金额小于0的记录

|  |
| --- |
| SQL 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"。

* 向fund表添加约束

为fund表的f\_amount列添加大于等于0的约束。

|  |
| --- |
| SQL alter table fund add constraint c\_f\_mount check(f\_amount >= 0); |

* 向insurance表添加约束

为insurance表的i\_amount列添加大于等于0的约束。

|  |
| --- |
| SQL alter table insurance add constraint c\_i\_amount check(i\_amount >= 0); |

3. **查询数据**

目的：掌握基本通用查询以及更深入了解openGauss数据库的复杂查询操作。

请用SQL语句实现下面题目：

1. 查询银行卡信息表。
2. 查询资产信息中‘可用’的资产数据。
3. 查询银行卡信息表中，储蓄卡和信用卡的个数。
4. 查询保险信息表中，保险金额的平均值。

4. **数据的修改和删除**

* 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.

* 删除指定数据

删除基金信息表中编号小于3的行。

**五、jdbc连接实现finance-CRUD功能**

请从实现的功能中任选之一使用jdbc实现数据库连接，并在实验报告中附上java代码以及贴出结果图。

例：测试查询保险信息表中，保险金额的平均值。

|  |
| --- |
| SQL select avg(i\_amount) from insurance; |

结果如下：

