《数据库》

GaussDB(for openGauss)

场景综合实验作业设计



华为技术有限公司

目录

[1 作业设计 2](#_Toc69215261)

[1.1 系统背景 2](#_Toc69215262)

[1.2 系统功能要求 2](#_Toc69215263)

[1.3 实验技术要求 3](#_Toc69215264)

[1.4 作业要求 3](#_Toc69215265)

[1.4.1 分析系统功能，提交相应的E-R图，完成对应的关系模型(10分) 3](#_Toc69215266)

[1.4.2 完成具体的物理设计，编写数据字典(12分) 4](#_Toc69215267)

[1.4.3 创建数据库表(18分) 4](#_Toc69215268)

[1.4.4 模拟场景，对表中的数据进行查询操作。(20分) 5](#_Toc69215269)

[1.4.5 创建视图和索引（9分） 6](#_Toc69215270)

[1.4.6 数据修改和删除（5分） 6](#_Toc69215271)

[1.4.7 创建新用户（6分） 6](#_Toc69215272)

[1.4.8 使用JDBC连接GaussDB(for openGauss)数据库（10分） 6](#_Toc69215273)

[1.5 实验目的 7](#_Toc69215274)

# 作业设计

## 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。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，如理财产品，基金和保险。购买后的银行产品统称为客户的资产。

## 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：

* 客户在银行办理银行卡或者购买各类金融产品时，需要提供客户姓名、客户身份证、客户联系方式，由于客户可以登录系统进行资产查询，所以需要设置客户登录密码，并且部分客户会提供电子邮箱接收电子账单等资料。
* 客户可以办理多张银行卡，不仅可以办理储蓄卡，如果符合要求还可以办理信用卡，客户办理成功后获得银行卡的卡号。
* 银行提供多种理财产品，客户根据理财产品的信息（包括产品编号，产品名称，产品描述）购买理财产品，一个客户可以购买多种理财产品，不同的理财产品有不同的购买金额和理财年限。
* 银行提供多种保险，客户根据保险的信息（包括保险编号，保险名称，适用人群，保险项目）购买理财产品，一个客户可以购买多种保险，不同的保险有不同的保险金额和保险年限。
* 银行提供多种基金，客户根据基金的信息（包括基金编号，基金名称，基金类型，风险等级和基金管理者）购买理财产品，一个客户可以购买多种基金，不同的基金有不同的基金金额。
* 客户购买银行产品（理财产品、保险、基金）成为自己的资产后，需要记录这些资产的状态（可用，冻结），每个资产的购买数量，每个资产的收益和购买时间。

## 实验技术要求

掌握数据库基础设计相关理论知识（包括需求分析、概念设计、逻辑设计和物理设计）;

能够安装部署GaussDB(for openGauss)数据库；

掌握数据库基本DDL语法，能够根据物理设计创建对应的数据库对象；

掌握数据库基本DML语法，能够按照业务功能对表中的数据进行增删改操作；

掌握数据库基本DQL语法，能够按照业务需求对表中的数据进行查询操作；

安装Java运行环境；

具备Java语言的基本开发能力，连接GaussDB(for openGauss)数据库。

## 作业要求

### 分析系统功能，提交相应的E-R图，完成对应的关系模型(10分)

任务：

分析系统功能，提交相应的E-R图；(5分)

根据设计完成的E-R图，完成对应的关系模式，包括对象的具体属性描述和对象之间的关系描述，需满足第三范式要求。(5分)

输出：

E-R图。

对象的具体属性描述；

对象之间的关系描述；

示例：

E-R图：以下E-R图仅为示例，表明E-R图的格式。

客户名称

客户身份证

客户编号

银行卡号

N

1

银行卡

办理

客户

E-R图示例

属性描述：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）

对象之间的关系描述：一个客户可以办理多张银行卡。

### 完成具体的物理设计，编写数据字典(12分)

任务：根据逻辑设计模型，进行具体的物理设计（可以根据操作效率或应用效率做一些反范式处理），提交物理模型（包含表名、字段名、约束名等），通过物理设计编写数据字典。（12分）

输出：每个数据表的数据字典文档。

示例：数据字典文档：

Client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 |

### 创建数据库表(18分)

任务：

创建finance数据库作为项目数据库，数据库编码为UTF-8。（2分）

连接finance数据库，创建名为finance的schema，并设置finance为当前的schema。（2分）

在finance模式下完成金融管理系统中所有数据库对象（数据表）的创建，并完成数据的填充。其中客户数据不少于20条，银行卡数据不少于10条，其他数据不少于5条；（6分）

对表添加外键约束，在银行信息表和资产信息表中，都存在每个银行卡必须有一个持卡者、每份资产必须都有一个资产拥有者这样的对应关系。因此针对这种对应关系，创建外键约束。（4分）

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。（4分）

输出：

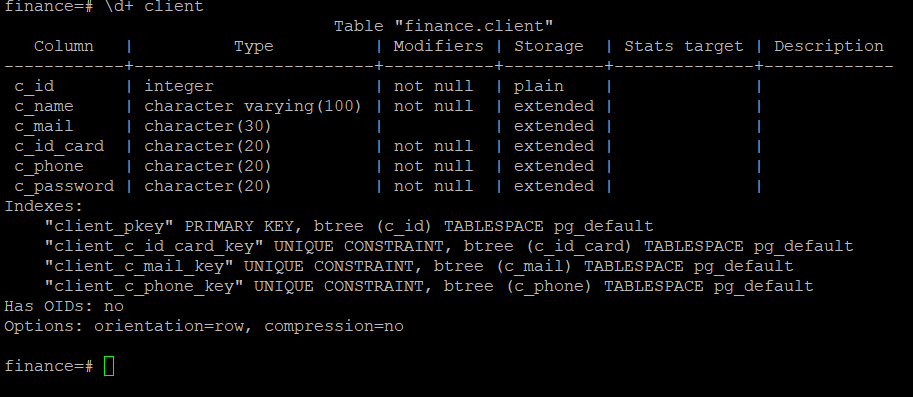
输出查询所有表的数据字典的查询语句和结果，可以截图或者复制黏贴代码。

输出查询所有表的数据的查询语句和结果，可以截图或者复制黏贴代码。

输出所有创建约束的语句和创建结果，可以截图或者复制黏贴代码。

示例：

截图



复制查询语句和结果

finance=# \d+ client

Table "finance.client"

Column | Type | Modifiers | Storage | Stats target | Description

------------+------------------------+-----------+----------+--------------+-------------

c\_id | integer | not null | plain | |

c\_name | character varying(100) | not null | extended | |

c\_mail | character(30) | | extended | |

c\_id\_card | character(20) | not null | extended | |

c\_phone | character(20) | not null | extended | |

c\_password | character(20) | not null | extended | |

Indexes:

"client\_pkey" PRIMARY KEY, btree (c\_id) TABLESPACE pg\_default

"client\_c\_id\_card\_key" UNIQUE CONSTRAINT, btree (c\_id\_card) TABLESPACE pg\_default

"client\_c\_mail\_key" UNIQUE CONSTRAINT, btree (c\_mail) TABLESPACE pg\_default

"client\_c\_phone\_key" UNIQUE CONSTRAINT, btree (c\_phone) TABLESPACE pg\_default

Has OIDs: no

Options: orientation=row, compression=no

复制SQL语句

select count(\*) from client;

### 模拟场景，对表中的数据进行查询操作。(20分)

任务：

至少完成1条单表查询和1条表查询。（4分）

至少完成两条聚合查询，例如查询用户表中有多少个用户；查询保险信息表中，保险金额的平均值等。（4分）

* 至少完成3条连接查询，例如：半连接、反连接、多表查询和子查询。（6分）
* 至少完成1条ORDER BY查询。（2分）
* 至少完成1条GROUP BY……HAVING查询。（4分）

输出：每条查询语句和查询结果，具体格式见1.4.4的示例。

### 创建视图和索引（9分）

任务：

视图：创建一个视图；修改视图；使用视图进行查询；重命名视图；删除视图。（5分）

* 索引：创建索引；重建索引；重命名索引；删除索引。（4分）

输出：每条DDL语句和结果，具体格式见1.4.4的示例。

### 数据修改和删除（5分）

任务：

* 修改数据：至少修改2个不同的数据。（3分）
* 删除指定数据：至少删除2个不同数据表的数据。（2分）

输出：每条DML语句和结果，具体格式见1.4.4的示例

### 创建新用户（6分）

任务：

新用户的创建和授权：创建用户dbuser，密码为Gauss#3demo；给用户dbuser授予finance数据库下银行卡信息表的查询和插入权限，并将finance模式的权限也授予dbuser用户。（3分）

新用户连接数据库：使用新用户连接finance数据库；访问finance数据库的银行卡信息表。（2分）

删除finance模式。（1分）

输出：每条SQL语句和结果，具体格式见1.4.4的示例。

### 使用JDBC连接GaussDB(for openGauss)数据库（10分）

任务：

创建测试数据库finance；创建名为root的schema，并设置root为当前的schema；创建测试表websites（id,name,url），数据为（'1', 'openGauss', 'https://opengauss.org/zh/')，('2', '华为云', 'https://www.huaweicloud.com/'), ('3', 'openEuler', 'https://openeuler.org/zh/'), ('4', '华为support中心', 'https://support.huaweicloud.com/')。（3分）

连接GaussDB(for openGauss)并执行java代码，查询测试表websites的所有数据，并显示在Java控制台。（7分）

输出：每一步的操作和操作结果，具体格式见1.4.4的示例。

## 实验目的

通过对具体的金融场景下的实验，使学生掌握数据库设计基础，并通过实践，将设计的模型转化为具体的对象；

通过对场景的实现，使学生能够掌握基础的SQL语言的应用；

通过对场景的实现，使学生掌握数据库中对象的作用；

通过使用GaussDB(for openGauss)，熟悉并掌握GaussDB(for openGauss)数据库的使用。