

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS2003班 |
| 学 号： | U202015375 |
| 姓 名： | 汪宇飞 |
| 指导教师： | 瞿彬彬 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2022 年12 月 31日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

|  |  |
| --- | --- |
| 总分 |  |

目 录

[1 课程任务概述 1](#_Toc123561318)

[2 任务实施过程与分析 2](#_Toc123561319)

[2.1 数据库、表与完整性约束的定义(Create) 2](#_Toc123561320)

[2.2 表结构与完整性约束的修改(ALTER) 3](#_Toc123561321)

[2.3 数据查询(Select)之一 5](#_Toc123561322)

[2.4 数据查询(Select)之二 13](#_Toc123561323)

[2.5 数据的插入、修改与删除(Insert,Update,Delete) 13](#_Toc123561324)

[2.6 视图 15](#_Toc123561325)

[2.7 存储过程与事务 15](#_Toc123561326)

[2.8 触发器 17](#_Toc123561327)

[2.9 用户自定义函数 18](#_Toc123561328)

[2.10 安全性控制 19](#_Toc123561329)

[2.11 并发控制与事务的隔离级别 21](#_Toc123561330)

[2.12 备份+日志：介质故障与数据库恢复 21](#_Toc123561331)

[2.13数据库设计与实现 21](#_Toc123561332)

[2.14 数据库应用开发(JAVA篇) 22](#_Toc123561333)

[2.15数据库的索引B+树实现 27](#_Toc123561334)

[3 课程总结 28](#_Toc123561335)

# 1 课程任务概述

“数据库系统原理实践”是配合“数据库系统原理”课程独立开设的实践课，注重理论与实践相结合。本课程以MySQL为例，系统性地设计了一系列的实训任务，内容涉及以下几个部分：

1. 数据库、表、索引、视图、约束、存储过程、函数、触发器、游标等数据对象的管理与编程；
2. 数据查询，数据插入、删除与修改等数据处理相关任务；
3. 数据库的安全性控制，完整性控制，恢复机制，并发控制机制等系统内核的实验；
4. 数据库的设计与实现；
5. 数据库应用系统的开发(JAVA 篇)。

课程依托头歌实践教学平台，实践课程url见相关课堂教师发布链接及其邀请码。实验环境为Linux操作系统下的MySQL 8.0.28（主要为8.028版本，部分关卡使用8.022版本，使用中基本无差别）。在数据库应用开发环节，使用JAVA 1.8。

# 2 任务实施过程与分析

## 2.1 数据库、表与完整性约束的定义(Create)

本小节共6个任务，要求参考所给出的指引实现连接数据库、创建数据库、创建表和创建表的主码约束、外码约束、CHECK约束、DEFAULT约束以及UNIQUE约束。

### 2.1.1 创建数据库

本关任务：创建数据库。

按照指引，使用create database语句创建用于2022年北京冬奥会信息系统的数据库:beijing2022。，代码如下所示。

1. create database beijing2022;

### 2.1.2 创建表与表的主码约束

本关任务：在指定的数据库中创建一个表，并为表指定主码。

按照指引，使用create table语句在指定的数据库TestDb中创建表t\_emp，并在代码第4行为主码id添加primary key的约束。

1. create database TestDb;
2. use TestDb;
3. create table t\_emp(
4. id int primary key,
5. name varchar(32),
6. deptId int,
7. salary float
8. );

### 2.1.3 创建外码约束

本关任务：创建外码约束（参照完整性约束）。

按照指引，使用create table语句在指定的数据库MyDb中创建表dept和表staff，并在代码第14行为表staff添加外码约束

1. create database MyDb;
2. use MyDb;
3. create table dept(
4. deptNo int primary key,
5. deptName varchar(32)
6. );
7. create table staff(
8. staffNo int primary key,
9. staffName varchar(32),
10. gender char(1),
11. dob date,
12. salary numeric(8,2),
13. deptNo int,
14. constraint FK\_staff\_deptNo foreign key(deptNo) references dept(deptNo)
15. );

### 2.1.4 CHECK约束

本关任务：为表创建CHECK约束.

按照指引，为表中的brand和price添加CHECK约束，核心代码如下。

1. constraint CK\_products\_brand check(brand in ('A','B')),
2. constraint CK\_products\_price check(price>0)

### 2.1.5 DEFAULT约束

本关任务：创建默认值约束(Default约束),为字段指定默认值。

按照指引，为表中的name和mz添加DEFAULT约束，核心代码如下。

1. name varchar(32) not null,
2. mz char(16) default '汉族'

### 2.1.6 UNIQUE约束

本关任务：创建唯一性约束(UNIQUE约束),以保证字段取值的唯一性。

按照指引，为表中的ID添加UNIQUE约束，核心代码如下。

1. ID char(18) unique

## 2.2 表结构与完整性约束的修改(ALTER)

本小节共4个任务，要求使用alter语句实现表的修改，包括表名的修改、字段的修改、约束的修改等。

### 2.2.1 修改表名

根据实验指引，修改表的语法alter table如下。

1. ALTER TABLE 表名
2. [修改事项 [, 修改事项] ...]

本关任务：修改表的名称。

按照指引，使用alter table语句将表名修改为my\_table，代码如下。

1. alter table your\_table rename to my\_table;

### 2.2.2 添加与删除字段

本关任务：为表添加和删除字段。

按照指引，使用alter table语句删除表orderDetail中的列orderDate、添加列unitPrice，代码如下。

1. alter table orderDetail drop orderDate;
2. alter table orderDetail add unitPrice numeric(10,2);

### 2.2.3 修改字段

本关任务：修改字段。

按照指引，使用alter table语句修改QQ号的数据类型以及修改weixin的列名，代码如下。

1. alter table addressBook modify QQ char(12);
2. alter table addressBook rename column weixin to wechat;

### 2.2.4 添加、删除与修改约束

本关任务：添加、删除与修改约束。

按照指引，使用alter table语句为各表添加、删除以及修改各自相应的约束，代码如下。

1. *#(1) 为表Staff添加主码*
2. alter table Staff add primary key(staffNo);
3. *#(2) Dept.mgrStaffNo是外码，对应的主码是Staff.staffNo,请添加这个外码，名字为FK\_Dept\_mgrStaffNo:*
4. alter table Dept add constraint FK\_Dept\_mgrStaffNo foreign key(mgrStaffNo) references Staff(staffNo);
5. *#(3) Staff.dept是外码，对应的主码是Dept.deptNo. 请添加这个外码，名字为FK\_Staff\_dept:*
6. alter table Staff add constraint FK\_Staff\_dept foreign key(dept) references Dept(deptNo);
7. *#(4) 为表Staff添加check约束，规则为：gender的值只能为F或M；约束名为CK\_Staff\_gender:*
8. alter table Staff add constraint CK\_Staff\_gender check (gender in ('F','M'));
9. *#(5) 为表Dept添加unique约束：deptName不允许重复。约束名为UN\_Dept\_deptName：*
10. alter table Dept add constraint UN\_Dept\_deptName unique (deptName);

## 2.3 数据查询(Select)之一

本小节共19个任务，要求实现各自的查询任务，运用各种查询方法使用一条select查询语句实现每一个关卡的具体要求。

### 2.3.1 金融应用场景介绍，查询客户主要信息

本关任务：熟悉本实训数据库的内容；查询客户的主要信息。

按照指引，使用select语句查询所有的客户的名称、手机号和邮箱信息。查询结果按照客户编号排序，代码如下。

1. select c\_name,c\_phone,c\_mail from client
2. order by c\_id;

### 2.3.2 邮箱为null的用户

本关任务：查询客户表(client)中没有填写邮箱的客户的编号、名称、身份证号、手机号。

按照指引，使用select语句查询，使用where限定邮箱为null的用户，代码如下。

1. select c\_id,c\_name,c\_id\_card,c\_phone
2. from client
3. where c\_mail is null;

### 2.3.3 既买了保险又买了基金的用户

本关任务：查询既买了保险又买了基金的客户的名称和邮箱。

按照指引，使用select语句查询，使用IN的嵌套查询，分别查询买了保险和买了基金的客户的id，然后使用and多条件查询获得交集，从而获得符合要求的客户，代码如下。

1. select c\_name,c\_mail,c\_phone
2. from client
3. where c\_id in(
4. select pro\_c\_id
5. from property
6. where pro\_type=2
7. )and c\_id in(
8. select pro\_c\_id
9. from property
10. where pro\_type=3
11. )
12. order by c\_id;

### 2.3.4 办理了储蓄卡的客户信息

本关任务：查询办理了储蓄卡的客户名称、手机号、银行卡号。

按照指引，使用select语句查询，使用多表连接查询同时在client和bank\_card两个表中进行查询，代码如下。

1. select c\_name,c\_phone,b\_number from client,bank\_card
2. where c\_id=b\_c\_id
3. and b\_type="储蓄卡"
4. order by c\_id;

### 2.3.5 每份金额在30000~50000之间的理财产品

本关任务：查询每份金额在30000～50000之间的理财产品，并对结果进行排序。

按照指引，使用select语句查询，使用between限制金额在30000～50000之间，使用order by来进行结果的排序。

1. select p\_id,p\_amount,p\_year from finances\_product
2. where p\_amount between 30000 and 50000
3. order by p\_amount,p\_year desc;

### 2.3.6 商品收益的众数

本关任务：查询资产表中所有资产记录里商品收益的众数和它出现的次数。

按照指引，使用select语句查询，首先使用子查询用count函数获得所有资产记录里商品收益出现的次数，如果某个商品收益出现的次数大于等于所有出现的次数则为众数，代码如下。

1. select pro\_income,count(\*) as presence
2. from property
3. group by pro\_income
4. having count(\*) >= all(
5. select count(\*) from property
6. group by pro\_income
7. );

### 2.3.7 未购买任何理财产品的武汉居民

本关任务：查询未购买任何理财产品的武汉居民的信息。

按照指引，使用select语句查询，首先使用like ‘4021%’来选择武汉居民，然后使用not exist谓词选择没有购买任何理财产品的用户，代码如下。

1. select c\_name,c\_phone,c\_mail
2. from client
3. where c\_id\_card like '4201%' and not exists(
4. select \* from property
5. where c\_id=pro\_c\_id and pro\_type=1
6. )order by c\_id;

### 2.3.8 持有两张信用卡的用户

本关任务：查询在本行持有两张及以上信用卡的客户信息。

按照指引，使用select语句查询，首先在子查询中获得每个用户信用卡的数量，然后通过该信用卡数量获得持有两张及以上信用卡的客户信息，随后排序即可，代码如下。

1. select c\_name,c\_id\_card,c\_phone
2. from client,(
3. select b\_c\_id,count(b\_number)
4. from bank\_card
5. where b\_type="信用卡"
6. group by b\_c\_id
7. )as a(b\_c\_id,b\_num)
8. where c\_id=b\_c\_id and b\_num>=2
9. order by c\_id;

### 2.3.9 购买了货币型基金的客户信息

本关任务：查询购买了货币型基金的客户信息。

按照指引，使用select语句查询，首先在子查询中使用多表查询获得购买了货币型基金的客户id，然后根据id查询客户信息即可，代码如下。

1. select c\_name,c\_phone,c\_mail
2. from client
3. where c\_id in(
4. select pro\_c\_id
5. from property,fund
6. where pro\_pif\_id=f\_id
7. and pro\_c\_id=c\_id
8. and pro\_type=3
9. and f\_type="货币型"
10. )
11. order by c\_id;

### 2.3.10 投资总收益前三名的客户

本关任务：查询投资总收益前三名的客户。

按照指引，使用select语句查询，首先将client表和property表根据用户id进行连接，选择status为可用的元组按照id进行分组统计，使用sum函数进行计算收益总和并排序，使用limit 3获得前三名的用户，代码如下。

1. select c\_name,c\_id\_card,SUM(pro\_income) as total\_income
2. from client,property
3. where client.c\_id = property.pro\_c\_id and property.pro\_status like '可用'
4. group by c\_name,c\_id\_card
5. order by total\_income desc limit 3;

### 2.3.11 黄姓客户持卡数量

本关任务：查询黄姓用户的编号、名称、办理的银行卡的数量。

按照指引，使用select语句查询，首先使用子查询分组统计每一个用户id所拥有的银行卡数量，然后将client表和子查询获得的表根据用户id进行左外连接，并限定用户名为黄姓，使用ifnull函数对于银行卡数量为0的情况进行处理，最后进行排序，代码如下。

1. select c\_id,c\_name,ifnull(number\_of\_cards,0) number\_of\_cards
2. from client left outer join(
3. select b\_c\_id,count(b\_number) as number\_of\_cards
4. from bank\_card
5. group by b\_c\_id
6. )as a(b\_c\_id,number\_of\_cards)
7. on c\_id=b\_c\_id where c\_name like "黄%"
8. order by number\_of\_cards desc,c\_id;

### 2.3.12 客户理财、保险与基金投资总额

本关任务：查询客户理财、保险、基金投资金额的总和，并排序。

按照指引，使用select语句查询，首先需要分别计算理财、保险和基金投资的数额，将property表分别和finance\_product表、insurance表和fund表根据id进行左外连接，使用sum函数计算各自的数额；然后使用union all将这三个查询结果合并获得派生表；将client表与派生表进行左外连接，使用ifnull函数处理总额为0的情况，最后进行排序，代码如下。

1. select c\_name,c\_id\_card,ifnull(sum(total),0) as total\_amount
2. from client left outer join (
3. select pro\_c\_id,sum(pro\_quantity\*p\_amount) as total
4. from property left outer join finances\_product on(pro\_pif\_id = p\_id)
5. where pro\_type = 1
6. group by pro\_c\_id
7. union all
8. select pro\_c\_id,sum(pro\_quantity\*i\_amount) as total
9. from property left outer join insurance on(pro\_pif\_id = i\_id)
10. where pro\_type = 2
11. group by pro\_c\_id
12. union all
13. select pro\_c\_id,sum(pro\_quantity\*f\_amount) as total
14. from property left outer join fund on(pro\_pif\_id = f\_id)
15. where pro\_type = 3
16. group by pro\_c\_id
17. ) as P on (c\_id = pro\_c\_id)
18. group by c\_id
19. order by total\_amount desc;

### 2.3.13 客户总资产

本关任务：查询客户在本行的总资产。

按照指引，使用select语句查询，客户总资产为储蓄卡总余额，投资总额，投资总收益的和，再扣除信用卡透支的总金额。分别查询储蓄卡的总额、信用卡的总额以及投资收益总额，将client表基于客户id左外连接查询所得的三个表，使用ifnull(bc\_dep\_property,0)-ifnull(bc\_cre\_property,0)+ifnull(total\_income,0)+ ifnull(total\_amount,0)计算客户总资产，具体代码如下。

1. select c\_id,c\_name,ifnull(bc\_dep\_property,0)-ifnull(bc\_cre\_property,0)+ifnull(total\_income,0)+ifnull(total\_amount,0) as total\_property
2. from client left outer join (
3. select b\_c\_id as bc\_dep\_id,sum(b\_balance) as bc\_dep\_property
4. from bank\_card
5. where b\_type like '储蓄卡'
6. group by b\_c\_id
7. ) as dep on(bc\_dep\_id = c\_id)
8. left outer join (
9. select b\_c\_id as bc\_cre\_id,sum(b\_balance) as bc\_cre\_property
10. from bank\_card
11. where b\_type like '信用卡'
12. group by b\_c\_id
13. ) as cre on(bc\_cre\_id = c\_id)
14. left outer join (
15. select pro\_c\_id as pro\_shouxi\_id,sum(pro\_income) as total\_income
16. from property
17. group by pro\_c\_id
18. ) as shouyi on(pro\_shouxi\_id = c\_id)
19. left outer join (
20. select c\_id as pro\_amount\_id,ifnull(sum(total),0) as total\_amount
21. from client left outer join (
22. select pro\_c\_id,sum(pro\_quantity\*p\_amount) as total
23. from property left outer join finances\_product on(pro\_pif\_id = p\_id)
24. where pro\_type = 1
25. group by pro\_c\_id
26. union all
27. select pro\_c\_id,sum(pro\_quantity\*i\_amount) as total
28. from property left outer join insurance on(pro\_pif\_id = i\_id)
29. where pro\_type = 2
30. group by pro\_c\_id
31. union all
32. select pro\_c\_id,sum(pro\_quantity\*f\_amount) as total
33. from property left outer join fund on(pro\_pif\_id = f\_id)
34. where pro\_type = 3
35. group by pro\_c\_id
36. ) as P on (c\_id = pro\_c\_id)
37. group by c\_id
38. ) as amount on(pro\_amount\_id = c\_id)
39. order by c\_id;

### 2.3.14 第N高问题

该关卡任务已完成，实验情况本报告略过。

### 2.3.15 基金收益两种方式排名

本关任务：对客户基金投资收益实现两种方式的排名次。

按照指引，使用select语句查询，定义相关变量currank、rankcount和rank\_revenue进行排序的实现，具体代码如下。

1. *-- (1) 基金总收益排名(名次不连续)*
2. select pro\_c\_id,total\_revenue,myrank as 'rank'
3. from (
4. select pro\_c\_id,total\_revenue,@rankcount := @rankcount + 1,if(@rank\_revenue = total\_revenue, @currank, @currank := @rankcount) as myrank,@rank\_revenue := total\_revenue
5. from (
6. select pro\_c\_id,sum(pro\_income) as total\_revenue
7. from property
8. where pro\_type = 3
9. group by pro\_c\_id
10. order by total\_revenue desc,pro\_c\_id
11. ) as i(pro\_c\_id,total\_revenue),(select @currank := 0,@rankcount := 0,@rank\_revenue := null) as r
12. ) as temp;
13. *-- (2) 基金总收益排名(名次连续)*
14. select pro\_c\_id,total\_revenue,myrank as 'rank'
15. from (
16. select pro\_c\_id,total\_revenue,if(@rank\_revenue = total\_revenue, @currank, @currank := @currank + 1) as myrank,@rank\_revenue := total\_revenue
17. from (
18. select pro\_c\_id,sum(pro\_income) as total\_revenue
19. from property
20. where pro\_type = 3
21. group by pro\_c\_id
22. order by total\_revenue desc,pro\_c\_id
23. ) as i(pro\_c\_id,total\_revenue),(select @currank := 0,@rank\_revenue := null) as r
24. ) as temp;

### 2.3.16 持有完全相同基金组合的客户

本关任务：查询持有完全相同基金组合的客户。

按照指引，使用select语句查询，对基金组合进行两次查询，分别为p1和p2，对于p1和p2中基金组合相同但用户id不同的，即是符合要求的客户，具体代码如下所示。

1. select pro\_c\_id1 as c\_id1,pro\_c\_id2 as c\_id2
2. from(
3. select pro\_c\_id,group\_concat(pro\_pif\_id order by pro\_pif\_id SEPARATOR ' ') as pifset
4. from property
5. where pro\_type = 3
6. group by pro\_c\_id
7. ) as p1(pro\_c\_id1,pifset1),
8. (
9. select pro\_c\_id,group\_concat(pro\_pif\_id order by pro\_pif\_id SEPARATOR ' ') as pifset
10. from property
11. where pro\_type = 3
12. group by pro\_c\_id
13. ) as p2(pro\_c\_id2,pifset2)
14. where p1.pifset1 = p2.pifset2 and p1.pro\_c\_id1 < p2.pro\_c\_id2
15. order by pro\_c\_id1;

### 2.3.17 购买基金的高峰期

该关卡任务已完成，实验情况本报告略过。

### 2.3.18 至少有一张信用卡余额超过5000元的客户信用卡总余额

本关任务：至少有一张信用卡余额超过5000元的客户信用卡总金额。

按照指引，使用select语句查询，使用子查询获得余额超过5000的信用卡的卡号，使用多表查询获得拥有这种卡的客户的信用卡总余额，代码如下。

1. select b\_c\_id,sum(b\_balance) as credit\_card\_amount
2. from bank\_card,(
3. select b\_c\_id
4. from bank\_card
5. where b\_type = '信用卡' and b\_balance >= 5000
6. order by b\_c\_id
7. ) as id(sc\_id)
8. where b\_type = '信用卡' and bank\_card.b\_c\_id = id.sc\_id
9. group by b\_c\_id
10. order by b\_c\_id;

### 2.3.19 以日历表格式显示每日基金购买总金额

该关卡任务已完成，实验情况本报告略过。

## 2.4 数据查询(Select)之二

本小节共6个任务，是在2.3的基础之上另外再添加的6个查询任务。

### 2.4.1 查询销售总额前三的理财产品

该关卡任务已完成，实验情况本报告略过。

### 2.4.2 投资积极且偏好理财类产品的客户

该任务关卡跳过。

### 2.4.3 查询购买了所有畅销理财产品的客户

该关卡任务已完成，实验情况本报告略过。

### 2.4.4 查找相似的理财产品

该关卡任务已完成，实验情况本报告略过。

### 2.4.5 查询任意两个客户的相同理财产品数

该关卡任务已完成，实验情况本报告略过。

### 2.4.6 查找相似的理财用户

该关卡任务已完成，实验情况本报告略过。

## 2.5 数据的插入、修改与删除(Insert,Update,Delete)

本小节共6个任务，要求使用insert、update和delete语句进行数据的插入、修改和删除操作。

### 2.5.1 插入多条完整的客户信息

本关任务：向客户表client插入数据。

使用insert语句将客户信息进行插入，代码如下。

1. insert into client
2. values('1','林惠雯','960323053@qq.com','411014196712130323','15609032348','Mop5UPkl');
3. insert into client
4. values('2', '吴婉瑜','1613230826@gmail.com','420152196802131323','17605132307','QUTPhxgVNlXtMxN');
5. insert into client
6. values('3','蔡贞仪','252323341@foxmail.com','160347199005222323','17763232321','Bwe3gyhEErJ7');

### 2.5.2 插入不完整的客户信息

本关任务：向客户表client插入一条数据不全的记录。

使用insert语句将不完整的客户信息插入，代码如下。

1. insert into client(c\_id,c\_name,c\_phone,c\_id\_card,c\_password)
2. values('33','蔡依婷','18820762130','350972199204227621','MKwEuc1sc6');

### 2.5.3 批量插入数据

本关任务：向客户表client批量插入数据。

使用insert语句将全部客户信息进行插入，代码如下。

1. insert into client
2. select \* from new\_client;

### 2.5.4 删除没有银行卡的客户信息

本关任务：删除在本行没有银行卡的客户信息。

使用delete语句将client中没有银行卡的客户信息进行删除，代码如下。

1. delete from client
2. where c\_id not in(
3. select b\_c\_id from bank\_card
4. );

### 2.5.5 冻结客户资产

本关任务：冻结客户的投资资产。

使用update语句将客户的状态更新为冻结，代码如下。

1. update property,client
2. set pro\_status='冻结'
3. where pro\_c\_id=c\_id and c\_phone='13686431238';

### 2.5.6 连接更新

本关任务：根据客户表的内容修改资产表的内容。

使用update语句对property表和client表进行更新，代码如下。

1. update property,client
2. set pro\_id\_card=c\_id\_card
3. where pro\_c\_id=c\_id;

## 2.6 视图

本小节共2个任务，要求实现视图的创建以及基于视图的查询，

### 2.6.1 创建所有保险资产的详细记录视图

本关任务：创建所有保险资产的详细记录视图。

使用create view语句，将查询获得的结果建立视图，代码如下。

1. create view v\_insurance\_detail
2. as select c\_name,c\_id\_card,i\_name,i\_project,pro\_status,pro\_quantity,i\_amount,i\_year,pro\_income,pro\_purchase\_time
3. from client,property,insurance
4. where c\_id=pro\_c\_id and pro\_pif\_id=i\_id and pro\_type=2;

### 2.6.2 基于视图的查询

本关任务：基于视图v\_insurance\_detail查询每位客户保险资产的总额和保险总收益。

使用select语句进行查询，使用sum函数对于总额进行计算，最后排序，代码如下。

1. select c\_name,c\_id\_card,sum(pro\_quantity\*i\_amount) as insurance\_total\_amount,sum(pro\_income) as insurance\_total\_revenue
2. from v\_insurance\_detail
3. group by c\_name,c\_id\_card
4. order by insurance\_total\_amount desc;

## 2.7 存储过程与事务

本小节共3个任务，要求分别实现使用流程控制语句、游标和事务的存储过程。

### 2.7.1 使用流程控制语句的存储过程

本关任务：创建一个存储过程，向表fibonacci插入斐波拉契数列的前n项。

1. drop procedure if exists sp\_fibonacci;
2. delimiter $$
3. create procedure sp\_fibonacci(in m int)
4. begin
5. declare n,fibn,a,b int;
6. set n = 2,a = 0,b = 1;
7. if m > 0 then
8. insert into fibonacci values(0,0);
9. end if;
10. if m > 1 then
11. insert into fibonacci values(1,1);
12. end if;
13. while m > n do
14. set fibn = a + b;
15. insert into fibonacci values(n,fibn);
16. set a = b;
17. set b = fibn;
18. set n = n + 1;
19. end while;
20. end $$
21. delimiter ;

### 2.7.2 使用游标的存储过程

该任务关卡跳过。

### 2.7.3 使用事务的存储过程

本关任务：编写实现转账功能的存储过程。

1. delimiter $$
2. create procedure sp\_transfer(
3. IN applicant\_id int,
4. IN source\_card\_id char(30),
5. IN receiver\_id int,
6. IN dest\_card\_id char(30),
7. IN amount numeric(10,2),
8. OUT return\_code int)
9. BEGIN
10. set autocommit = off;
11. start transaction;
12. update bank\_card set b\_balance = b\_balance - amount where b\_number = source\_card\_id and b\_c\_id = applicant\_id and b\_type = '储蓄卡';
13. update bank\_card set b\_balance = b\_balance + amount where b\_number = dest\_card\_id and b\_c\_id = receiver\_id and b\_type = '储蓄卡';
14. update bank\_card set b\_balance = b\_balance - amount where b\_number = dest\_card\_id and b\_c\_id = receiver\_id and b\_type = '信用卡';
15. if not exists(select \* from bank\_card where b\_number = source\_card\_id and b\_c\_id = applicant\_id and b\_type = '储蓄卡' and b\_balance >= 0) then
16. set return\_code = 0;
17. rollback;
18. elseif not exists(select \* from bank\_card where b\_number = dest\_card\_id and b\_c\_id = receiver\_id) then
19. set return\_code = 0;
20. rollback;
21. else
22. set return\_code = 1;
23. commit;
24. end if;
25. END$$
26. delimiter ;

## 2.8 触发器

本小节共1个任务，要求实现一个触发器，实现合法性检查，即在表上进行insert、delete、update等操作时触发器被触发执行判断操作是否合法。

### 2.8.1 为投资表property实现业务约束规则-根据投资类别分别引用不同表的主码

本关任务：为表property(资产表)编写一个触发器，以实现以下完整性业务规则：

* 如果pro\_type = 1, 则pro\_pif\_id只能引用finances\_product表的p\_id；
* 如果pro\_type = 2, 则pro\_pif\_id只能引用insurance表的i\_id；
* 如果pro\_type = 3, 则pro\_pif\_id只能引用fund表的f\_id；
* pro\_type不接受(1,2,3)以外的值。

按照指引，参照实验所提供的创建触发器的语句，实现所要求的触发器。对于pro\_type为1、2、3以外的值，触发器进行非法提示，如下列代码8-11所示；对于pro\_type为1、2、3之内某一值，则按照对应的id是否存在进行相应的提示，如下列代码13-26所示。

1. use finance1;
2. drop trigger if exists before\_property\_inserted;
3. delimiter $$
4. CREATE TRIGGER before\_property\_inserted BEFORE INSERT ON property
5. FOR EACH ROW
6. BEGIN
7. declare msg varchar(40);
8. if new.pro\_type <> 1 and new.pro\_type <> 2 and new.pro\_type <> 3 then
9. set msg = concat('type ',new.pro\_type,' is illegal!');
10. signal sqlstate '45000' set message\_text = msg;
11. end if;
12. if new.pro\_type = 1 and not exists(select \* from finances\_product where finances\_product.p\_id = new.pro\_pif\_id) then
13. set msg = concat('finances product #',new.pro\_pif\_id,' not found!');
14. signal sqlstate '45000' set message\_text = msg;
15. end if;
16. if new.pro\_type = 2 and not exists(select \* from insurance where insurance.i\_id = new.pro\_pif\_id) then
17. set msg = concat('insurance #',new.pro\_pif\_id,' not found!');
18. signal sqlstate '45000' set message\_text = msg;
19. end if;
20. if new.pro\_type = 3 and not exists(select \* from fund where fund.f\_id = new.pro\_pif\_id) then
21. set msg = concat('fund #',new.pro\_pif\_id,' not found!');
22. signal sqlstate '45000' set message\_text = msg;
23. end if;
24. END$$
26. delimiter ;

## 2.9 用户自定义函数

本小节共1个任务，要求完成用户自定义函数的实现并使用自定义的函数完成相应功能。

### 2.9.1 创建函数并在语句中使用它

本关任务：编写一个依据客户编号计算其在本金融机构的存储总额的函数,并在SELECT语句使用这个函数。

按照指引，根据create function语句的语法，在给出的待完成的代码中填入用户自定义函数体，如图下代码第9-14行所示，使用select into语句传递参数进行查询，用return语句返回所查询的结果即指定用户的储蓄卡余额之和。

在完成get\_deposit函数的函数体定义之后，调用该函数完成预期功能，如下代码18-21行所示，将所定义的函数运用于select查询语句中，使用where语句进行限定存款总额在100万以上的客户，使用order by语句按照存储总额进行排序。

1. use finance1;
2. set global log\_bin\_trust\_function\_creators=1;
3. drop function IF EXISTS get\_deposit;
4. delimiter $$
5. create function get\_deposit(client\_id int)
6. returns numeric(10,2)
7. begin
8. declare deposit numeric(10,2);
9. select sum(b\_balance) into deposit
10. from bank\_card
11. where b\_type = '储蓄卡' and b\_c\_id = client\_id
12. group by b\_c\_id;
13. return deposit;
14. end$$
15. delimiter ;
16. select c\_id\_card,c\_name,get\_deposit(c\_id) as total\_deposit
17. from client
18. where get\_deposit(c\_id) >= 1000000
19. order by get\_deposit(c\_id) desc;

## 2.10 安全性控制

本小节共2个任务，要求实现对数据库中用户、角色和权限进行相关安全性控制操作。

### 2.10.1 用户和权限

本关任务：在金融应用场景数据库环境中，创建用户，并给用户授予指定的权限。

使用create语句进行角色的创建，使用grant语句进行权限的授予，使用revoke语句收回权限，代码如下。

1. *#(1) 创建用户tom和jerry，初始密码均为'123456'；*
2. create user 'tom' identified by '123456';
3. create user 'jerry' identified by '123456';
4. *#(2) 授予用户tom查询客户的姓名，邮箱和电话的权限,且tom可转授权限；*
5. grant select(c\_name,c\_phone,c\_mail)
6. on client
7. to tom
8. with grant option;
9. *#(3) 授予用户jerry修改银行卡余额的权限；*
10. grant update(b\_balance)
11. on bank\_card
12. to jerry;
13. *#(4) 收回用户Cindy查询银行卡信息的权限。*
14. revoke select
15. on bank\_card
16. from Cindy;

### 2.10.2 用户、角色和权限

本关任务：创建角色，授予角色一组权限，并将角色代表的权限授予指定的一组用户。

使用create语句进行角色的创建，使用grant语句进行权限的授予，代码如下。

1. *# (1) 创建角色client\_manager和fund\_manager；*
2. create role client\_manager;
3. create role fund\_manager;
4. *# (2) 授予client\_manager对client表拥有select,insert,update的权限；*
5. grant select,insert,update
6. on client
7. to client\_manager;
8. *# (3) 授予client\_manager对bank\_card表拥有查询除银行卡余额外的select权限；*
9. grant select(b\_number,b\_type,b\_c\_id)
10. on bank\_card
11. to client\_manager;
12. *# (4) 授予fund\_manager对fund表的select,insert,update权限；*
13. grant select,insert,update
14. on fund
15. to fund\_manager;
16. *# (5) 将client\_manager的权限授予用户tom和jerry；*
17. grant client\_manager
18. to tom,jerry;
19. *# (6) 将fund\_manager权限授予用户Cindy.*
20. grant fund\_manager
21. to Cindy;

## 2.11 并发控制与事务的隔离级别

本小节共6个任务，要求实现并发控制与事务的隔离级别相关内容，包括读脏、不可重复读、幻读、主动加锁保证可重复读、可串行化等内容。

### 2.11.1 并发控制与事务的隔离级别

该任务关卡跳过。

### 2.11.2 读脏

该任务关卡跳过。

### 2.11.3 不可重复读

该任务关卡跳过。

### 2.11.4 幻读

该任务关卡跳过。

### 2.11.5 主动加锁保证可重复读

该任务关卡跳过。

### 2.11.6 可串行化

该任务关卡跳过。

## 2.12 备份+日志：介质故障与数据库恢复

本小节共2个任务，要求实现介质故障与数据库恢复相关内容。

### 2.12.1 备份与恢复

该任务关卡跳过。

### 2.12.2 备份+日志：介质故障的发生与数据库的恢复

该任务关卡跳过。

## 2.13数据库设计与实现

本小节共3个任务，要求实现数据库的设计与实现相关的内容，包括从概念模型到MySQL模型、从需求分析到逻辑模型以及建模工具的使用。除此之外也对制约因素分析与设计和工程师责任及其分析进行了相关的讨论。

### 2.13.1 从概念模型到MySQL实现

该关卡任务已完成，实验情况本报告略过。

### 2.13.2 从需求分析到逻辑模型

该任务关卡跳过。

### 2.13.3 建模工具的使用

该关卡任务已完成，实验情况本报告略过。

### 2.13.4 制约因素分析与设计

在数据库的设计与实现过程中有诸多制约因素需要考虑并进行实现。例如最基础的数据实体完整性、参照完整性以及用户定义完整性，需要基于实际情况设置主码约束、外码约束、CHECK约束等约束，实现有效且全面的完整性约束。

除此之外也需要考虑安全性、隐私性和保密性的问题，对于一些重要信息，如用户个人隐私、身份证号、密码等信息，需要考虑如何对其进行保护，采用加密等方法。

### 2.13.5 工程师责任及其分析

作为工程师，不仅需要使用自身技能完整高效地实现所分配的具体任务，实现数据结构、系统结构等设计，开发所预期的系统，还需要基于所处的环境、基于开发情况承担社会责任。

工程师在开发设计时，需要尽可能考虑到受众用户的具体情况以及开发产品运行的具体环境。

对于开发方而言，工程师最重要的责任是保证开发产品的完善，能够满足开发方的各个需求并能够正确处理开发方的各方面问题。在保证了开发的完整性和正确性的基础上，工程师也应当考虑所开发的开销成本，尽可能使用高效便捷的开发方法，实现开发成本的降低以及开发效率的提升。

对于用户而言，工程师所开发的产品应当是高效、简明、易上手的，应该有利于各个年龄、种族、学历等的用户快速了解并熟悉。除此之外，也应当是健康、安全、遵守法律法规、尊重各民族文化的，工程师所开发的产品应当尽可能排除对用户有害的信息，保护用户个人隐私信息，尊重用户的文化，对于潜在的风险，工程师应当进行排查并修护。

## 2.14 数据库应用开发(JAVA篇)

本小节共7个任务，要求基于JDBC应用实现数据库应用的开发，包含简单的查询、用户登录、新用户添加、银行卡销户、用户修改密码、事务与转账操作和把稀疏表格转换为键值对存储等。

### 2.16.1 JDBC体系结构和简单的查询

本关任务：查询client表中邮箱非空的客户信息，列出客户姓名，邮箱和电话。

根据指引，需要构建JDBC应用。首先导入JDBC包，使用Class.forName方法注册驱动程序，如代码第7行所示；然后使用DriverManager.getConnection方法进行数据库连接，实现URL、用户名和密码的传递，如代码第11行所示；其次需要进行SQL操作，使用connection.createStatement方法创建接口并构造相应的操作字符串，之后使用statement.executeQuery方法进行SQL语句的执行，如代码12-14行所示；将查询所得结果进行next方法遍历，如代码第16行所示；操作全部完成之后使用close方法将创建的对象全部关闭。

具体代码如下。

1. public class Client {
2. public static void main(String[] args) {
3. Connection connection = null;
4. Statement statement = null;
5. ResultSet resultSet = null;
6. try {
7. Class.forName("com.mysql.cj.jdbc.Driver");
8. String URL = "jdbc:mysql://127.0.0.1:3306/finance?useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";
9. String USER = "root";
10. String PASS = "123123";
11. connection = DriverManager.getConnection(URL, USER, PASS);
12. statement = connection.createStatement();
13. String SQL = "select \* from client where c\_mail is not null;";
14. resultSet = statement.executeQuery(SQL);
15. System.out.println("姓名\t邮箱\t\t\t\t电话");
16. while (resultSet.next()) {
17. System.out.print(resultSet.getString("c\_name")+'\t');
18. System.out.print(resultSet.getString("c\_mail")+"\t\t");
19. System.out.println(resultSet.getString("c\_phone"));
20. }
21. } catch (ClassNotFoundException e) {
22. System.out.println("Sorry,can`t find the JDBC Driver!");
23. e.printStackTrace();
24. } catch (SQLException throwables) {
25. throwables.printStackTrace();
26. } finally {
27. try {
28. if (resultSet != null) {resultSet.close();}
29. if (statement != null) {statement.close();}
30. if (connection != null) {connection.close();}
31. } catch (SQLException throwables) {
32. throwables.printStackTrace();
33. }
34. }
35. }
36. }

### 2.16.2 用户登录

本关任务：编写客户登录程序，提示用户输入邮箱和密码，并判断正确性，给出适当的提示信息。

与2.16.1类似，提示用户输入用户名和密码，将用户名用于查询的SQL字符串中，将查询获得的resultSet的密码与输入的密码进行对比，若相同则可以成功登录，否则提示错误，具体代码如下。

1. public class Login {
2. public static void main(String[] args) {
3. Connection connection = null;
4. Statement statement = null;
5. ResultSet resultSet = null;
6. Scanner input = new Scanner(System.in);
7. System.out.print("请输入用户名：");
8. String loginName = input.nextLine();
9. System.out.print("请输入密码：");
10. String loginPass = input.nextLine();
11. try {
12. Class.forName("com.mysql.cj.jdbc.Driver");
13. String userName = "root";
14. String passWord = "123123";
15. String url = "jdbc:mysql://127.0.0.1:3306/finance?useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";
16. connection = DriverManager.getConnection(url, userName, passWord);
17. statement = connection.createStatement();
18. String SQL = "select \* from client where c\_mail = '"+loginName+"' ;";
19. resultSet = statement.executeQuery(SQL);
20. if(resultSet.next()){
21. if(resultSet.getString("c\_password").equals(loginPass))
22. System.out.println("登录成功。");
23. else System.out.println("用户名或密码错误！");
24. }
25. else System.out.println("用户名或密码错误！");
26. } catch (ClassNotFoundException e) {
27. e.printStackTrace();
28. } catch (SQLException throwables) {
29. throwables.printStackTrace();
30. } finally {
31. try {
32. if (resultSet != null) {resultSet.close();}
33. if (statement != null) {statement.close();}
34. if (connection != null) {connection.close();}
35. } catch (SQLException throwables) {
36. throwables.printStackTrace();
37. }
38. }
39. }
40. }

### 2.16.3 添加新用户

本关任务：编程完成向client(客户表)插入记录的方法。

自定义一个insertClient方法，将SQL字符串设置为insert into语句，将各个参数通过setString方法进行设置，从而实现新用户的添加，代码如下所示。

1. public static int insertClient(Connection connection, int c\_id, String c\_name, String c\_mail,String c\_id\_card, String c\_phone, String c\_password){
2. PreparedStatement pstmt = null;
3. int n = 0;
4. try {
5. String sql = "insert into client values (?,?,?,?,?,?);";
6. pstmt = connection.prepareStatement(sql);
7. pstmt.setInt(1,c\_id);
8. pstmt.setString(2,c\_name);
9. pstmt.setString(3,c\_mail);
10. pstmt.setString(4,c\_id\_card);
11. pstmt.setString(5,c\_phone);
12. pstmt.setString(6,c\_password);
13. n = pstmt.executeUpdate();
14. } catch (SQLException throwables) {
15. throwables.printStackTrace();
16. } finally {
17. try {
18. if (pstmt != null) {
19. pstmt.close();
20. }
21. } catch (SQLException throwables) {
22. throwables.printStackTrace();
23. }
24. }
25. return n;
26. }

### 2.16.4 银行卡销户

本关任务：编写一个能删除指定客户编号的指定银行卡号的方法。

自定义一个removeBankCard方法，将SQL字符串设置为delete from语句，将各个参数通过setString方法进行设置，从而实现银行卡的销户，代码如下所示。

1. public static int removeBankCard(Connection connection,int b\_c\_id, String b\_number){
2. PreparedStatement pstmt = null;
3. int n = 0;
4. try {
5. String sql = "delete from bank\_card where b\_c\_id = ? and b\_number = ?;";
6. pstmt = connection.prepareStatement(sql);
7. pstmt.setInt(1,b\_c\_id);
8. pstmt.setString(2,b\_number);
9. n = pstmt.executeUpdate();
10. } catch (SQLException throwables) {
11. throwables.printStackTrace();
12. } finally {
13. try {
14. if (pstmt != null) {
15. pstmt.close();
16. }
17. } catch (SQLException throwables) {
18. throwables.printStackTrace();
19. }
20. }
21. return n;
22. }

### 2.16.5 客户修改密码

该关卡任务已完成，实验情况本报告略过。

### 2.16.6 事务与转账操作

该关卡任务已完成，实验情况本报告略过。

### 2.16.7 把稀疏表格转为键值对存储

该关卡任务已完成，实验情况本报告略过。

## 2.15数据库的索引B+树实现

本小节共5个任务，要求实现数据库B+数相关内容。

### 2.15.1 BPlusTreePage的设计

该任务关卡跳过。

### 2.15.2 BPlusTreeInternalPage的设计

该任务关卡跳过。

### 2.15.3 BPlusTreeLeafPage的设计

该任务关卡跳过。

### 2.15.4 B+树索引：Insert

该任务关卡跳过。

### 2.15.5 B+树索引：Remove

该任务关卡跳过。

# 3 课程总结

本次数据库系统原理实践以理论课知识为基础，以MySQL为例，设计了一系列的数据库相关应用实训任务。完成了数据库、表与完整性约束的定义，表结构与完整性约束的修改，数据查询之一，数据的插入、修改与删除，视图，触发器，用户自定义函数，安全性控制以及数据库应用开发（JAVA篇）的全部关卡。选择性地完成了数据查询之二，存储过程与事务，数据库设计与实现的部分关卡。最终的educoder平台得分为110分，达到了实验预期的分数要求。

完成了数据库、表、索引、视图、约束、存储过程、函数、触发器、游标等数据对象的管理与编程部分，实现了对数据库、表、视图等的创建和增删改查等操作，实现了约束的增加与删除，实现了用户自定义函数和触发器的实现与应用。

完成了数据查询，数据插入、删除与修改等数据处理相关任务部分，基于实验所给出的金融应用场景，实现了按照各种不同的要求进行查询，运用多种查询方法进行数据的查询。

完成了数据库的安全性控制，完整性控制，恢复机制，并发控制机制等系统内核的实验，实现了用户和角色的创建以及权限的授予和收回。

完成了数据库的设计与实现，实现了从概念模型到MySQL的实现，了解并使用了相关建模工具，并对制约因素分析与设计和工程师责任及其分析进行了相关的讨论。

完成了数据库应用系统的开发（JAVA篇），实现了基于JAVA语言的数据库系统的操作，实现了JDBC应用的构建，并基于该应用使用了不同的SQL语言进行了多种具体数据库增删改查操作，使用JAVA语言和SQL语言结合自定义了方法并使用。

总体而言本次课程实践任务工作量较多、难度较为适中，不过老师给了充足且自由的时间来完成，且老师和主教也积极地为我们解答问题，因此能够顺利完成。本次实践除了理论课的知识的应用以外，也加入了许多课外的知识，例如使用JAVA进行数据库应用开发等，这些课外知识的应用向我们展示了数据库系统的广发应用以及强大的兼容性。

对于本次课程实践的建议是，可以适当对每一个实训任务进行合理的实践分配，而不是所有的实训任务的截止日期都相同，这样有利于促进学生的完成，避免学生将所有任务堆积到最后才一起做。