

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | 计科本硕博2001班 |
| 学 号： | U202015628 |
| 姓 名： | 柳子淇 |
| 指导教师： | 袁平鹏 |

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

20 年 1 月 2 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](#_Toc123609079)

[2 任务实施过程与分析 2](#_Toc123609080)

[2.1 数据库、表与完整性约束的定义（Create） 2](#_Toc123609082)

[2.2 表结构与完整性约束的修改(ALTER) 3](#_Toc123609083)

[2.3 数据查询(Select)之一 6](#_Toc123609084)

[2.4 数据查询(Select)之二 11](#_Toc123609085)

[2.5 数据的插入、修改与删除(Insert,Update,Delete) 12](#_Toc123609086)

[2.6 视图 14](#_Toc123609087)

[2.7 存储过程与事务 15](#_Toc123609088)

[2.8 触发器 16](#_Toc123609089)

[2.9 用户自定义函数 17](#_Toc123609090)

[2.10 安全性控制 17](#_Toc123609091)

[2.11 并发控制与事务的隔离级别 19](#_Toc123609092)

[2.12 备份+日志：介质故障与数据库恢复 22](#_Toc123609093)

[2.13 数据库设计与实现 23](#_Toc123609094)

[2.14 数据库应用开发(JAVA篇)（最多跳过一题） 23](#_Toc123609095)

[2.15 数据库的索引B+树实现 35](#_Toc123609096)

[3 课程总结 37](#_Toc123609097)

# 课程任务概述

实训任务内容依托头歌实践教学平台，涉及以下几个部分：

1. 数据库、表与完整性约束的定义
2. 表结构与完整性约束的修改
3. 数据查询
4. 数据的插入、修改与删除
5. 视图
6. 存储过程与事务
7. 触发器
8. 用户自定义函数
9. 安全性控制
10. 并发控制与事务的隔离级别
11. 备份+日志（介质故障与数据库的恢复）
12. 数据库设计与实现
13. 数据库应用的开发（JAVA）
14. 数据库的索引B+树实现
15. 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）。在数据库应用开发环节，使用JAVA 1.8。

# 任务实施过程与分析



## 数据库、表与完整性约束的定义（Create）

本节的六个关卡涉及数据库的创建、表的创建以及表中主码、外码、CHECK、DEFAUT 和 UNIQUE 等约束的建立。

### 创建数据库

本关任务：创建用于2022年北京冬奥会信息系统的数据库:beijing2022。

首先连入MySQL：

mysql ‐h host ‐u user ‐ppassword dbname

创建数据库：

create database beijing2022;

### 创建表及主码约束

本关任务：在指定的数据库中创建一个表，并为表指定主码。

create database TestDb;

use TestDb;

create table t\_emp(

    id INT PRIMARY KEY,

    name VARCHAR(32),

    deptId INT,

    salary FLOAT

);

### 创建外码约束(foreign key)

本关任务：创建外码约束（参照完整性约束）。

create database MyDb;

use MyDb;

CREATE TABLE dept

(

    deptNo INT PRIMARY KEY,

    deptName VARCHAR(32)

);

CREATE TABLE staff

(

    staffNo INT PRIMARY KEY,

    staffName VARCHAR(32),

    gender char(1),

    dob date,

    salary numeric(8,2),

    deptNo INT,

**CONSTRAINT** FK\_staff\_deptNo **FOREIGN KEY**(deptNo) **REFERENCES** dept(deptNo)

);

### CHECK约束

本关任务：请在数据库MyDb中创建表products，并分别实现对品牌和价格的约束，两个CHECK约束的名称分别为CK\_products\_brand和CK\_products\_price，主码约束不要显示命名。

create database MyDb;

use MyDb;

create table products(

    pid char(10) PRIMARY KEY,

    name varchar(32),

    brand char(10) **constraint** CK\_products\_brand **CHECK**(brand in ('A','B')),

    price int  **constraint** CK\_products\_price **CHECK**(price>0)

)

### DEFAULT约束

该关卡任务已完成，实施情况本报告略过。

### UNIQUE约束

该关卡任务已完成，实施情况本报告略过。

## 表结构与完整性约束的修改(ALTER)

本节的四个关卡围绕数据库中表的基本修改操作展开，设计用alter 语句对表的定义进行

修改，如更换/修改表名、列名、列的类型、列约束、表约束；添加或删除列、约束等。

### 修改表名

本关任务：数据库TestDb1中有表your\_table，请根据提示，在右侧代码文件编辑窗中添加恰当的语句，将表名your\_table更改为my\_table。

ALTER TABLE简化后的语法：

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

可见，在一条ALTER TABLE语句里，可以同时对表作多项修改。可选的修改事项还有很多：

可选的修改事项有：

1. 修改事项 ::=
2. ADD [COLUMN] 列名 数据类型 [列约束]
3. [FIRST | AFTER col\_name]
4. | ADD {INDEX|KEY} [索引名] [类型] (列1,...)
5. | ADD [CONSTRAINT [约束名]] 主码约束
6. | ADD [CONSTRAINT [约束名]] UNIQUE约束
7. | ADD [CONSTRAINT [约束名]] 外码约束
8. | ADD [CONSTRAINT [约束名]] CHECK约束
9. | DROP {CHECK|CONSTRAINT} 约束名
10. | ALTER [COLUMN] 列名 {SET DEFAULT {常量 | (表达式)} | DROP DEFAULT}
11. | CHANGE [COLUMN] 列名 新列名 数据类型 [列约束]
12. [FIRST | AFTER col\_name]
13. | DROP [COLUMN] 列名
14. | DROP {INDEX|KEY} 索引名
15. | DROP PRIMARY KEY
16. | DROP FOREIGN KEY fk\_symbol
17. | MODIFY [COLUMN] 列名 数据类型 [列约束]
18. [FIRST | AFTER col\_name]
19. | RENAME COLUMN 列名 TO 新列名
20. | RENAME {INDEX|KEY} 索引名 TO 新索引名
21. | RENAME [TO|AS] 新表名

用Alter Table语句修改表名如下：

USE TestDb1;

#请在以下空白处添加恰当的语句，将表名your\_table更改为my\_table:

alter table your\_table rename my\_table

### 添加与删除字段

本关任务：为表添加和删除字段。

#语句1：删除表orderDetail中的列orderDate

    alter table orderDetail DROP COLUMN orderDate;

#语句2：添加列unitPrice

    alter table orderDetail ADD COLUMN unitPrice numeric(10,2);

### 修改字段

本关任务：你的编程任务是对表addressBook作以下修改：

将QQ号的数据类型改为char(12);

将列名weixin改为wechat。

alter table addressBook modify column QQ char(12);

alter table addressBook rename column Weixin to wechat;

### 添加、删除与修改约束

本关任务：添加、删除与修改约束。

对每个要求修选取合适的修改事项即可，代码如下：

#(1) 为表Staff添加主码

alter table Staff add  primary key (staffNo);

#(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:

alter table Dept add **constraint** FK\_Dept\_mgrStaffNo **foreign key** (mgrStaffNo) **references** Staff(staffNo);

#(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:

alter table Staff add **constraint** FK\_Staff\_dept **foreign key** (dept) **references** Dept(deptNo);

#(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:

alter table Staff add **constraint** CK\_Staff\_gender **check**(gender in ('F','M'));

#(5) 为表Dept添加unique约束：deptName不允许重复。约束名为UN\_Dept\_deptName：

alter table Dept add **constraint** UN\_Dept\_deptName unique (deptName);

## 数据查询(Select)之一

本节的19 个关卡都是select 语句在不同场景下的应用。

### 金融应用场景介绍,查询客户主要信息

本关任务：查询所有客户的名称、手机号和邮箱信息，查询结果按照客户编号排序。简单的select 查询，只涉及最基本的select语法，代码如下：

SELECT c\_name, c\_phone, c\_mail FROM client

### 邮箱为null的客户

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。

注意，null的判断要用 is null，而不能用0或1

SELECT c\_id, c\_name, c\_id\_card, c\_phone FROM client where c\_mail is null

### 既买了保险又买了基金的客户

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。

涉及嵌套查询，和排序，代码如下：

SELECT c\_name, c\_mail,c\_phone FROM client

where c\_id in (

    SELECT pro\_c\_id FROM (SELECT pro\_c\_id FROM property WHERE pro\_type='2') as C2

    where pro\_c\_id in (

        SELECT pro\_c\_id FROM property WHERE pro\_c\_id=C2.pro\_c\_id  AND pro\_type='3'

    )

)

ORDER BY c\_id;

### 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。

涉及多表连接。

select c\_name,c\_phone,b\_number from bank\_card,client client

    where client.c\_id = b\_c\_id

    AND b\_type = '储蓄卡'

ORDER BY c\_id;

### 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。

SELECT p\_id,p\_amount,p\_year FROM finances\_product

where p\_amount>=30000 AND p\_amount<=50000

ORDER BY p\_amount ASC,p\_year DESC;

### 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。

在property 表中基于pro\_income 分组，利用count 函数统计数量，如果某个分组数量大于等于所有分组或者数量最大的分组，则为众数。代码如下：

select pro\_income , count(\*) as presence

from property group by pro\_income

having count(\*)  >= ALL (

select count(\*)

from property

group by pro\_income )

### 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。已知身份证前6位表示居民地区，其中4201开头表示湖北省武汉市。查询身份证隶属武汉市没有买过任何理财产品的客户的名称、电话号、邮箱。依客户编号排序

已知湖北省武汉市居民的身份证号开头为“4201”，用“like "4201%"” 即可筛选出武汉居民，再用exists 语句查询是否购买理财产品即可。代码如下：

SELECT c\_name, c\_phone, c\_mail FROM client

where c\_id\_card LIKE "4201%"

AND NOT EXISTS (

SELECT pro\_c\_id FROM property where pro\_c\_id = client.c\_id AND pro\_type=1

        )

order by c\_id;

### 持有两张信用卡的用户

本关任务： 查询持有两张(含）以上信用卡的用户的名称、身份证号、手机号。查询结果依客户编号排序。

SELECT c\_name,c\_id\_card,c\_phone FROM client

    where (c\_id,"信用卡") in (

        SELECT b\_c\_id,b\_type FROM bank\_card

            group by b\_c\_id, b\_type

            having count(\*) > 1

    );

### 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(f\_type='货币型')基金的用户的名称、电话号、邮箱。依客户编号排序。

SELECT c\_name, c\_phone, c\_mail  FROM client where c\_id in (

SELECT pro\_c\_id FROM property

where pro\_c\_id=client.c\_id

AND pro\_type=3

AND pro\_pif\_id in (

SELECT f\_id fund FROM fund

where f\_id=pro\_pif\_id  AND f\_type='货币型')

)

order by c\_id;

### 投资总收益前三名的客户

本关任务：查询当前总的可用资产收益(被冻结的资产除外)前三名的客户的名称、身份证号及其总收益，按收益降序输出，总收益命名为total\_income。不考虑并列排名情形。

利用“left join” 对用户表client 和资产表property 做连接，按c\_id 分组统计收益和并降序排序，再利用“limit” 输出前3 名用户即可。代码如下：

SELECT c\_name,c\_id\_card, SUM(pro\_income) as total\_income

    FROM property left join client on client.c\_id=pro\_c\_id

    where pro\_status != '冻结'

    group by c\_id

    order by total\_income DESC

    LIMIT 0,3

### 黄姓客户持卡数量

该关卡任务已完成，实施情况本报告略过。

### 客户理财、保险与基金投资总额

该关卡任务已完成，实施情况本报告略过。（与下一关思路一致）

### 客户总资产

综合客户表(client)、资产表(property)、理财产品表(finances\_product)、保险表(insurance)、基金表(fund)，列出所有客户的编号、名称和总资产，总资产命名为total\_property。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(信用卡余额即为透支金额)。客户总资产包括被冻结的资产。

对于每个客户每种类型的资产总和都用一条 select 语句获取，将全部的结果利用union all 取可重并集，再与用户表client 做连接，最后再按c\_id 分组，统计资产和并降序排序

即可。

select

    c\_id,

    c\_name,

    ifnull(sum(amount), 0) as total\_property

from client

    left join (

        select

            pro\_c\_id,

            pro\_quantity \* p\_amount as amount

        from property, finances\_product

        where pro\_pif\_id = p\_id

        and pro\_ty pe = 1

        union all

        select

            pro\_c\_id,

            pro\_quantity \* i\_amount as amount

        from property, insurance

        where pro\_pif\_id = i\_id

        and pro\_type = 2

        union all

        select

            pro\_c\_id,

            pro\_quantity \* f\_amount as amount

        from property, fund

        where pro\_pif\_id = f\_id

        and pro\_type = 3

        union all

        select

            pro\_c\_id,

            sum(pro\_income) as amount

        from property

        group by pro\_c\_id

        union all

        select

            b\_c\_id,

            sum(if(b\_type = "储蓄卡", b\_balance, -b\_balance)) as amount

        from bank\_card

        group by b\_c\_id

    ) pro

    on c\_id = pro.pro\_c\_id

group by c\_id

order by c\_id;

### 第N高问题

该关卡任务已完成，实施情况本报告略过。

### 基金收益两种方式排名

本关任务：查询资产表中客户编号，客户基金投资总收益,基金投资总收益的排名(从高到低排名)。总收益相同时名次亦相同(即并列名次)。总收益命名为total\_revenue, 名次命名为rank。第一条SQL语句实现全局名次不连续的排名，第二条SQL语句实现全局名次连续的排名。不管哪种方式排名，收益相同时,客户编号小的排在前。

分别调用rank()，和dense\_rank()即可。

select

    pro\_c\_id,

    sum(pro\_income) as total\_revenue,

    rank() over(order by sum(pro\_income) desc) as "rank"

from property

where pro\_type = 3

group by pro\_c\_id

order by total\_revenue desc, pro\_c\_id;

*-- (2) 基金总收益排名(名次连续)*

select

    pro\_c\_id,

    sum(pro\_income) as total\_revenue,

    dense\_rank() over(order by sum(pro\_income) desc) as "rank"

from property

where pro\_type = 3

group by pro\_c\_id

order by total\_revenue desc, pro\_c\_id;

### 持有完全相同基金组合的客户

该关卡任务已完成，实施情况本报告略过。

### 购买基金的高峰期

该关卡任务已完成，实施情况本报告略过。

### 至少有一张信用卡余额超过5000元的客户信用卡总余额

该关卡任务已完成，实施情况本报告略过。

### 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，实施情况本报告略过。

## 数据查询(Select)之二

本小节子任务仍然以第2.3子任务的数据库内容为背景。

### 查询销售总额前三的理财产品

查询2010年和2011年这两年每年销售总额前3名（如果有并列排名，则后续排名号跳过之前的并列排名个数，例如1、1、3）的统计年份、销售总额排名值、理财产品编号、销售总额。按照年份升序排列，同一年份按照销售总额的排名值升序排列，如遇到并列排名则按照理财产品编号升序排列。

select \*  from(

    select pro\_purchase\_time as pyear,

        rank() over(partition by pro\_purchase\_time order by sum(p\_amount\*pro\_quantity) desc) as rk,

        p\_id,

        sum(p\_amount\*pro\_quantity) as sumamount

    from

        (select pro\_pif\_id, pro\_quantity,pro\_type,year(pro\_purchase\_time) as pro\_purchase\_time from property) as a

        left join finances\_product  on a.pro\_pif\_id = finances\_product.p\_id

     where pro\_type = 1 and pro\_purchase\_time in (2010,2011)

     group by p\_id,pro\_purchase\_time

) as b

limit 0,6

* + 1. **投资积极且偏好理财类产品的客户**

**该任务关卡跳过。**

* + 1. **查询购买了所有畅销理财产品的客户**

**该任务关卡跳过。**

* + 1. **查找相似的理财产品**

**该任务关卡跳过。**

* + 1. **查询任意两个客户的相同理财产品数**

**该任务关卡跳过。**

* + 1. **查找相似的理财客户**

**该任务关卡跳过。**

## 数据的插入、修改与删除(Insert,Update,Delete)

本节的6 个关卡围绕Insert, Update, Delete 语句在不同场景下的应用展开。

### 插入多条完整的客户信息

本关任务：向客户表client插入数据。

向客户表插入以下3条数据:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **c\_id** | **c\_name** | **c\_mail** | **c\_id\_card** | **c\_phone** | **c\_password** |
| 1 | 林惠雯 | [960323053@qq.com](mailto:960323053@qq.com) | 411014196712130323 | 15609032348 | Mop5UPkl |
| 2 | 吴婉瑜 | [1613230826@gmail.com](mailto:1613230826@gmail.com) | 420152196802131323 | 17605132307 | QUTPhxgVNlXtMxN |
| 3 | 蔡贞仪 | [252323341@foxmail.com](mailto:252323341@foxmail.com) | 160347199005222323 | 17763232321 | Bwe3gyhEErJ7 |

insert into client value ( 1, '林惠雯',    '960323053@qq.com', '411014196712130323',   '15609032348',  'Mop5UPkl');

insert into client value ( 2, '吴婉瑜',    '1613230826@gmail.com', '420152196802131323',   '17605132307',  'QUTPhxgVNlXtMxN');

insert into client value (3, '蔡贞仪', '252323341@foxmail.com',    '160347199005222323',   '17763232321',  'Bwe3gyhEErJ7' );

### 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。

*-- 请用一条SQL语句将这名客户的信息插入到客户表(client)：*

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

### 批量插入数据

本关任务：向客户表client批量插入数据。已知表new\_client保存了一批新客户信息，该表与client表结构完全相同。请用一条SQL语句将new\_client表的全部客户信息插入到客户表(client)。

insert into client select \* from new\_client;

### 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。请用一条SQL语句删除client表中没有银行卡的客户信息。注意：MySQL的delete语句中from关键词不能省略。

delete from client where not exists (select b\_c\_id from bank\_card where bank\_card.b\_c\_id = client.c\_id);

### 冻结客户资产

本关任务：冻结客户的投资资产。请用一条update语句将手机号码为“13686431238”这位客户的投资资产(理财、保险与基金)的状态置为”冻结”。

update property set pro\_status='冻结'

    where pro\_c\_id = (

        select c\_id from client where property.pro\_c\_id = client.c\_id AND c\_phone = '13686431238'

    );

### 连接更新

本关任务：根据客户表的内容修改资产表的内容。在金融应用场景数据库中，已在表property(资产表)中添加了客户身份证列，列名为pro\_id\_card，类型为char(18)，该列目前全部留空(null)。

请用一条update语句，根据client表中提供的身份证号(c\_id\_card)，填写property表中对应的身份证号信息(pro\_id\_card)。

update property set pro\_id\_card=(

    select c\_id\_card from client where property.pro\_c\_id = client.c\_id);

## 视图

### 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。

创建包含所有保险资产记录的详细信息的视图v\_insurance\_detail，包括购买客户的名称、客户的身份证号、保险名称、保障项目、商品状态、商品数量、保险金额、保险年限、商品收益和购买时间。

视图名：v\_insurance\_detail

视图列：c\_name,c\_id\_card,i\_name,i\_project,pro\_status,pro\_quantity,i\_amount,i\_year,pro\_income,pro\_purchase\_time

create view  v\_insurance\_detail as

    select  c\_name,c\_id\_card,i\_name,i\_project, pro\_status,pro\_quantity, i\_amount, i\_year, pro\_income, pro\_purchase\_time

    from property,client,insurance

    where c\_id = pro\_c\_id and pro\_type =2 and pro\_pif\_id=i\_id

### 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。

基于上一关创建的视图v\_insurance\_detail进行分组统计查询，列出每位客户的姓名，身份证号，保险投资总额(insurance\_total\_amount)和保险投资总收益(insurance\_total\_revenue),结果依保险投资总额降序排列。

输出：c\_name,c\_id\_card,insurance\_total\_amount,insurance\_total\_revenue

select

    c\_name,

    c\_id\_card,

    sum(pro\_quantity \* i\_amount) as insurance\_total\_amount,

    sum(pro\_income) as insurance\_total\_revenue

from v\_insurance\_detail

group by c\_id\_card

order by insurance\_total\_amount desc

## 存储过程与事务

### 使用流程控制语句的存储过程

本关任务：创建存储过程sp\_fibonacci(in m int)，向表fibonacci插入斐波拉契数列的前m项，及其对应的斐波拉契数。fibonacci表初始值为一张空表。请保证存储过程可以多次运行而不出错。

斐波那契， 维护三个变量a b c，每轮循环，计算c = a+b ，然后更新a，b（向右移动）然后n++ 直到n=m

use fib;

*-- 创建存储过程`sp\_fibonacci(in m int)`，向表fibonacci插入斐波拉契数列的前m项，及其对应的斐波拉契数。fibonacci表初始值为一张空表。请保证你的存储过程可以多次运行而不出错。*

drop procedure if exists sp\_fibonacci;

delimiter $$

create procedure sp\_fibonacci(in m int)

begin

######## 请补充代码完成存储过程体 ########

declare n int **default** 0;

declare a int **default** -1;

declare b decimal(21) unsigned **default** 1;

declare c decimal(21) unsigned **default** 0;

TRUNCATE TABLE fibonacci;

WHILE n<m DO

    SET c=a+b;

    SET a=b;

    SET b=c;

    INSERT into fibonacci values(n,c);

    SET n=n+1;

END WHILE;

end $$

delimiter ;

### 使用游标的存储过程

该关卡任务已完成，实施情况本报告略过。

### 使用事务的存储过程

该关卡任务已完成，实施情况本报告略过。

## 触发器

### 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。

各投资品种一经销售，不会再改变；

也不需考虑finances\_product，insurance，fund的业务规则(一经销售的理财、保险和基金产品信息会永久保存，不会被删除或修改，即使不再销售该类产品)。

use finance1;

drop trigger if exists before\_property\_inserted;

*-- 请在适当的地方补充代码，完成任务要求：*

delimiter $$

CREATE TRIGGER before\_property\_inserted BEFORE INSERT ON property

FOR EACH ROW

BEGIN

    declare tp int **default** new.pro\_type;

    declare id int **default** new.pro\_pif\_id;

    declare msg varchar(50);

    if tp = 1 then

        if id not in (select p\_id from finances\_product) then

            set msg = concat("finances product #", id, " not found!");

        end if;

    elseif tp = 2 then

        if id not in (select i\_id from insurance) then

            set msg = concat("insurance #", id, " not found!");

        end if;

    elseif tp = 3 then

        if id not in (select f\_id from fund) then

            set msg = concat("fund #", id, " not found!");

        end if;

    else

        set msg = concat("type ", tp, " is illegal!");

    end if;

    if msg is not null then

        signal sqlstate "45000" set message\_text = msg;

    end if;

END$$

delimiter ;

## 用户自定义函数

### 创建函数并在语句中使用它

该任务关卡跳过。

## 安全性控制

本节的2 个关卡涉及数据库中的用户、角色和权限等内容。

### 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。

填写语句，完成以下创建用户和授权操作：

(1) 创建用户tom和jerry，初始密码均为'123456'；

(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；

(3) 授予用户jerry修改银行卡余额的权限；

(4) 收回用户Cindy查询银行卡信息的权限。

# 请填写语句，完成以下功能：

#(1) 创建用户tom和jerry，初始密码均为'123456'；

create user tom identified by "123456";

create user jerry identified by "123456";

#(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；

grant select (c\_mail, c\_name, c\_phone) on client to tom with grant option;

#(3) 授予用户jerry修改银行卡余额的权限；

grant update (b\_balance) on bank\_card to jerry;

#(4) 收回用户Cindy查询银行卡信息的权限。

revoke select on bank\_card from Cindy;

### 用户、角色与权限

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。

填写语句，完成以下创建用户和授权操作：

(1) 创建角色client\_manager和fund\_manager；

(2) 授予client\_manager对client表拥有select,insert,update的权限；

(3) 授予client\_manager对bank\_card表拥有查询除银行卡余额外的select权限；

(4) 授予fund\_manager对fund表的select,insert,update权限；

(5) 将client\_manager的权限授予用户tom和jerry；

(6) 将fund\_manager权限授予用户Cindy.

# 请填写语句，完成以下功能：

# (1) 创建角色client\_manager和fund\_manager；

create user client\_manager;

create user fund\_manager;

# (2) 授予client\_manager对client表拥有select,insert,update的权限；

grant select, insert, update on client to client\_manager;

# (3) 授予client\_manager对bank\_card表拥有查询除银行卡余额外的select权限；

grant select (b\_c\_id, b\_number, b\_type) on bank\_card to client\_manager;

# (4) 授予fund\_manager对fund表的select,insert,update权限；

grant select, insert, update on fund to fund\_manager;

# (5) 将client\_manager的权限授予用户tom和jerry；

grant client\_manager to tom, jerry;

# (6) 将fund\_manager权限授予用户Cindy.

grant fund\_manager to Cindy;

## 并发控制与事务的隔离级别

本节的6 个关卡涉及数据库中并发控制与事务的隔离级别相关内容，包括隔离级别的设置，事务的开启、提交和回滚等，还通过添加等待代码实现了读脏、不可重复读、幻读等出错场景。

背景介绍：有表ticket记录了航班余票数，其结构如下表所示：

| **列** | **类型** | **说明** |
| --- | --- | --- |
| flight\_no | char(6) | primary key |
| tickets | int | 余票数 |

有两个涉及该表的并发事务t1和t2，分别定义在t1.sql和t2.sql代码文件中。平台会让两个事务并发执行，请同学们通过修改代码文件来达到题目预期的并发执行效果。

### 并发控制与事务的隔离级别

本关任务：设置事务的隔离级别。

MySQL 的事务隔离级别从低到高分以下四级：

1. 读未提交（READ UNCOMITTED ）

2. 读已提交（READ COMMITTED ）

3. 可重复读（REPEATABLE READ ）

4. 可串行化（SERIALIZABLE ）

低隔离级别可以支持更高的并发处理，同时占用的系统资源更少,但可能产生数据不一致的情形也更多一些。

MySQL 事务隔离级别及其可能产生的问题如下表所示：

| **隔离级别** | **读脏** | **不可重复读** | **幻读** |
| --- | --- | --- | --- |
| READ UNCOMITTED | √ | √ | √ |
| READ COMMITTED | × | √ | √ |
| REPEATABLE READ | × | × | √ |
| SERIALIZABLE | × | × | × |

根据提示补充适当的代码，将事务的隔离级别设置为read committed; 并以rollback语句结束事务。

*-- 请不要在本代码文件中添加空行！！！*

use testdb1;

# 设置事务的隔离级别为 read uncommitted

set session transaction isolation level read uncommitted;

*-- 开启事务*

start transaction;

insert into dept(name) values('运维部');

# 回滚事务：

rollback;

*/\* 结束 \*/*

### 读脏

本关任务：选择合适的事务隔离级别，构造两个事务并发执行时，发生“读脏”现象。

请在两个代码文件适当的地方补充代码，构造“读脏”现象。t1是读脏的那个事务,而t2是那个修改数据后又撤销的事务。

t1.sql

*-- 事务1:*

use testdb1;

## 请设置适当的事务隔离级别

set session transaction isolation level read uncommitted;

start transaction;

*-- 时刻2 - 事务1读航班余票,发生在事务2修改之后*

## 添加等待代码，确保读脏

set @n = sleep(1);

select tickets from ticket where flight\_no = 'CA8213';

commit;

t2.sql

*-- 事务2*

use testdb1;

## 请设置适当的事务隔离级别

set session transaction isolation level read uncommitted;

start transaction;

*-- 时刻1 - 事务2修改航班余票*

update ticket set tickets = tickets - 1 where flight\_no = 'CA8213';

*-- 时刻3 - 事务2 取消本次修改*

## 请添加代码，使事务1在事务2撤销前读脏;

set @n = sleep(2);

rollback;

### 不可重复读

该任务关卡跳过。

### 幻读

本题中t2.sql已提前写好。在代码文件t1.sql适当的地方补充代码，要求如下:

两次查询余票超过300张的航班信息(第2次查询已提前写好)；

在第1次查询之后，事务t2插入了一条航班信息并提交；

第2次查询的记录数增多,发生“幻读”。

不得修改t1的事务隔离级别(保持默认的repeatable read)。

*-- 事务1（采用默认的事务隔离级别- repeatable read）:*

use testdb1;

select @@transaction\_isolation;

start transaction;

*-- 第1次查询余票超过300张的航班信息*

select \* from ticket where tickets > 300;

set @n = sleep(2);

*-- 修改航班MU5111的执飞机型为A330-300：*

update ticket set aircraft = 'A330-300' where flight\_no = 'MU5111';

*-- 第2次查询余票超过300张的航班信息*

select \* from ticket where tickets > 300;

commit;

### 主动加锁保证可重复读

该任务关卡跳过。

### 可串行化

该任务关卡跳过。

## 备份+日志：介质故障与数据库恢复

本节的2 个关卡涉及数据库的备份与恢复方法。

### 备份与恢复

设有居民人口登记数据库residents,请为该数据库做一次静态的(个人独享服务器)海量逻辑备份，备份文件命名为residents\_bak.sql。 然后再用该逻辑备份文件恢复数据库。

本关任务：备份数据库，然后再恢复它。

使用mysqldump 指令将服务器上的数据库residents 备份至文件residents\_bak.sql 中：

mysqldump -h127.0.0.1 -uroot *--databases residents > residents\_bak.sql*

### 备份+日志：介质故障的发生与数据库的恢复

该关卡任务已完成，实施情况本报告略过。

## 数据库设计与实现

### 从概念模型到MySQL实现

该关卡任务已完成，实施情况本报告略过。

### 从需求分析到逻辑模型

该任务关卡跳过。

### 建模工具的使用

该任务关卡跳过。

### 制约因素分析与设计

在从实际问题的建模到数据库的概念模型和逻辑模型的构建过程中，需要考虑若干制约因素。以机票订票系统为例，系统需要考虑到旅客的实际情况，旅客可以多次乘坐飞机，一张机票肯定是某个用户为某个特定的旅客购买的特定航班的机票，所以机票信息不仅跟乘坐人有关，同时还需要记录购买人信息(虽然两者有时是同一人) 。此外，对于系统的权限也存在若干要求，例如实体“用户” 就刻印根据权限分成两类，用户分两类：普通用户可以订票，管理

用户有权限维护和管理整个系统的运营。

### 工程师责任及其分析

社会方面，工程师应该能够基于工程相关背景知识进行合理分析，评价专业工程实践和复杂工程问题解决方案对社会、健康、安全、法律以及文化的影响，并理解应承担的责任。安全方面，工程师应该尽可能考虑系统中存在的安全漏洞，安全性是所有系统用户关心的重要命题；科学发展方面，工程师应该能够基于科学原理并采用科学方法对复杂工程问题进行研究，包括设计实验、分析与解释数据、并通过信息综合得出合理有效的结论。

## 数据库应用开发(JAVA篇)（最多跳过一题）

本节的7 个关卡从JDBC 体系结构出发，涉及JAVA 开发数据库应用的基本知识。

### 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。

(只显示了try-catch块中的内容)

…

try {

            Class.forName(JDBC\_DRIVER);

            connection = DriverManager.getConnection(DB\_URL, USER, PASS);

            statement = connection.createStatement();

            resultSet = statement.executeQuery("select c\_name, c\_mail, c\_phone from client where c\_mail is not null");

            System.out.println("姓名\t邮箱\t\t\t\t电话");

            while (resultSet.next()) {

                System.out.print(resultSet.getString("c\_name") + "\t");

                System.out.print(resultSet.getString("c\_mail") + "\t\t");

                System.out.println(resultSet.getString("c\_phone"));

            }

         } catch (**ClassNotFoundException** e) {

    …

### 用户登录

本节的7 个关卡从JDBC 体系结构出发，涉及JAVA 开发数据库应用的基本知识。

2.13.1 JDBC

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。

编程体验客户登录功能.程序先后提示客户输用户名和密码:

请输入用户名：

请输入密码：

客户的邮箱(c\_mail)充当用户名,而不是编号(c\_id).通常邮箱更容易记住.

根据客户的输入,输出以下两类信息之一:

登录成功。

用户名或密码错误！

import **java.sql.\***;

import **java.util.Scanner**;

**public** **class** Login {

**public** **static** **void** main(**String**[] args) {

**Connection** connection = null;

*//申明下文中的resultSet, statement*

**Statement** statement = null;

**ResultSet** resultSet = null;

**Scanner** input = new Scanner(System.in);

        System.out.print("请输入用户名：");

**String** loginName = input.nextLine();

        System.out.print("请输入密码：");

**String** loginPass = input.nextLine();

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

**String** userName = "root";

**String** passWord = "123123";

**String** url = "jdbc:mysql://127.0.0.1:3306/finance?useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";

            connection = DriverManager.getConnection(url, userName, passWord);

*// 补充实现代码:*

            Class.forName("com.mysql.cj.jdbc.Driver");

            statement = connection.createStatement();

**String** sql = "select \* from client where c\_mail = ? and c\_password = ?";

**PreparedStatement** ps = connection.prepareStatement(sql);

            ps.setString(1, loginName);

            ps.setString(2, loginPass);

            resultSet = ps.executeQuery();

            if (resultSet.next())

                System.out.println("登录成功。");

            else

                System.out.println("用户名或密码错误！");

         } catch (**ClassNotFoundException** e) {

            e.printStackTrace();

        } catch (**SQLException** throwables) {

            throwables.printStackTrace();

        } finally {

            try {

                if (resultSet != null) {

                    resultSet.close();

                }

                if (statement != null) {

                    statement.close();

                }

                if (connection != null) {

                    connection.close();

                }

            } catch (**SQLException** throwables) {

                throwables.printStackTrace();

            }

        }

    }

}

### 添加新客户

本关任务：编程完成向client(客户表)插入记录的方法。

import **java.sql.\***;

import **java.util.Scanner**;

**public** **class** AddClient {

**static** **final** **String** JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

**static** **final** **String** DB\_URL = "jdbc:mysql://127.0.0.1:3306/finance?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";

**static** **final** **String** USER = "root";

**static** **final** **String** PASS = "123123";

**public** **static** **int** insertClient(**Connection** con, **int** c\_id, **String** c\_name, **String** c\_mail,

**String** c\_id\_card, **String** c\_phone, **String** c\_password) {

**String** sql = "insert into client values (?, ?, ?, ?, ?, ?)";

        try {

**PreparedStatement** ps = con.prepareStatement(sql);

            ps.setInt(1, c\_id);

            ps.setString(2, c\_name);

            ps.setString(3, c\_mail);

            ps.setString(4, c\_id\_card);

            ps.setString(5, c\_phone);

            ps.setString(6, c\_password);

            return ps.executeUpdate();

        } catch (**SQLException** e) {

            e.printStackTrace();

        }

        return 0;

    }

**public** **static** **void** main(**String**[] args) **throws** **Exception** {

**Scanner** sc = new Scanner(System.in);

        Class.forName(JDBC\_DRIVER);

**Connection** connection = DriverManager.getConnection(DB\_URL, USER, PASS);

        while (sc.hasNext()) {

**String** input = sc.nextLine();

            if (input.equals(""))

                break;

**String**[]commands = input.split(" ");

            if (commands.length == 0)

                break;

**int** id = Integer.parseInt(commands[0]);

**String** name = commands[1];

**String** mail = commands[2];

**String** idCard = commands[3];

**String** phone = commands[4];

**String** password = commands[5];

            insertClient(connection, id, name, mail, idCard, phone, password);

        }

    }

}

### 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。

import **java.sql.\***;

import **java.util.Scanner**;

**public** **class** RemoveCard {

**static** **final** **String** JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

**static** **final** **String** DB\_URL = "jdbc:mysql://127.0.0.1:3306/finance?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";

**static** **final** **String** USER = "root";

**static** **final** **String** PASS = "123123";

*/\*\**

*\* 删除bank\_card表中数据*

*\**

*\* @param connection 数据库连接对象*

*\* @param b\_c\_id 客户编号*

*\* @param c\_number 银行卡号*

*\*/*

**public** **static** **int** removeBankCard(**Connection** connection,

**int** b\_c\_id, **String** b\_number){

**String** sql = "delete from bank\_card where b\_c\_id = ? and b\_number = ?";

        try {

**PreparedStatement** ps = connection.prepareStatement(sql);

            ps.setInt(1, b\_c\_id);

            ps.setString(2, b\_number);

            return ps.executeUpdate();

        } catch (**SQLException** e) {

            e.printStackTrace();

        }

        return 0;

    }

*// 不要修改main()*

**public** **static** **void** main(**String**[] args) **throws** **Exception** {

**Scanner** sc = new Scanner(System.in);

        Class.forName(JDBC\_DRIVER);

**Connection** connection = DriverManager.getConnection(DB\_URL, USER, PASS);

        while(sc.hasNext())

        {

**String** input = sc.nextLine();

            if(input.equals(""))

                break;

**String**[]commands = input.split(" ");

            if(commands.length ==0)

                break;

**int** id = Integer.parseInt(commands[0]);

**String** carNumber = commands[1];

**int** n = removeBankCard(connection, id, carNumber);

            if (n > 0) {

               System.out.println("已销卡数：" + n);

            } else {

               System.out.println("销户失败，请检查客户编号或银行卡号！" );

            }

        }

    }

}

### 客户修改密码

本关任务：

编写修改客户登录密码的的方法。实现修改密码的方法passwd()。客户修改密码通常需要确认客户身份，即客户需提供用户名(以邮箱为用户名)和密码，同时还需要输两次新密码，以免客户实际输入的密码与心中想的不一致，只有当所有条件(合法的客户，两次密码输入一致)时才修改密码。

import **java.sql.\***;

import **java.util.Scanner**;

**public** **class** ChangePass {

**static** **final** **String** JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

**static** **final** **String** DB\_URL = "jdbc:mysql://127.0.0.1:3306/finance?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";

**static** **final** **String** USER = "root";

**static** **final** **String** PASS = "123123";

*/\*\**

*\* 修改客户密码*

*\**

*\* @param connection 数据库连接对象*

*\* @param mail 客户邮箱,也是登录名*

*\* @param password 客户登录密码*

*\* @param newPass  新密码*

*\* @return*

*\*   1 - 密码修改成功*

*\*   2 - 用户不存在*

*\*   3 - 密码不正确*

*\*  -1 - 程序异常(如没能连接到数据库等）*

*\*/*

**public** **static** **int** passwd(**Connection** connection,

**String** mail,

**String** password,

**String** newPass){

**String** sql = "select \* from client where c\_mail = ?";

        try {

**PreparedStatement** ps = connection.prepareStatement(sql);

            ps.setString(1, mail);

**ResultSet** res = ps.executeQuery();

            if (res.next()) {

                if (password.equals(res.getString("c\_password"))) {

                    sql = "update client set c\_password = ? where c\_mail = ? and c\_password = ?";

                    ps = connection.prepareStatement(sql);

                    ps.setString(1, newPass);

                    ps.setString(2, mail);

                    ps.setString(3, password);

                    ps.executeUpdate();

                    return 1;

                } else

                    return 3;

            } else

                return 2;

        } catch (**SQLException** e) {

            e.printStackTrace();

        }

        return -1;

    }

*// 不要修改main()*

**public** **static** **void** main(**String**[] args) **throws** **Exception** {

**Scanner** sc = new Scanner(System.in);

        Class.forName(JDBC\_DRIVER);

**Connection** connection = DriverManager.getConnection(DB\_URL, USER, PASS);

        while(sc.hasNext())

        {

**String** input = sc.nextLine();

            if(input.equals(""))

                break;

**String**[]commands = input.split(" ");

            if(commands.length ==0)

                break;

**String** email = commands[0];

**String** pass = commands[1];

**String** pwd1 = commands[2];

**String** pwd2 = commands[3];

            if (pwd1.equals(pwd2)) {

**int** n = passwd(connection, email, pass, pwd1);

              System.out.println("return: " + n);

            } else {

              System.out.println("两次输入的密码不一样!");

            }

        }

    }

}

### 事务与转账操作

本关任务：编写一个银行卡转账的方法。

transferBalance()在被调用前，柜台已经确认过转出帐号持有者身份，所以转帐方法只接受转出卡号，转入卡号和转账金额三个参数。由调用者保证转账金额为正数。transferBalance()返回boolean值，true表示转帐成功，false表示转账失败，并不需要细分或解释失败的原因。

下列任一情形都不可转账(转账失败的原因)：

转出或转入帐号不存在

转出账号是信用卡

转出帐号余额不足

import **java.sql.\***;

import **java.util.Scanner**;

**public** **class** Transfer {

**static** **final** **String** JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

**static** **final** **String** DB\_URL = "jdbc:mysql://127.0.0.1:3306/finance?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";

**static** **final** **String** USER = "root";

**static** **final** **String** PASS = "123123";

*/\*\**

*\* 转账操作*

*\**

*\* @param connection 数据库连接对象*

*\* @param sourceCard 转出账号*

*\* @param destCard 转入账号*

*\* @param amount  转账金额*

*\* @return boolean*

*\*   true  - 转账成功*

*\*   false - 转账失败*

*\*/*

**public** **static** **boolean** transferBalance(**Connection** con, **String** sourceCard, **String** destCard, **double** amount) {

        try {

**String** sql = "select \* from bank\_card where b\_number = ?";

**PreparedStatement** ps = con.prepareStatement(sql);

            ps.setString(1, sourceCard);

**ResultSet** res = ps.executeQuery();

            if (!res.next() || res.getString("b\_type").equals("信用卡") || res.getDouble("b\_balance") < amount)

                return false;

            ps = con.prepareStatement(sql);

            ps.setString(1, destCard);

            res = ps.executeQuery();

            if (!res.next())

                return false;

**double** rcv\_amount = res.getString("b\_type").equals("信用卡") ? -amount : amount;

            sql = "update bank\_card set b\_balance = b\_balance + ? where b\_number = ?";

            ps = con.prepareStatement(sql);

            ps.setDouble(1, -amount);

            ps.setString(2, sourceCard);

            ps.executeUpdate();

            ps = con.prepareStatement(sql);

            ps.setDouble(1, rcv\_amount);

            ps.setString(2, destCard);

            ps.executeUpdate();

            return true;

        } catch (**SQLException** e) {

            e.printStackTrace();

        }

        return false;

    }

*// 不要修改main()*

**public** **static** **void** main(**String**[] args) **throws** **Exception** {

**Scanner** sc = new Scanner(System.in);

        Class.forName(JDBC\_DRIVER);

**Connection** connection = DriverManager.getConnection(DB\_URL, USER, PASS);

        while(sc.hasNext())

        {

**String** input = sc.nextLine();

            if(input.equals(""))

                break;

**String**[]commands = input.split(" ");

            if(commands.length ==0)

                break;

**String** payerCard = commands[0];

**String**  payeeCard = commands[1];

**double**  amount = Double.parseDouble(commands[2]);

            if (transferBalance(connection, payerCard, payeeCard, amount)) {

              System.out.println("转账成功。" );

            } else {

              System.out.println("转账失败,请核对卡号，卡类型及卡余额!");

            }

        }

    }

}

### 把稀疏表格转为键值对存储

本关任务：

将一个稀疏的表中有保存数据的列值，以键值对(列名，列值 )的形式转存到另一个表中，这样可以直接丢失没有值列。sc表初始为空表, 程序依前述规则将entrance\_exam表的值转写到sc表。对每一行，请从左至右依次考察每一列，转存非空列。

import **java.sql.\***;

**public** **class** Transform {

**static** **final** **String** JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

**static** **final** **String** DB\_URL = "jdbc:mysql://127.0.0.1:3306/sparsedb?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";

**static** **final** **String** USER = "root";

**static** **final** **String** PASS = "123123";

*/\*\**

*\* 向sc表中插入数据*

*\**

*\* @param con 数据库连接对象*

*\* @param sno 学号*

*\* @param col\_name 科目*

*\* @param col\_value 成绩*

*\* @return void*

*\*/*

**public** **static** **void** insertSC(**Connection** con, **int** sno, **String** col\_name, **int** col\_value) {

        try {

**String** sql = "insert into sc values (?, ?, ?)";

**PreparedStatement** ps = con.prepareStatement(sql);

            ps.setInt(1, sno);

            ps.setString(2, col\_name);

            ps.setInt(3, col\_value);

            ps.executeUpdate();

        } catch (**SQLException** e) {

            e.printStackTrace();

        }

    }

**public** **static** **void** main(**String**[] args) **throws** **Exception** {

        Class.forName(JDBC\_DRIVER);

**Connection** con = DriverManager.getConnection(DB\_URL, USER, PASS);

**String**[] subject = {"chinese", "math", "english", "physics", "chemistry", "biology", "history", "geography", "politics"};

        try {

**ResultSet** res = con.createStatement().executeQuery("select \* from entrance\_exam");

            while (res.next()) {

**int** sno = res.getInt("sno"), score;

                for (**String** sub : subject) {

                    score = res.getInt(sub);

                    if (!res.wasNull())

                        insertSC(con, sno, sub, score);

                }

            }

        } catch (**SQLException** e) {

            e.printStackTrace();

        }

    }

}

## 数据库的索引B+树实现

### BPlusTreePage的设计

作为B+树索引结点类型的数据结构设计的第一部分：实现BPlusTreePage类，该类是B+树叶结点类型和内部结点类型的父类，提供B+树结点的基本功能。

该任务关卡跳过。

### BPlusTreeInternalPage的设计

作为B+树索引结点类型的数据结构设计的第二部分：实现BPlusTreeInternalPage类，该类作为B+树的内部结点类型，提供B+树内部结点的功能。

该任务关卡跳过。

### BPlusTreeLeafPage的设计

作为B+树索引结点类型的数据结构设计的第三部分：实现BPlusTreeLeafPage类，该类作为B+树的叶子结点类型，提供B+树叶结点的功能。

该任务关卡跳过。

### B+树索引：Insert

在完成B+树相关数据结构的设计后，请完成B+树索引的插入功能。实现B+树迭代器。

该任务关卡跳过。

### B+树索引：Remove

在成功实现B+树索引的插入功能后，请完成B+树索引的删除功能。

该任务关卡跳过。

# 3 课程总结

本次数据库课程实验包含了表的完整性约束的创建和修改、数据查询、数据的插入、修改与删除、视图的创建与使用、存储过程与事务、触发器、用户自定义函数、安全性控制、并发控制与事务的隔离级别、数据库的备份与日志、数据库的设计与实现和数据库应用开发、数据库的索引B+树实现等15个实训实验。

在实验进行的过程中，我通过自主查阅资料边做边学，学到了很多实用的SQL编程知识和技巧，也对理论课程所学内容进行了巩固，深化了理解。此次课程实验内容充实完整，引导性强，有成就感。感谢老师和助教的悉心指导！