

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS2006 |
| 学 号： | U202015471 |
| 姓 名： | 杨释钧 |
| 指导教师： | 左琼 |

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

2023 年 1 月 1 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](#_Toc123487615)

[2 任务实施过程与分析 2](#_Toc123487616)

[2.1 数据库、表与完整性约束的定义（Create） 2](#_Toc123487617)

[2.2 表结构与完整性约束的修改（ALTER） 3](#_Toc123487618)

[2.3 基于金融应用的数据查询（select） 4](#_Toc123487619)

[2.4 数据查询(Select)之二 9](#_Toc123487620)

[2.5 数据的插入、修改与删除（Insert、Update、Delete） 10](#_Toc123487621)

[2.6 视图 11](#_Toc123487622)

[2.7 存储过程与事务 12](#_Toc123487623)

[2.8 触发器 13](#_Toc123487624)

[2.9 用户自定义函数 14](#_Toc123487625)

[2.10 安全性控制 14](#_Toc123487626)

[2.11 并发控制与事务的隔离级别 16](#_Toc123487627)

[2.12 备份+日志：介质故障与数据库恢复 18](#_Toc123487628)

[2.13 数据库设计与实现 18](#_Toc123487629)

[2.14 数据库应用开发(JAVA篇) 19](#_Toc123487630)

[2.15 数据库的索引B+树实现 23](#_Toc123487631)

[3 课程总结 25](#_Toc123487632)

# 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。

本课程相关资料网站：

MYSQL手册：<https://dev.mysql.com/doc/>

JAVA手册：<https://docs.oracle.com/javase/8/docs/api/index.html>

课程开放资源：<https://gitee.com/kylin8575543/db2022-spring>

# 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）

本节需要我们掌握MySQL中创建数据库表，定义数据完整性约束，需要在实践中掌握MySQL中用Create创建数据库表以及增加如主码约束、外码约束等完整性约束的方法，这一部分在Educoder平台上给出了较为详细的知识介绍，难度较低。

### 2.1.1 创建数据库

本关任务要求创建用于2022年北京冬奥会信息系统的数据库beijing2022，是对create命令的直接使用，比较简单，通过代码如下。

**Create** **database** **beijing2022**;

### 2.1.2 创建表及表的主码约束

本关要求在数据库TestDb中创建表t\_emp，要求字段id作为表的主码。本关需要我们了解主码约束的概念以及用CONSTRAINT关键字创建约束的方法。通过代码如下：

**use** TestDb;

**create** **table** **t\_emp**(

id INT,

**name** VARCHAR(32),

    deptId INT,

    salary **FLOAT**,

**constraint** PK\_t\_emp **PRIMARY KEY** (id)

);

### 2.1.3 创建外码约束

本关要求在数据库MyDb中新建两个表，并建立合适的外码约束。本官需要我们了解外码约束的概念以及用CONSTRAINT关键字创建约束的方法。核心代码如下：

**constraint** FK\_staff\_deptNo **foreign key**(deptNo) **references** dept(deptNo)

### 2.1.4 CHECK约束

本关要求在数据库MyDb中新建表，并且为其创建需要的CHECK约束，关键还是在于用CONSTRAINT关键字创建CHECK约束的语法。核心代码如下：

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

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

### 2.1.5 DEFAULT约束

本关要求在数据库MyDb中新建表，并且为其创建需要的DEFAULT约束为字段指定默认值，关键在于default关键字的使用。核心代码如下：

mz char(16) **default** '汉族'

### 2.1.6 UNIQUE约束

本关要求在数据库MyDb中新建表，并且为其创建需要的UNIQUE约束以保证字段取值的唯一性，关键在于unique关键字的使用。核心代码如下：

ID char(18) **unique**

## 2.2 表结构与完整性约束的修改（ALTER）

在本节中，需要我们在实践中掌握MySQL中表结构与完整性约束的修改方法，重点是掌握关键字ALTER的使用，这一部分的相关知识在Educoder平台上有很完备的介绍，整体难度较低。

### 2.2.1 修改表名

本关要求我们使用ALTER关键字修改表名，主要是学习ALTER关键字中的rename方法的使用方法，核心代码如下所示。

**alter** **table** your\_table rename my\_table;

### 2.2.2 添加与删除字段

本关要求我们使用ALTER关键字对表中的字段进行增加或删除，重点是学习ALTER TABLE的语法，同时学习其ADD和DROP方法的使用方法，核心代码如下所示。

#语句1：删除表orderDetail中的列orderDate

**alter** **table** orderDetail **drop** orderDate;

#语句2：添加列unitPrice

**alter** **table** orderDetail **add** unitPrice numeric(10,2);

### 2.2.3 修改字段

本关要求我们使用ALTER关键字对表中的字段进行修改，这里仅仅要求我们修改一个列名以及修改一列的数据类型，主要要求我们掌握这两种动作的实现方法，关键在于modify关键字和rename关键字的相关语法，核心代码如下所示。

**alter** **table** addressBook **modify** QQ char(12),rename column weixin **to** wechat;

### 2.2.4 添加、删除与修改约束

本关要求我们掌握用ALTER TABLE语句对主码约束、外码约束、CHECK约束、UNIQUE约束的添加、删除与修改的方法，关键在于add和drop关键字的使用，核心代码如下所示。

**alter** **table** Staff **add** **primary key**(staffNo);

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

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

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

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

## 2.3 基于金融应用的数据查询（select）

在本节中，需要我们在实践中掌握MySQL中数据查询的方法，关键在于学习select的使用，这一部分中的一些题目在逻辑上比较复杂，可以说是本次实训中花费时间最多也最难的。这一部分主要涉及到数据查询中的单表查询、连接查询、嵌套查询、集合查询、基于派生表查询等常见的查询方法。

### 2.3.1 查询客户主要信息

本关是简单的单表查询，逻辑上比较简单，要求对最终结果进行排序，这就使用到了MySQL中的order by对结果进行排序，最终代码如下所示。

**select** c\_name,c\_phone,c\_mail **from** client **order by** c\_id;

### 2.3.2 邮箱为null的客户

本关是简单的单表查询，逻辑上比较简单要求查找邮箱字段值为null的客户，这就涉及到了where子句的用法，值得注意的是在判断邮箱字段值是否为null的时候需要使用is null。最终代码如下所示。

**select** c\_id,c\_name,c\_id\_card,c\_phone **from** client **where** c\_mail **is** **null**;

### 2.3.3 既买了保险又买了基金的客户

本关需要涉及到嵌套查询和多条件查询，由于我们需要查找同时具有保险和基金两种资产的客户，所以需要判断当前的客户在property表中是否同时具有这两种类型的资产。这里我使用的是exists关键字来判断存在性，最终需要用order by进行排序。最终代码如下所示。

**select** c\_name,c\_mail,c\_phone **from** client **where**(**exists**(**select** 1 **from** property **where** c\_id**=**pro\_c\_id **and** pro\_type**=**2) **and** **exists**(**select** 1 **from** property **where** c\_id**=**pro\_c\_id **and** pro\_type**=**3)) **order by** c\_id;

### 2.3.4 办理了储蓄卡的客户信息

本关需要涉及到多表连接和条件查询。因为题目要求我们查找办理了储蓄卡的客户相关信息，我们需要将表client和表bank\_card按客户id进行等值连接，找出其中银行卡类型为储蓄卡的元组，最后用order by按照客户id进行排序即可，最终代码如下所示。

**select** c\_name,c\_phone,b\_number **from** client,bank\_card **where** c\_id**=**b\_c\_id **and** b\_type**=**'储蓄卡' **order by** c\_id;

### 2.3.5 每份金额在30000~50000之间的理财产品

本关是单表查询，但是涉及到了between关键字和按多个字段升序降序排序的知识。我们需要对finances\_product进行单表查询，在where子句中用between and关键字查询金额在指定范围内的产品，然后根据题目中的要求按金额升序排序，金额相同时按理财年限降序排序，这就涉及到了order by子句的高级用法，也就是按多字段排序以及使用关键字desc进行降序排序，最终代码如下所示。

**select** p\_id,p\_amount,p\_year **from** finances\_product **where** p\_amount **between** 30000 **and** 50000 **order by** p\_amount,p\_year **desc**;

### 2.3.6 商品收益的众数

本关要求求众数，是一个比较典型的数据查询需求，涉及到子查询、分组统计、COUNT()函数、ALL关键字等用法。首先我们需要对表property用group by关键字进行分组统计，分组统计依据就是商品收益字段pro\_income，然后使用having子句筛选出各分组内元组个数最多的分组，这一过程需要使用关键字all、子查询、COUNT()函数来实现，最终代码如下所示。

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

**group by** pro\_income

**having** **count**(\*) **>=** all(**select** **count**(\*) **from** property **group by** pro\_income);

### 2.3.7 未购买任何理财产品的武汉居民

本关涉及到用LIKE进行模糊匹配、子查询、not exists谓词等知识的使用。首先我们需要使用关键字LIKE根据身份证号前四位匹配出武汉居民，然后通过子查询查找出购买了理财产品的客户id，也就是在property表中进行查找，最后用not exists谓词对该结果进行查找，也即得到了没有购买过理财产品的居民，最终代码如下所示。

**select** c\_name,c\_phone,c\_mail **from** client **where**

c\_id\_card **like** '4201%' **and** **not** **exists**(**select** 1 **from** property **where**

c\_id**=**pro\_c\_id **and** pro\_type **=** 1) **order by** c\_id;

### 2.3.8 持有两张以上信用卡的用户

本关涉及到子查询和分组统计的知识。首先通过一个子查询查找bank\_card表，根据该卡的持有者进行分组统计，也就是使用了关键字group by，同时也要用where查询出信用卡，通过COUNT()函数获取分组内信用卡的数量，大于等于2的时候就是我们需要的用户信息。最后在外层查询中使用关键字in判断当前的用户id是否在我们子查询结果中并用order by进行排序即可，最终代码如下所示。

**select** c\_name,c\_id\_card,c\_phone **from** client **where**

c\_id **in** (**select** b\_c\_id **from** bank\_card **where** b\_type**=**'信用卡'

**group by**

b\_c\_id

**having**

**count**(\*) **>=**2) **order by** c\_id;

### 2.3.9 购买了货币型基金的客户信息

本关涉及到嵌套查询和exists关键字的用法，本身逻辑上比较简单，只需要根据用户的id查找到购买了基金并且同时购买了货币型基金的用户，将最终的结果用order by按照id排序即可，最终代码如下所示。

**select** c\_name,c\_phone,c\_mail **from** client

**where** **exists**(**select** 1 **from** property **where** c\_id**=**pro\_c\_id **and** pro\_type**=**3

**and** **exists**(**select** 1 **from** fund **where** f\_id**=**pro\_pif\_id **and** f\_type**=**'货币型'))

**order by** c\_id;

### 2.3.10 投资总收益前三名的客户

本关的逻辑相对比较复杂，涉及到子查询、等值连接、分组统计、limit等知识。根据任务要求，总收益需要排除被冻结的资产，因此我们首先通过一个子查询得到一个客户id和客户总收益的表，需要排除掉冻结资产，用group by按照id进行分组统计，然后对新表和旧表进行等值连接，按照总资产进行降序排序，最终用limit 3获取投资总收益前三名的客户，这里也涉及到了用关键字as进行起别名的作用，最终代码如下所示。

**select** c\_name,c\_id\_card,total\_income **as** total\_income **from** client, (**select** pro\_c\_id,**sum**(pro\_income) **as** total\_income **from** property **where** pro\_status**!=**'冻结' **group by** pro\_c\_id) **as** newpro **where** c\_id**=**newpro.pro\_c\_id **order by** total\_income **desc** **limit** 3;

### 2.3.11 黄姓客户持卡数量

本关的逻辑相对复杂，涉及到外连接、分组统计、模糊匹配等知识。本关的一个坑点在于黄姓客户持卡数量可能是0，如果使用等值连接去查询的话这些客户信息全都会被舍弃掉，所以这里需要根据用户id使用左外连接连接表client和表bank\_card。此后需要通过like找出所有黄姓用户，然后根据用户id分组用COUNT()函数统计持卡数量，按照要求进行排序即可，这里也使用了ifnull关键字进行判定，最终代码如下所示。

**select** c\_id,c\_name,**ifnull**(**count**(b\_number),0) **as** number\_of\_cards **from** client **left join** bank\_card **on** c\_id**=**b\_c\_id **where** c\_name **like** '黄%' **group by** c\_id **order by** number\_of\_cards **desc**,c\_id;

### 2.3.12 客户理财、保险与基金投资总额

本关在撰写上比较复杂，但是在逻辑上其实是比较简单的，涉及到了多个表之间的关系，涉及到分表合并、分组连接、外连接等知识，需要多个子查询根据题目中的要求去对总额进行计算，也就是首先用三个分别的子查询得到三种投资产品的总额，然后再把这三个总额加起来即可，比较关键的点是和任务2.3.11一样，需要注意对外连接的使用，否则可能会因为某个用户没有其中一项投资产品而出现错误。具体代码实现由于篇幅原因此处略过。

### 2.3.13 客户总资产

本关要求我们查询客户在某个银行的总资产，涉及到多表连接查询、嵌套查询、衍生表的用法。核心的思路和2.3.13差不多，也是代码撰写上比较复杂，但是在逻辑上还是相对简单的，任务书中给出了总资产的计算方法，只需要按照这个计算方法来即可。由于这里定义的总收益是储蓄卡总余额、投资总额、投资总收益的和，所以我们需要用多个子查询来计算这些值，需要注意的事项和2.3.12也差不多。具体代码实现由于篇幅原因此处略过。

### 2.3.14 第N高问题

本关涉及到的主要知识有排序、去重、取第N高等，本身的逻辑比较简单，首先通过子查询找出第4高的保险金额，具体思路就是将表insurance查询，用distinct关键字去重之后用limit关键字找出第4高的保险金额，这里采用了limit+offset关键字的组合用法，在外层查询中再对表insurance进行查询即可，这层查询是为了找出保险金额等于内层子查询结果的元组，最终代码如下所示。

**select** i\_id,insurance.i\_amount **from** (**select distinct** i\_amount **from** insurance

**order by** i\_amount **desc** **limit** 1 offset 3) **as** temp,insurance **where** temp.i\_amount **=**insurance.i\_amount;

### 2.3.15 基金收益两种方式排名

该任务关卡跳过。

### 2.3.16 持有完全相同基金组合的客户

该任务关卡跳过。

### 2.3.17 购买基金的高峰期

该任务关卡跳过。

### 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

该任务关卡跳过。

### 2.3.19 以日历表格式显示每日基金购买总金额

该任务关卡跳过。

## 2.4 数据查询(Select)之二

本节的核心内容仍然是MySQL中的数据查询相关知识，主要是对2.3节的补充，题目在逻辑上更加复杂。

### 2.4.1 查询销售总额前三的理财产品

本关要求实现查询销售总额前三的理财产品，涉及到了嵌套查询、over(partition by … order by)等知识点，这里在具体实验中还使用了MySQL中的year函数。本题在逻辑上其实并不复杂，需要把该算的东西算出来，重点在于对over(partition by … order by)语句的使用。最终代码如下所示。

**select** **year**(pro\_purchase\_time) pyear,**rank**() **over**(**partition** **by** **year**(pro\_purchase\_time)**order by** p\_amount \* pro\_quantity **desc**)rk,p\_id,p\_amount\*pro\_quantity sumamount

**from** finances\_product,property **where** pro\_pif\_id**=**p\_id **and** pro\_type**=**1 **and** **year**(pro\_purchase\_time) **in** (2010,2011) **limit** 0,6;

### 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

该任务关卡跳过。

### 2.4.4 查找相似的理财产品

该任务关卡跳过。

### 2.4.5 查询任意两个客户的相同理财产品数

该任务关卡跳过。

### 2.4.6 查找相似的理财客户

该任务关卡跳过。

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

本节主要需要我们在实践中学习MySQL中数据的插入、修改和删除的一些方法，重点在于对关键字Insert、Update、Delete的学习和使用，本身在逻辑上难度较低。

### 2.5.1 插入多条完整的客户信息

本任务需要我们用insert语句向表client中插入几条完整的数据，只需要按照表的格式来即可，难度较低，下面仅给出插入一条完整数据的代码。

**insert into** client

**values**(

    1,

    '林惠雯',

    '960323053@qq.com',

    '411014196712130323',

    '15609032348',

    'Mop5UPkl'

);

### 2.5.2 插入不完整的客户信息

相比于2.5.1，本任务需要我们仅仅插入一条客户信息中的部分字段，这个时候就需要指定插入的信息属于哪一列，仍然需要使用insert指令，最终代码如下所示。

**insert into** client(

    c\_id, c\_name, c\_id\_card, c\_phone, c\_password

)

**values**(

    33,

    '蔡依婷',

    '350972199204227621',

    '18820762130',

    'MKwEuc1sc6'

);

### 2.5.3 批量插入数据

本任务涉及到了insert较高级的用法，需要将另一个和表client格式一致的表中的数据批量导入到表client中，最终代码如下。

**insert into** client(

    c\_id, c\_name, c\_mail, c\_id\_card, c\_phone, c\_password

)

**select** \* **from** new\_client;

### 2.5.4 删除没有银行卡的客户信息

本任务涉及到了delete的用法，主要是通过not exists确定哪些客户没有银行卡，然后把这些用户删除即可，最终代码如下。

**delete** **from** client **where** **not** **exists**

(**select** 1 **from** bank\_card **where** b\_c\_id**=**c\_id);

### 2.5.5 冻结客户资产

本任务涉及到了update的用法，主要是用来更新字段的内容。本任务要求冻结指定手机号的客户的资产，所以我们需要先通过一个查询获取手机号，然后用关键字in判断指定的手机号是否在查询到的手机号中，最终代码如下。

**update** property

**set** pro\_status**=**'冻结'

**where** '13686431238' **in** (**select** c\_phone **from** client **where** c\_id**=**pro\_c\_id);

### 2.5.6 连接更新

本任务需要根据client表中的身份证号信息更新property表中的身份证号信息，最终代码如下。

**update** property,client **set** pro\_id\_card**=**c\_id\_card **where** c\_id**=**pro\_c\_id;

## 2.6 视图

本节中开始涉及到视图相关的知识，合理的使用视图可以大大简化一些SQL操作。本节首先将介绍视图的创建方法，并且将基于视图进行数据查询，让我们在实践中感受到在一些场景下合理使用视图可以简化SQL操作的复杂性，从而写出更易读的SQL代码。

### 2.6.1 创建所有保险资产的详细记录视图

本任务涉及到用create语句创建视图的知识，需要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，也就是说需要先用一个select语句得到这些信息，然后使用create view创建视图，最终代码如下所示。

**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,insurance,client

**where** c\_id**=**pro\_c\_id **AND** pro\_type**=**2 **AND** i\_id**=**pro\_pif\_id;

### 2.6.2 基于视图的查询

本任务需要基于2.6.1中创建的视图v\_insurance\_detail做一些查询，重点是要计算保险投资总额和保险投资总收益这两个字段，也就是需要进行分组统计，需要根据身份证号这一字段进行分组，最终计算出这些信息之后再按照保险投资总额降序排列即可，最终代码如下所示。

**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**;

## 2.7 存储过程与事务

本节主要涉及到了存储过程这一知识。存储过程是一种可编程的数据对象，可以让传统的SQL语言拥有更加强大的功能，使得SQL代码编写的过程更加规范，具有模块化的特点。本节将主要介绍用三种控制结构进行存储过程的构造，即流程控制语句、游标、事务。此外，由于这部分代码较长，所以仅仅介绍思路，而不直接粘贴代码。

### 2.7.1 使用流程控制语句的存储过程

本任务让我们用流程控制语句实现斐波那契数列的求解，也就是向表fibonacci中插入斐波那契数列的前n项，需要我们掌握使用流程控制语句的存储过程的编写。其实这里和常见的编程语言的编写已经很类似了，用递推公式来迭代计算斐波那契数列的下一项，每计算出一项就将它插入到表fibonacci中即可，当然在最开始需要加入一些特判。

### 2.7.2 使用游标的存储过程

本任务在逻辑上相对复杂一点，其实是一个用SQL的模拟过程，只需要将整个排班要求模拟出来即可，这里用一个循环变量i遍历从start\_date到end\_date之间的所有日期，用MySQL自带的dayofweek来获取这个日期具体是周几，注意dayofweek的返回和我们的常识有点区别。如果当前是周末并且轮到的医生是主任医生的话，就设置一个flag表示发生了这种特殊情况，然后读出下一个医生的信息，对其进行排班；如果当前是周一的话就看一下之前设置的flag，看一下有没有主任医生在之前进行了轮班，如果有的话就对其进行排班；否则就按最正常的进行排班即可。

### 2.7.3 使用事务的存储过程

本任务涉及到了事务的定义和应用相关的知识，用的是一个非常典型的银行转账的例子，事务定义的核心大概就是模拟，也就是按照任务中给出的注意事项来确定此时的转账能否发生，如果不能的话就rollback，否则就commit。

## 2.8 触发器

本节主要涉及到了触发器的相关知识，触发器是与某个表绑定的命名存储对象，与存储过程一样，它由一组语句组成，当这个表上发生某个操作(insert,delete,update)时，触发器被触发执行，这一部分编写的代码也比较长，所以仅仅叙述核心思路。

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

本任务需要我们用触发器实现一套完整性业务规则，在进行INSERT、DELETE、UPDATE等操作时激活触发器对数据进行检查，看看它是否符合业务规则，涉及到了MySQL中触发器的创建方法、抛出异常的方法、字符串拼接函数CONCAT()的使用。

主要的流程在任务书中已经写的比较明确了，也就是对new表中的pro\_type进行判断：如果不是我们预设的类型的话，用CONCAT()函数构造违法信息，用signal sqlstate抛出异常；如果是类型1的话，查看pro\_pif\_id是不是finances\_product中的某个主码值，如果不是就抛出异常，具体可以用关键字not exists进行判断；如果是类型2或类型3的话，只需要用类似的方法对约束进行检查即可，此处不再赘述。

## 2.9 用户自定义函数

本节中需要我们学习MySQL中创建并调用一个自定义函数的方法，自定义函数在创建之后就可以像库函数一样使用它，从而进一步提升了编写SQL语句的模块化程度，大大提升编程效率。

### 2.9.1 创建函数并在语句中使用它

本任务涉及到MySQL中函数的定义以及在select语句中应用自定义函数的方法，关键是利用create function语句创建函数的方法，这和我们以往学习的高级程序语言类似，创建的函数有参数也有返回值，只需要按照语法规则编写即可。我们这里需要编写一个计算所有储蓄卡余额的函数，只需要用select进行单表查询并用sum()进行计算即可，函数定义代码如下。

**create** **function** **get\_deposit**(client\_id int) **returns** numeric(10,2)

**begin**

**declare** ans numeric(10,2);

**select** **sum**(b\_balance) **into** ans **from** bank\_card **where** b\_type**=**'储蓄卡'

**and** b\_c\_id**=**client\_id;

**return** ans;

**end**$$

调用方法和调用库函数类似，调用部分的代码如下。

**select**  c\_id\_card,c\_name,get\_deposit(c\_id) **as** total\_deposit **from** client

**where** get\_deposit(c\_id)**>=**1000000 **order by** get\_deposit(c\_id) **desc**;

## 2.10 安全性控制

本节主要涉及到了MySQL中的安全性控制相关的知识，介绍了MySQL中的用户、角色、权限等方面的知识。在实践中我们可以掌握MySQL中用户、角色、权限的创建和回收，并且通过这些安全性控制手段防止数据泄露、更改或者被破坏。

### 2.10.1 用户和权限

本任务要求我们在金融应用场景数据库环境中，创建用户，并给用户授予指定权限，涉及到了MySQL中的安全性控制机制、CREATE USER语句的使用、GRANT和REVOKE语句的使用等知识。本任务的介绍中详细给出了这些知识的解释以及使用方法，所以在具体实现上难度并不算很大。最终代码如下。

#(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\_name,c\_mail,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';

### 2.10.2 用户、角色与权限

本关我们需要创建角色，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。相关的知识在2.10.1节中已经介绍过，这里主要涉及到了用create role语句创建角色。最终代码如下。

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

**create** **role** client\_manager;

**create** **role** 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\_type,b\_c\_id,b\_number) **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;

## 2.11 并发控制与事务的隔离级别

本节主要涉及到了MySQL中的并发控制与事务隔离级别相关的知识。在并发操作中，很有可能出现数据不一致性，如丢失修改、读脏数据、不可重复读、幻读等数据不一致性。为解决上述不一致性问题，DBMS设计了专门的并发控制子系统，采用封锁机制进行并发控制，以保证事务的隔离性和一致性，本节我们将会学习到这些机制。

### 2.11.1 并发控制与事务隔离级别

本任务需要将事务的隔离级别设置成read uncommitted，根据任务中的知识介绍，只需要学会MySQL中事务隔离级别的设置方法即可。核心代码如下。

**set** **session** **transaction** **isolation** **level** **read** **uncommitted**;

### 2.11.2 读脏

脏数据实际上是一个事务读取到了另一个事务修改后的数据，而另一个事务后来撤销了本次修改。本任务要求我们设置合适的事务隔离级别，构造出读脏这一现象。实际上为了演示读脏现象，我们需要把事务隔离级别设置成最低级别READ UNCOMMITTED。为了控制事务执行的次序，我们需要借助sleep语句，也就是让事务1开始时sleep1秒，事务2对数据发生修改后sleep2秒再rollback，从而构造出读脏这种情况，代码此处略过。

2.11.3 不可重复读

不可重复读是指一个事务读取到某数据后，另一个事务修改了该数据，后来当第一个事务再次读取该数据的时候，发现两次读取的数据不一致。本任务要求我们设置合适的事务隔离级别，构造出不可重复读这一现象。根据分析，我们可以将事务隔离级别设置成READ UNCOMMITTED，然后设置合适的事务执行次序即可。最开始先让事务1读一次数据，事务2sleep1秒，事务1读过之后直接sleep2秒，事务2读一次之后再次sleep2秒，这个时候事务1会对数据进行修改，然后sleep3秒，然后事务2再读一次数据，发现数据不一致，产生了不可重复读现象。

### 2.11.4 幻读

幻读是在某一事物多次读取数据之间，另外一个事务对数据集进行了insert或delete。本任务要求在较高隔离级别repeatable read下重现幻读现象。本关实现起来比较简单，只需要让事务1在第一次查询之后sleep2秒即可，最终代码如下。

**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**;

### 2.11.5 主动加锁保证可重复读

本任务要求在事务隔离级别较低的read uncommitted情形下，通过主动加锁，保证事务的一致性。主要需要我们掌握MySQL中的锁机制，以及通过任务2.11.3中类似的方法控制并发顺序来尝试触发不可重复读，最终发现通过主动加锁的方式可以保证可重复读。代码略过。

### 2.11.6 可串行化

本任务要求选择除serializable(可串行化)以外的任何隔离级别，保证两个事务并发执行的结果是可串行化的。本关的实现较为简单，核心思路就是在合适的地方用sleep控制顺序以及在合适的地方加锁即可，代码此处略过。

## 2.12 备份+日志：介质故障与数据库恢复

本节主要涉及到了MySQL中的介质故障与恢复服务。为了保证数据库软件在实际生产环境中的高可用性，如今的数据库软件都提供了一套备份恢复机制，也就是在发生如存储介质故障这样的问题的时候可以利用数据库备份和日志文件来恢复数据库，解决这些问题，提高数据安全性和可靠性。

### 2.12.1 备份与恢复

本任务主要涉及了MySQL中提供的一些备份与恢复工具，这里主要使用了MySQL dump工具来对MySQL进行备份。整体逻辑比较简单，因为任务书中给出了这部分知识的用法，需要先用MySQL dump将数据库备份，然后再用备份文件重启数据库，最终代码如下。

# 对数据库residents作海量备份,备份至文件residents\_bak.sql:

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

# 利用备份文件residents\_bak.sql还原数据库:

mysql -h127.0.0.1 -uroot **<** residents\_bak.sql

### 2.12.2 备份+日志：介质故障的发生与数据库的恢复

本任务模拟了介质故障的发生，介绍了利用备份和备份之后的日志恢复数据库的方法，用到的工具仍然是MySQL dump，不过增加了工具MySQL binlog的使用。在备份的时候，相比2.12.1需要使用—flush-logs刷新日志，代码如下所示。

mysqldump -h127.0.0.1 -uroot --*flush-logs --databases train > train\_bak.sql*

随后需要用备份的文件恢复数据库，代码如下所示。

mysql -h127.0.0.1 -uroot **<** train\_bak.sql

mysqlbinlog --*no-defaults log/binlog.000018 | mysql -uroot*

## 2.13 数据库设计与实现

本节主要涉及了数据库的设计与实现的过程，主要是对实际生产环境中的场景进行建模。任务内容设计了从概念模型到MySQL实现、从需求分析到逻辑模型、建模工具的使用三个部分，囊括了将实际生产环境中的场景抽象成数据库表再到最终应用到MySQL数据库中的过程。

### 2.13.1 从概念模型到MySQL实现

本任务中给出了场景的E-R图，要求根据概念模型建立实际的MySQL表，只需要根据概念模型一步步来把整个数据库表建立完成即可，涉及到的代码较长，但是逻辑不复杂，此处省略。

### 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

本任务需要学习数据库的设计与实现过程中常见的建模工具。任务文档中介绍了几种常用的建模工具，最终任务实现要求使用MySQL Workbench，使用该工具将任务文档中给出的模型文件导出成SQL脚本即可，整体过程较为简单。

### 2.13.4 制约因素分析与设计

根据本节任务，可以看到在将现实生产场景建模形成数据库的概念模型和逻辑模型的过程中，有很多制约因素需要考虑。以2.13.1中的机票购买系统为例，我们需要根据这个场景考虑各个量之间的制约关系，比如机票信息会包括乘坐人信息和购买人信息，并且这其中也需要考虑系统权限上的要求，比如对于机票购买系统，很典型的就有普通用户和管理员用户等，普通用户仅仅有正常的机票购买权限，管理员用户可以对整个系统进行运维。常见的生产场景都有类似的约束，我们需要综合考虑之后将这些约束转换成数据库中具体的设计。

### 2.13.5工程师责任及其分析

工程师应当承担起各个方面相应的责任。在社会方面，工程师应该结合自身的工程背景评价某一项工程的具体意义，探究其落地对各个方面的实际影响；在安全方面，工程师应该合理运用自己的工程知识，认真分析系统中存在的潜在安全问题，保证系统的健壮性。总的来说，工程师应该能够结合科学原理设计可靠的系统来推动社会发展。

## 2.14 数据库应用开发(JAVA篇)

本节要求用java进行数据库应用开发，任务基于JDBC这组用于执行Java语句的Java API。本节需要我们掌握JDBC的体系结构和一些对数据库的简单操作，学习实际生产中后端的一些CRUD操作。本节中任务书给出的知识介绍非常详细，即使是对于没有Java基础的人来说想要完成本节的实验也比较轻松。

### 2.14.1 JDBC体系结构和简单的查询

本任务需要我们了解JDBC体系结构并且对于用JDBC操作数据库的具体方法有简单的了解，具体要求我们查询client表中邮箱非空的客户，列出相关信息。利用JDBC开发的主要过程大概是首先创建数据库连接对象、创建Statement对象、用Statement对象执行SQL语句、借助ResultSet对结果进行输出。这里大部分步骤在任务书中都有示例代码，因此我们只需要了解这些代码的实际含义并且按照任务书中的方式写出来即可，并且需要执行的SQL语句也比较简单，核心代码如下所示。

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

String USER="root";

String PASS="123123";

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

connection=DriverManager**.getConnection**(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**.print**("姓名\t邮箱\t\t\t\t电话\n");

**while**(resultSet**.next**()){

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

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

    System**.**out**.print**(resultSet**.getString**("c\_phone")+"\n");

}

### 2.14.2 从需求分析到逻辑模型用户登录

本任务要求我们在正确使用JDBC的基础上进行条件不确定的查询。当待执行的SQL语句中带有变化的部分时，通常有两种解决方案，一是可以将变量直接拼到SQL语句中，二是PreparedStatement类把SQL语句中变化的部分当成参数。这里使用了第一种方法，输入的内容是用户的邮箱和密码，我们需要根据该信息作为登录信息，查看信息是否正确，输出相应的提示，整体难度较低。核心代码如下所示。

statement=connection**.createStatement**();

String SQL="select c\_password from client where c\_mail='"+loginName+"';";

resultSet=statement**.executeQuery**(SQL);

**if**(resultSet**.next**())

**if**(resultSet**.getString**("c\_password")**.equals**(loginPass)){

        System**.**out**.print**("登录成功。\n");

    }**else**{

        System**.**out**.print**("用户名或密码错误！\n");

    }

**else** System**.**out**.print**("用户名或密码错误！\n");

### 2.14.3 添加新客户

本任务要求我们掌握借助JDBC向表client中插入数据的方法。根据任务书，我们只需要在成功连接数据库后实例化PreparedStatement类的一个对象，设置相应信息，然后调用对象的executeUpdate()方法即可，这里任务书中给出了用PreparedStatement的执行插入语句的方法，因此我们这里也使用该类进行插入数据，逻辑比较简单，核心代码如下所示。

PreparedStatement pps=**null**;

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

pps=connection**.prepareStatement**(SQL);

pps**.setInt**(1,c\_id);

pps**.setString**(2,c\_name);

pps**.setString**(3,c\_mail);

pps**.setString**(4,c\_id\_card);

pps**.setString**(5,c\_phone);

pps**.setString**(6,c\_password);

n=pps**.executeUpdate**();

### 2.14.4 银行卡销户

本任务要求我们编写一个能删除指定客户编号的指定银行卡号的方法，需要我们掌握Statement或PreparedStatement类的应用。删除的方法和插入的方法类似，完全可以参考2.14.3的实现，逻辑比较简单，核心代码如下所示。

PreparedStatement pps=**null**;

int n=0;

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

pps=connection**.prepareStatement**(sql);

pps**.setInt**(1,b\_c\_id);

pps**.setString**(2,b\_number);

n=pps**.executeUpdate**();

### 2.14.5 客户修改密码

本任务要求我们编写客户登录修改密码的方法，具体的方法和前面几关都类似，使用PreparedStatement类即可，总体难度不大，需要注意任务书里的要求还是比较多的。核心代码如下所示。

PreparedStatement pps=**null**;

ResultSet rs=**null**;

int n=-1;

String SQL1="select c\_password from client where c\_mail=?;";

pps=connection**.prepareStatement**(SQL1);

pps**.setString**(1,mail);

rs=pps**.executeQuery**();

**if**(**!**rs**.next**()) n=2;

**else**{

**if**(rs**.getString**("c\_password")**.equals**(password)**==false**){

        n=3;

    }**else**{

        String SQL2="update client set c\_password = ? where c\_mail=?;";

        pps=connection**.prepareStatement**(SQL2);

        pps**.setString**(1,newPass);

        pps**.setString**(2,mail);

        n=pps**.executeUpdate**();

    }

}

### 2.14.6 事务与转账操作

本任务要求我们编写一个银行卡转账的方法，主要涉及到JDBC的事务处理。其实核心的逻辑和之前完成的并发控制任务相关的内容差不多，只是需要借助JDBC对数据库进行操作，其实只需要调用commit和rollback即可，逻辑比较相似，相关代码此处略过。

### 2.14.7 把稀疏表格转为键值对存储

本任务要求我们编写一个程序把稀疏表格转为键值对存储，主要涉及到对JDBC的综合运用，需要把entrance\_exam表的值转写到sc表，整体的逻辑也比较简单，只需要执行一条查询语句把entrance\_exam表的值读出来然后插入到SC表中即可，核心代码如下所示，注意由于篇幅关系这里仅仅涉及到了一部分转换。

String sql = "select \* from entrance\_exam;";

rest = stmt**.executeQuery**(sql);

**while**(rest**.next**()){

    int politics = rest**.getInt**("politics");

**if**(politics **!=** 0){

**insertSC**(**conn**,**sno**,"politics",**politics**);

    }

}

## 2.15 数据库的索引B+树实现

本节需要我们实现B+树。索引是数据库中重要的优化环节，可以大大减少查询数据时消耗的时间，B+树是在建立索引时比较常用的数据结构，有着稳定高效的优化能力。通过本节实验，我们将会了解到实现B+树实现的具体细节。

### 2.15.1 BPlusTreePage的设计

本关的实现比较简单，只需要补全BPlusTreePage类的一些成员函数即可，根据注释就可以知道这个函数的功能。具体实现略过。

### 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

本次实验以MySQL数据库为例进行数据库应用的实践操作，涉及到了数据库理论知识的方方面面。在所有任务中，除了实验三数据查询之一的后五个关卡、实验四数据查询之二的后五个关卡、实验十三数据库设计与实现的第二个关卡、实验十五数据库的索引B+树实现的后四个关卡没有完成之外，其余关卡全部完成，并且通过了educoder的测试。

在本次实验中，我成功基于MySQL数据库完成了大部分实验任务。在实训一与实训二中，我学习了用Create和Alter语句对数据库、表、约束进行创建和修改；在实训三、实训四与实训六中，我学习了数据查询的各种知识点，如单表查询、嵌套查询、分组统计等，也学习了用视图简化查询思考量的方法，这一部分可以说是整个实验最为复杂的部分；在实训五中，我学会了用Insert、Delete、Update对数据库中的数据进行修改的方法；在实训七、实训八与实训九中，我学习了MySQL中更为结构化的编程方法；在实训十与实训十二中，我学会了MySQL中通过权限、角色来对数据安全性进行控制的方法，学会了赋予用户权限的方法，同时也学会了在介质故障的情况下如何通过相应的工具对数据库进行恢复；在实训十一中，我学会了MySQL中并发控制与事务隔离级别相关的知识，主要通过控制事务执行次序构造出了种种不一致现象；在实训十三中，我学会了数据库的设计方法，学会了将实际生产场景建模成数据库的过程，学习了常见的建模工具的使用；在实训十四中，我学会了用JDBC对数据库进行操作的方法；在实训十五中，我学习了常见的数据库索引B+树的设计。

通过本次实验，我对理论课上学习的数据库相关的知识有了更进一步的理解，同时也接触了用Java对数据库进行操作的方法。本次实验的大部分任务的难度都相对较小，因为在educoder平台上给出了非常详细的知识介绍，只需要看懂了相关的知识具体讲了什么，就可以很容易的通过这个关卡，当然也有一些难度比较大的任务，特别是数据查询的两个实训，编写的代码相对会复杂很多，但是任务书上也给出了要完成这部分内容需要涉及到的知识，这在一定程度上降低了难度，即使最开始不会MySQL中的一些语句的用法，也可以在网上很容易的查到。总之，本次实验带给我的收获很大，我对MySQL数据库的使用方法有了更进一步的理解。