

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | 计卓2101 |
| 学 号： | U202112071 |
| 姓 名： | 王彬 |
| 指导教师： | 谢美意 |

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

2023 年 6 月 27 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](#_Toc138845145)

[2 任务实施过程与分析 2](#_Toc138845146)

[2.1 数据库、表与完整性约束的定义(Create) 2](#_Toc138845147)

[2.2 表结构与完整性约束的修改(ALTER) 3](#_Toc138845148)

[2.3 数据查询(Select)之一 3](#_Toc138845149)

[2.4 触发器 7](#_Toc138845150)

[2.5 数据库设计与实现 7](#_Toc138845151)

[2.6 数据库应用开发(JAVA篇) 9](#_Toc138845152)

[2.7 数据库的索引B+树实现 10](#_Toc138845153)

[3 课程总结 11](#_Toc138845154)

# 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。

具体实验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)。

本课程将任务分为15个实训单元，依托头歌实践教学平台，对数据库、表、完整性约束，查询，插入、修改和删除等数据库环节进行实践支撑。

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

本次实践课程在头歌平台进行，实践任务均在平台上提交代码，所有完成的任务、关卡均通过了自动测评。本次实践最终完成了任务书中的2.1~2.10、2.13~2.15子任务，下面将重点针对其中的2.3数据查询任务阐述其完成过程中的具体工作。

## 2.1 数据库、表与完整性约束的定义(Create)

本任务环节要求使用MySQL提供的数据定义语句对数据库、表与完整性约束进行创建。具体应使用相关定义语句、主码、外码、CHECK语句、DEFAULT语句和CONSTRAINT完整性约束语句完成上述内容。

本任务已完成1~6所有关卡。

### 2.1.1 创建外码约束(foreign key)

本题需要创建两个已经给定的表，为表定义主键，并给表staff创建外键，这个外键约束的名称为FK\_staff\_deptNo。在创建表之前需要先创建数据库MyDb，并且将两张表创建在MyDb数据库中。不需考虑关于性别的约束。

使用CREATE TABLE子句实现表的创建和完整性约束定义。其中，主码使用列级完整性约束定义，可在数据项定义后添加PRIMARY KEY约束；我们同时对外码创建表级参照完整性约束定义。其关键代码为：

**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)

);

### 2.1.2 CHECK约束

使用完整性约束可以对表实现CHECK约束定义，其具体实现为[CONSTRAINT [约束名]] CHECK (条件表达式)]。其关键代码为： **CONSTRAINT** CK\_products\_brand **CHECK**(brand **IN** ('A','B')),

**CONSTRAINT** CK\_products\_price **CHECK**(price > 0)

## 2.2 表结构与完整性约束的修改(ALTER)

本任务环节要求使用MySql提供的ALTER语句对数据库、表与完整性约束进行修改。具体应使用相应的ALTER语句修改表名、列名、列类型或列约束等。

### 2.2.1 添加或删除字段

我们使用ALTER语句为表添加新的字段或删除旧有字段，其具体语句为ALTER TABLE <表名> ADD [COLUMN] 列名 数据类型 [列约束] [FIRST | AFTER 列名]。其中，关键字FIRST指示新添加的列为第1列；AFTER指示新添加的列紧跟在指定列的后面。删除字段的语句则为ALTER TABLE 表名 DROP [COLUMN] 列名。其关键代码如下。

#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** unitPrice **NUMERIC**(10,2);

### 2.2.2 添加、删除与修改约束

对于约束的修改事项有多种分支，如对主码的删除与修改、对CHECK约束的删除与添加等。我们将为题中所给表添加主码和添加UNIQUE约束为例，分析对约束的增删改过程。

对主码约束的修改，可以使用ALTER TABLE <表名> ADD CONSTRAINT <约束名>指令进行实现；而特别地，添加UNIQUE约束则可以用ALTER TABLE <表名> ADD [CONSTRAINT [约束名]] UNIQUE(列1,...)。我们这两个关键操作的关键代码如下。

#(1) 为表Staff添加主码

**ALTER** **TABLE** Staff **ADD** **CONSTRAINT** **PRIMARY** **KEY**(staffNo);

#(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);

## 2.3 数据查询(Select)之一

### 2.3.1 查询既买了保险又买了基金的客户

本题为关卡3习题。

我们使用带有EXISTS的子查询，在property表中查询其自身连接中既含有保险的用户（pro\_type=2）又包含基金的用户（pro\_type=3），并投影出查询结果的pro\_c\_id。随后，在client表中使用上述嵌套搜索找到c\_id对应的目标字段元组，作为最终查询结果并返回。其完整代码如下。

**SELECT** c\_name,c\_mail,c\_phone

**FROM** client

**WHERE** c\_id **IN** (

**SELECT** pro\_c\_id

**FROM** property P

**WHERE** **EXISTS**(**SELECT** 1 **FROM** property **WHERE** pro\_c\_id=P.pro\_c\_id **AND** pro\_type=2)

**AND** **EXISTS**(**SELECT** 1 **FROM** property **WHERE** pro\_c\_id=P.pro\_c\_id **AND** pro\_type=3)

)

**ORDER** **BY** c\_id;

### 2.3.2 商品收益的众数

本题对应关卡6，要求查询资产表中所有资产记录中商品收益的众数及其出现次数。

众数即出现最多的数据记录，因此只需要对于property表使用group by子句作分组统计，对商品收益pro\_income进行分组，同时选择其组内统计元组个数比所有统计的元组个数都多的记录（即众数的定义）。

具体地，我们使用关键字ALL的子查询实现这一选择操作，并使用嵌套查询的方式，在子查询内得出所有pro\_income分组的元组个数。在外部查询中，分别取出符合条件的pro\_income和元组统计值即可。其完整代码如下。

**SELECT** pro\_income,**COUNT**(\*) presence

**FROM** property

**GROUP** **BY** pro\_income

**HAVING** **COUNT**(\*) >= **ALL**(

**SELECT** **COUNT**(\*)

**FROM** property

**GROUP** **BY** pro\_income);

### 2.3.3 购买了货币型基金的客户信息

本题对应关卡9，我们使用多层嵌套的方式进行求解。在子查询中，对property表和fund表中符合f\_type=’货币型’的元组进行连接。外层查询中，选择那些在子查询里id相等且子查询存在结果的，并将查询结果以c\_id进行升序排序。

**SELECT** c\_name,c\_phone,c\_mail

**FROM** client

**WHERE** **EXISTS**(

**SELECT** 1 **FROM** property, fund

**WHERE** pro\_c\_id=c\_id **AND** pro\_type=3 **AND** pro\_pif\_id=f\_id **AND** f\_type='货币型')

**ORDER** **BY** c\_id;

### 2.3.4 客户理财、保险与基金投资总额

本题对应关卡12，我们考虑使用分表合并、分表统计的方式进行查询。

首先，我们可以分别对客户理财、保险、投资的金额总和进行查询，其中的查询结果使用UNION运算进行合并。例如，如果对各个客户理财的金额总和进行统计，可以使用如下语句：

**SELECT** c\_id, c\_name, c\_id\_card,ifnull(**SUM**(pro\_quantity\*p\_amount),0) **AS** total\_sum

**FROM** client **LEFT** **JOIN** property **ON** c\_id=pro\_c\_id

**JOIN** finances\_product **ON** pro\_pif\_id=p\_id

**WHERE** pro\_type=1

**GROUP** **BY** c\_id

而我们对三种不同类型的投资金额总和进行统计后，需要合并入未成功查询的量，我们将未成功查询的数据项置为0，这是为了对没有查询成功的数据项实现从NULL至零的填充，以便之后对于c\_id进行分组查询。其语句如下：

**SELECT** c\_id,c\_name,c\_id\_card,0 **AS** total\_sum

**FROM** client

之后我们对所有的查询结果派生为tabl表，对该表进行分组和查询操作，可以使用下列语句：

**SELECT** c\_name,c\_id\_card,ifnull(**SUM**(total\_sum),0) **AS** total\_amount

**FROM** tabl

**GROUP** **BY** c\_id,c\_name,c\_id\_card

**ORDER** **BY** total\_amount **DESC**;

### 2.3.5 第N高问题

本题对应关卡14，查询每份保险金额第4高保险产品的编号和保险金额。

本问题中我们分别要考虑排序和消重。对于排序，则需要对insurance表的i\_amount数据项进行自身连接，其连接条件为前表该项的值小于等于后表，这样可以统计出比该数多的i\_amount值的数量。如此，只需要对tabl.i\_amount值进行分组，得到数量为4的即为第四高的保险产品。随后，在外层查询中，我们需要用IN关键词查询出所有和子查询结果相等的i\_amount对应的i\_id。其完整代码为：

**SELECT** i\_id,i\_amount

**FROM** insurance

**WHERE** i\_amount **IN** (

**SELECT** tabl.i\_amount **AS** i\_amount

**FROM** (**SELECT** tabl\_1.i\_amount **AS** i\_amount **FROM**

(**SELECT** **DISTINCT** i\_amount **FROM** insurance) **AS** tabl\_1

**JOIN** (**SELECT** **DISTINCT** i\_amount **FROM** insurance) **AS** tabl\_2

**ON** tabl\_1.i\_amount<=tabl\_2.i\_amount

) **AS** tabl

**GROUP** **BY** tabl.i\_amount

**HAVING** **COUNT**(\*)=4

);

### 2.3.6 持有完全相同基金组合的客户

本题对应关卡16，查询持有完全相同基金组合的客户。

持有完全相同基金组合的客户，即对于客户A所持有的基金，客户B也持有；反过来客户B所持有的所有基金，客户A同样持有。

因此我们对于每一组二元客户对(C, D)，对它们持有的基金进行检查。首先对于客户C，不存在D中的基金元组C不具有的，而且不存在C中的基金元组D不具有的。这种结构可以用两层NOT EXISTS运算实现，即在这两层NOT EXISTS中分别引入property表，检查后者的pro\_c\_id=D.c\_id、后者pro\_type应为基金型对应代码，而前后property表中的pro\_pif\_id应相等。

分别对两个“相互不存在”的关系进行编写后，同时还需要保证这样的客户是有选择基金的，亦即他的基金选择清单非空。因此需要用存在子句EXISTS保证property表中有客户C的基金记录。

本题的具体代码如下。

**SELECT** C.c\_id **AS** c\_id1, D.c\_id **AS** c\_id2

**FROM** client **AS** C **JOIN** client **AS** D **ON** C.c\_id < D.c\_id # 对(C, D)进行取值

# 不存在D中的基金元组C不具有的

**WHERE** **NOT** **EXISTS**(

**SELECT** 1 **FROM** property **AS** tabl\_1

**WHERE** **NOT** **EXISTS**(

**SELECT** 1 **FROM** property **AS** tabl\_2

**WHERE** tabl\_2.pro\_c\_id = D.c\_id **AND** tabl\_1.pro\_pif\_id = tabl\_2.pro\_pif\_id **AND** tabl\_2.pro\_type=3

) **AND** tabl\_1.pro\_c\_id = C.c\_id **AND** tabl\_1.pro\_type=3

# 而且不存在C中的基金元组D不具有的

) **AND** **NOT** **EXISTS**(

**SELECT** 1 **FROM** property **AS** tabl\_1

**WHERE** **NOT** **EXISTS**(

**SELECT** 1 **FROM** property **AS** tabl\_2

**WHERE** tabl\_2.pro\_c\_id = C.c\_id **AND** tabl\_1.pro\_pif\_id = tabl\_2.pro\_pif\_id **AND** tabl\_2.pro\_type=3

) **AND** tabl\_1.pro\_c\_id = D.c\_id **AND** tabl\_1.pro\_type=3 )

# 保证property表中有客户C的基金记录

**AND** **EXISTS**(**SELECT** 1 **FROM** property **WHERE** C.c\_id = pro\_c\_id **AND** pro\_type = 3);

## 2.4 触发器

本环节需要创建触发器对于事务操作支持安全性、完整性检查。当这个表上发生某个操作(insert,delete,update)时，触发器被触发执行，以实现业务完整性规则。当primary key, foreign key, check等约束都无法实现某个复杂的业务规则时，可以考虑用触发器来实现。本环节我们需要掌握构造触发器，new表和old表的使用等。

### 2.4.1 为投资表property实现业务约束规则

本题对于关卡1，首先定义触发器before\_property\_inserted，本触发器被触发执行的时间为在插入前执行，因此本触发器的定义为：

**CREATE** **TRIGGER** before\_property\_inserted **BEFORE** **INSERT** **ON** property

随后我们需要对插入元组实现判断，共设计四种错误类型的判断：（1）若新增元组类型不在三种投资类型内，则判断new.pro\_type属性，判定非法并设置错误信息，并将错误信息传送到msg中，随后使用语句signal sqlstate "45000" set message\_text = msg进行错误传输；（2）判断新增元组为非法理财产品，即元组类型为理财产品，但是finances\_product表中不包含新增产品编号，这时需要报错；（3）判断新增元组为非法保险产品，判断方法如上；（4）判断新增元组为非法基金产品，判断方法如上。

本题代码较长，此处不再赘述，详见提交材料中的附录代码部分。

## 2.5 数据库设计与实现

本环节要求对于数据库进行总体设计和实现。数据库的设计过程是在良好的需求分析的基础上，分别实现概念模型、逻辑模型的设计，最终落实到物理模型之中。在本节我们应良好地掌握概念模型设计、逻辑模型的使用和建模工具的使用。

### 2.5.1 从概念模型到MySQL实现

根据已建立的概念模型如图2.1所示，该E-R模型具有用户、旅客、机场、航空公司、民航飞机、航班常规调度表共六个实体，及其实体间的关系构成。



图2.1 订票系统E-R概念模型图

对于实体间的一对多关系，可以将联系的属性作为外码加入至多端表中。我们按照需求为数据库建立完整性约束，并通过测试。完整代码不再赘述，详见提交材料中附录代码。

### 2.5.2 制约因素分析与设计

在对实际问题的建模到数据库概念模型、逻辑模型的构建过程中，需要首先考虑需求的良好转化，建立确切的概念模型。本机票订票系统中，我们要充分考虑各个实体的属性集合以及实体间的联系。例如，机票信息中除了记录乘坐人信息，也需要记录购买人信息；也需要加入部分外码的参照关系，例如航班信息需要引入参照性约束，即航班的行程、飞机编号、所属公司等均需要作为外码加入表级约束。

同时，在数据库的正确性之上，我们还要充分考虑数据库的安全性问题。例如对于不同用户，他们访问数据库的权限往往是有差异的。这时候需要对用户授予不同的访问及增删改权限，以控制用户的行为。但这层权限往往可以采取一定的策略绕过，因此还需要有强制存取控制和审计机制，以对数据实现保护。对于数据本身，也可以采取加密的策略提高密文的安全度。

### 2.5.3 工程师责任及其分析

工程师的责任除了在于为任务提供解决方案，即对数据结构、需求分析、系统模型给出设计外，还需要对于复杂工程问题对于社会、健康、安全造成的影响做综合考量。工程师不能违背法律精神，制造社会对立或社会分裂，自觉将技术用于符合社会发展和社会进步的方向。不能对社会安全造成过大威胁，并对系统设计提供良好的安全防护和安全措施。

## 2.6 数据库应用开发(JAVA篇)

本任务环节需要使用JAVA语言调用MySQL中的各项功能实现协作开发。为了通过高级语言实现数据库的执行操作，JAVA语言支持执行SQL语句的API接口JDBC。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。我们下面利用该接口进行开发。

### 2.6.1 JDBC体系结构和简单的查询

本节对于关卡1，构建JDBC连接过程涉及以下四个步骤：

1. 导入JDBC包：将Java语言的import语句添加到Java代码中导入所需的类。
2. 注册JDBC驱动程序：此步骤将使JVM将所需的驱动程序实现加载到内存中，以便它可以满足您的JDBC请求。
3. 数据库URL配置：这是为了创建一个格式正确的地址，指向要连接到的数据库。
4. 创建连接对象：最后，调用DriverManager对象的getConnection（）方法来建立实际的数据库连接。

其具体代码详见提交文件中附录代码，此处不再赘述。

### 2.6.2 把稀疏表格转为键值对存储

本节对应关卡7，本题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。

我们应对本题的两个环节，即对稀疏表的查询和对于新表的插入分别设计方法，首先在主方法中依次查询稀疏表中的每个元组，并提取各科的成绩。随后调用子方法insertSC对新表进行插入，函数间用Connection类实现与SQL的通信。

本题的具体代码较长，详见提交材料中的附录代码，此处不再赘述。其中的关键代码为insertSC函数如下：

**public** **static** **int** insertSC(Connection connection, **int** sno, String col, **int** score){

**try** {

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

PreparedStatement ps = connection.prepareStatement(sql);

ps.setInt(1,sno);

ps.setString(2,col);

ps.setInt(3,score);

ps.executeUpdate();

} **catch** (SQLException e){

e.printStackTrace();

}

**return** 0;

}

## 2.7 数据库的索引B+树实现

数据库的B+树索引可用于数据库的查询和插入等操作的加速。其具体结构较复杂，包含了B+树结点的基本信息和功能，比如结点类型、包含元素存储最大值以及现存元素个数、父结点id等。B+树索引的实现需要实现一系列函数实现操作。

### 2.7.1 BPlusTreePage的设计

本关卡需要对B+树的一系列基本操作得到实现，包括返回最大孩子结点个数、修改最大孩子结点个数等。具体代码详见提交材料中的附录代码。

### 2.7.2 BPlusTreeInternalPage的设计

本关卡应实现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。其内部结点的功能即实现结点的初始化、查找、分裂，合并以及重分配等算法。

在本关中尤其需要注意内部结点的变动会影响其父节点的状态，因此需要迭代地对父节点的变化进行支持。

我们可以利用B+树索引中所有数据均为顺序排列的特性优化我们的实现函数代码，同时分裂出新结点需要对新的结点插入原结点的数据。本关卡实现代码较繁琐，详见提交材料中的代码部分。

# 3 课程总结

本次数据库系统原理实践课程，我完成了任务书中的2.1~2.10、2.13~2.15子任务的大部分子任务，在数据库理论的基础上加强了实践能力，并对数据库理论实现了更好的掌握。本次实验我们以MySQL作为平台锻炼我们的数据库开发能力，对数据库的各项操作、数据库设计和数据库底层开发与高级语言的API调用进行了实践，对数据库综合技能有了较好的掌握。

对于各个实训环节，我分别完成了数据库、表和完整性约束的所有子任务，并良好地学习了修改表属性和表级约束的所有关卡。对数据查询的大多数关卡都进行了实现，这一部分难度较大，我对不同的表进行较好的连接方案，实现对不同信息的查询操作的支持。随后我分别完成了数据插入、修改和删除的所有子任务，视图的所有子任务，存储过程与事务的多数任务，触发器、用户自定义函数、安全性控制的所有子任务。这一系列任务总体难度相较数据查询较为容易，实现起来需要学习新的语法之外，开发相对简单。在数据库的高级使用部分，我设计了机票系统数据库，并对JAVA的SQL接口的大部分子任务、B+树索引设计的大部分子任务都进行了支持，这一部分的难度也较大，实现较为不易，但对数据库的通信和底层设计产生了更好的理解。

通过上述实践，我更深刻地感受到了数据库在大型数据处理中的便利，也深感数据库引擎开发的不易。我建议本实践可以以一个具体项目为核心，给同学们一个项目的视角以对数据库的不同功能进行探索，并最终做出一个完整的数据库设计的系统操作，提高对数据库的系统认识。同时在这次数据库实践课程中，我已经受益匪浅，对数据库有了更好的理解与认识。最后，衷心感谢课程组对实验内容的精心设计和悉心指导！