华中科技大学

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

“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。

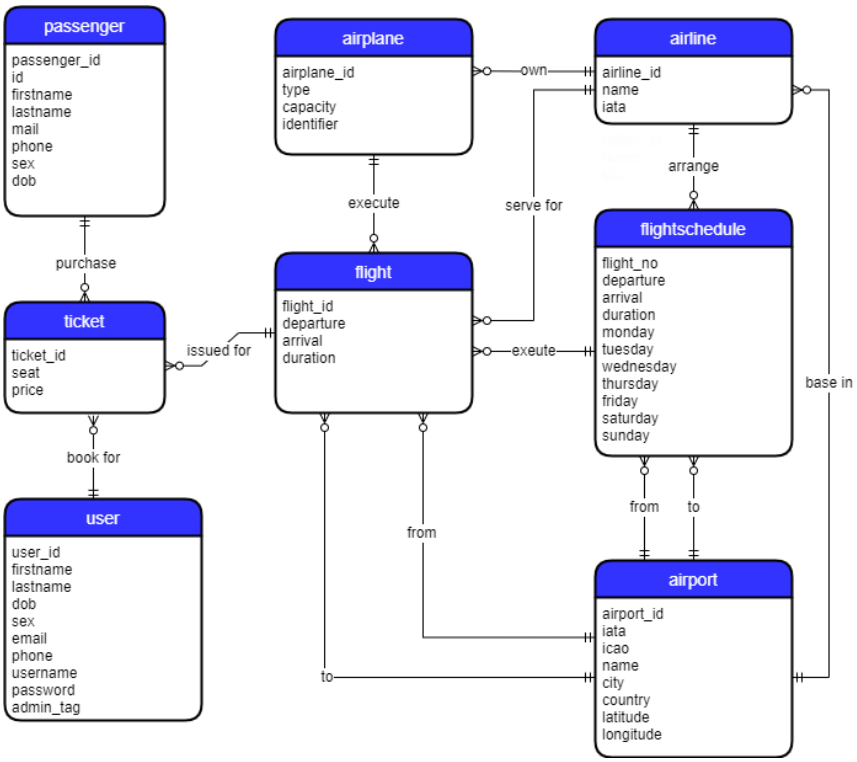

图 2.1 订票系统 E-R 概念模型图

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！