

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS2101 |
| 学 号： | U202115325 |
| 姓 名： | 宁毓伟 |
| 指导教师： | 瞿老师 |

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

2023 年 6月15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](#_Toc137760521)

[2 任务实施过程与分析 2](#_Toc137760522)

[**2.1** **数据库、表与完整性约束的定义(Create)** 2](#_Toc137760523)

[**2.2** **表结构与完整性约束的修改(ALTER)** 3](#_Toc137760524)

[**2.3** **数据查询(Select)之一** 5](#_Toc137760525)

[**2.4** **存储过程与事务** 13](#_Toc137760530)

[**2.5** **触发器** 14](#_Toc137760531)

[**2.6** **用户自定义函数** 15](#_Toc137760532)

[**2.7** **并发控制与事务的隔离级别** 16](#_Toc137760533)

[**2.8** **数据库应用开发(JAVA篇)** 16](#_Toc137760534)

[**2.9** **数据库设计与实现** 17](#_Toc137760535)

[3 课程总结 19](#_Toc137760536)

1. **课程任务概述**

本次实验课程是数据库系统概论课堂理论学习的延伸，目的是锻炼我们的动手实践能力，防止我们成为只会纸上谈兵的书呆子，并锻炼我们的工程素养，为日后成为一名优秀的工程师打下基础。

本次实验课程的所有任务都在头歌实践教学平台进行。头歌实践教学平台除了提供代码文件编辑界面之外，还提供了命令行环境让我们进行测试，这不仅能让我们更熟悉mysql客户端的使用，同时还能趁此机会学习Linux命令行交互界面的使用。

本次实验一共分成15个小任务，每个小任务中又包含多个关卡。任务涉及的知识范围十分广泛，包括mysql基础语法的使用，使用mysql客户端进行复杂的数据查询，对数据进行插入、修改以及删除，mysql的存储过程与事务，触发器的建立和使用，用户自定义函数及其使用，数据存储的安全性控制、并发控制与隔离，基于备份和日志的数据库恢复，数据库设计与实现以及数据库应用开发等内容。这些都将在头歌平台上完成，我们可以在大展身手的同时掌握数据库理论的实践方法。

1. **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2.1～2.14子任务，下面将重点针对其中的2.1、2.2、2.3、2.8、2.9、2.11、2.13以及2.14任务阐述其完成过程中的具体工作。

* 1. **数据库、表与完整性约束的定义(Create)**

本小节是数据库实验的第一个任务，主要内容是使用SQL语句创建数据库、创建表以及表的主码约束、创建外码约束、创建CHECK约束、创建DEFAULT约束以及创建UNIQUE约束。本小节任务的要求是在头歌实训平台中使用命令行或者代码文件完成相应的SQL语句编写。

* + 1. **创建数据库**
       1. 实验过程
  1. 在命令行中通过mysql客户端连接MySQL服务器。

在命令行中输入命令：“mysql -h127.0.0.1 -uroot -p123123”，并按下回车以连接MySQL服务器。

* 1. 输入命令创建用于2022年北京冬奥会信息系统的数据库:beijing2022。

在mysql客户端的交互界面中输入“CREATE DATABASE beijing2022；”并按下回车以创建数据库。

* + 1. **创建表及表的主码约束**
       1. 实验过程
  1. 创建数据库TestDb并使用它

1. create database TestDb;
2. use TestDb;
   1. 在TestDb中创建表t\_emp，并将id字段设为主码
3. CREATE TABLE t\_emp (
4. id INT PRIMARY KEY,
5. name VARCHAR(32),
6. deptId INT,
7. salary FLOAT
8. );
   1. **表结构与完整性约束的修改(ALTER)**

本小节的主要内容是使用SQL语句修改表名、添加或删除字段、修改字段以及添加/删除与修改约束。本小节任务的要求是在头歌实训平台的代码文件中完成相应的SQL语句编写。

* + 1. **修改表名**
       1. 实验过程：
  1. 将your\_table表的名字改为my\_table。

使用语句“alter table your\_table rename to my\_table;”即可将表名进行更改。

* + 1. **添加与删除字段**
       1. 实验过程：
  1. 删除表orderDetail中的列orderDate

使用语句“alter table orderDetail drop column orderDate;”即可删除orderDate字段·。

* 1. 添加列unitPrice

使用语句“alter table orderDetail add column unitPrice numeric(10, 2);”即可添加列unitPrice

* + 1. **修改字段**
       1. 实验过程：
  1. 将QQ号的数据类型改为char(12)

使用语句“alter table addressBook modify QQ char(12);”即可将QQ号的数据类型改为char(12)。

* 1. 将列名weixin改为wechat

使用语句“alter table addressBook rename column weixin to wechat;”即可将列名weixin改为wechat。

* + 1. **添加、删除与修改约束**
       1. 实验过程
  1. 为表Staff添加主码;

使用语句“alter table Staff add constraint primary key(staffNo);”即可添加主码约束。

* 1. 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;

使用语句“alter table Dept add constraint FK\_Dept\_mgrStaffNo foreign key(mgrStaffNo) references Staff(staffNo);”即可添加外码约束。

* 1. 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;

使用语句“alter table Staff add constraint FK\_Staff\_dept foreign key(dept) references Dept(deptNo);”即可添加外码约束。

* 1. 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;

使用语句“alter table Staff add constraint CK\_Staff\_gender check(gender='F'or gender='M');”即可添加check约束。

* 1. 为表Dept添加unique约束：deptName不允许重复。约束名为UN\_Dept\_deptName.

使用语句“alter table Dept add constraint UN\_Dept\_deptName UNIQUE(deptName);”即可添加unique约束。

* 1. **数据查询(Select)之一**
     1. **金融应用场景介绍,查询客户主要信息**

本关卡对client表进行查询，并使用order by子句对查询结果按照c\_id进行排序。代码如下所示：

1. select c\_name, c\_phone, c\_mail
2. from client order by
3. client.c\_id;
   * 1. **邮箱为null的用户**

在对client表进行查询的基础上，使用where子句筛选出邮箱为null的用户。代码如下所示：

1. select c\_id, c\_name, c\_id\_card, c\_phone
2. from client
3. where c\_mail is null;
   * 1. **既买了保险又买了基金的客户**
        1. 思路分析

本关卡要求查询的客户有两个条件，一是购买了保险，二是购买了基金。我们可以使用两个exists子句实现条件的选择。将当前的c\_id作为查询对象，如果在property表中存在pro\_c\_id与当前c\_id相同，且pro\_type=2（表示保险）的表项，则说明该用户购买了保险。对于该用户是否购买了基金，也同样可以用exists子句进行判断。

* + - 1. 实验过程：

根据实验要求从client表中选择c\_name, c\_mail, c\_phone。

1. select c\_name, c\_mail, c\_phone
2. from client

使用exists子句判断用户是否买了保险。

1. where exists (
2. select \* from property
3. where pro\_c\_id = c\_id and pro\_type = 2
4. ) # 买了保险

使用exists子句判断用户是否买了基金。

1. exists (
2. select \* from property
3. where pro\_c\_id = c\_id and pro\_type = 3
4. ) # 买了基金

根据要求按照c\_id升序排序。

1. order by c\_id;
   * 1. **办理了储蓄卡的客户信息**

将client表与bank\_card表进行连接后使用where子句选出银行卡类型为储蓄卡的信息，并使用order by子句对c\_id进行排序。完整代码如下所示：

1. select c\_name, c\_phone, b\_number
2. from client, bank\_card
3. where c\_id = b\_c\_id and b\_type = "储蓄卡"
4. order by c\_id;
   * 1. **每份金额在30000～50000之间的理财产品**

使用where子句选出p\_amount在30000～50000范围内的产品信息，并使用order by子句对p\_amount、p\_year进行降序排序。代码如下所示：

1. select p\_id, p\_amount, p\_year
2. from finances\_product
3. where p\_amount >= 30000 and p\_amount <= 50000
4. order by p\_amount, p\_year desc;
   * 1. **商品收益的众数**
        1. 思路分析

在资产表中商品收益为pro\_income字段。根据众数的定义，商品收益的众数为pro\_income出现次数最多的值，我们必须计算pro\_income每个值出现的次数。这个任务应该使用count(pro\_income)函数，并使用group by pro\_income子句来实现，表示根据pro\_income的不同值，计算count(pro\_income)的值。

接着判断某个pro\_income出现的次数是否为所有值中最大的那个。该部分使用having子句进行条件判断。只要当前的count >= 全部的count，即可认为当前的pro\_income是众数。

* + - 1. 实验过程

使用count函数与group by子句计算pro\_income中每个值出现的次数。

1. select pro\_income, count(pro\_income) as presence
2. from property
3. group by pro\_income

使用having子句判断当前pro\_income的值出现的次数是否大于或等于其他值出现的次数。

1. having presence >= all(
2. select count(pro\_income)
3. from property
4. sgroup by pro\_income
5. );
   * 1. **未购买任何理财产品的武汉居民**

武汉市的身份证号码为“4201”开头，本实验使用where子句和正则表达式选出武汉市居民的信息，接着使用not exists子句筛选出未购买理财产品的居民。代码如下所示：

1. select c\_name, c\_phone, c\_mail from client
2. where c\_id\_card rlike "^4201.\*" and
3. not exists(
4. select \* from property
5. where c\_id = pro\_c\_id and pro\_type = 1
6. );
   * 1. **持有两张信用卡的用户**
        1. 思路分析

题目的名字虽然叫做“持有两张信用卡的用户”，但实际上根据头歌实践教学平台的要求，在本关卡中我们需要选出持有两张（含）以上信用卡的用户。首先选出所有用户，接着计算当前用户的信用卡数量并与2做比较，若当前用户信用卡数量大于等于2，则该用户是结果之一。

* + - 1. 实验过程

从client表中根据要求选取用户的名字，身份证号以及电话号码。

1. select c\_name, c\_id\_card, c\_phone from client

在where子句中写出当前用户信用卡数量 >= 2的表达式。

1. where (
2. select count(\*) as cnt
3. from(
4. select \* from bank\_card
5. where c\_id = b\_c\_id and b\_type = "信用卡"
6. ) as card\_cnt
7. ) >= 2;
   * 1. **购买了货币型基金的客户信息**

首先从资产表property中选出购买了货币型基金的客户身份证号，在将client表中选出相应身份证号的客户信息。代码如下所示：

1. select c\_name, c\_phone, c\_mail
2. from client
3. where c\_id in (
4. select pro\_c\_id
5. from property
6. where pro\_type = 3 and pro\_pif\_id in (
7. select f\_id from fund
8. where f\_type = "货币型"
9. )
10. ) order by c\_id;
    * 1. **投资总收益前三名的客户**
         1. 思路分析

首先将client表与property表进行连接，连接的条件是c\_id = pro\_c\_id，并根据题目的要求选出c\_name, c\_id\_card以及sum(pro\_income)字段。选择的条件是pro\_status=’可用’，对pro\_income进行求和的时候要以c\_id进行分组求和，因为题目要求查询的是每个客户的总收益。接着将输出按照每个客户的总收益sum(pro\_income)降序排序，再限制输出三个元组即可。

* + - 1. 实验过程

将client表与property表进行连接并选出相应的字段，并设置条件为可用资产。

1. select c\_name, c\_id\_card, sum(pro\_income) as total\_income
2. from client join property on c\_id = pro\_c\_id
3. where pro\_status = '可用'

用group by子句设置求和分组条件。

1. group by c\_id

按照总收益降序排序，并限制输出三个表项。

1. order by total\_income desc
2. limit 3;
   * 1. **黄姓客户持卡数量**
        1. 思路分析

客户与银行卡的对应关系在表bank\_card中，其中b\_c\_id作为外码参照client表中的c\_id字段。在连接时选择使用client表左外连接bank\_card表，这是因为当某个客户没有任何一张银行卡时，我们也希望将其统计出来。上述左外连接对于没有任何一张银行卡的客户来说，其b\_c\_id字段为空，因此在count(b\_c\_id)的结果为0，这符合题目中“没有办卡的数量记为0”的要求。

接下来只需要根据要求选出黄姓客户以及按照银行卡数量降序排序即可。其中选出黄姓客户的办法是在where子句中使用正则表达式匹配c\_name，按照银行数量降序排序的方法是使用order by子句。

* + - 1. 实验过程

使用client表左外连接bank\_card表，并根据要求选出c\_id、c\_name字段以及将count(b\_c\_id)作为银行卡数量的统计值。

1. select c\_id, c\_name, count(b\_c\_id) as number\_of\_cards
2. from client left join bank\_card on c\_id = b\_c\_id

使用where子句以及正则表达式匹配选出黄姓客户。

1. where c\_name rlike "黄.\*"

为count(b\_c\_id)设置分组条件，并使用order by子句设置排序方式。

1. group by c\_id
2. order by number\_of\_cards desc, c\_id;
   * 1. **客户理财、保险与基金投资总额**
        1. 思路分析

由于理财产品、基金以及保险的信息分别存放在finances\_product、fund以及insurance这三张不同的表中，且客户购买产品的类型和数量存放在property表的pro\_type和pro\_quantuty字段中，因此我们首先需要使用property表分别连接finances\_product、fund以及insurance这三张不同的表，并使用pro\_quantity \* p\_amount、pro\_quantity \* i\_amount以及pro\_quantity \* f\_amount分别求出资产表property中每一份资产对应的金额。接着将连接后的三张表求并集即可得到一张含有所有资产对应的金额的表P，将client表与表P进行左外连接即可得到客户与投资金额的对应关系。

* + - 1. 实验过程

将property表分别与finances\_product、fund以及insurance这三张不同的表进行连接，并选出pro\_c\_id、pro\_type字段，并将pro\_quantity \* p\_amount作为amount字段表示资产的金额。下面给出property表与finances\_product表进行连接的代码，其他两张表的代码类似。

1. select pro\_c\_id, pro\_type, pro\_quantity \* p\_amount as amount
2. from property join finances\_product
3. on pro\_pif\_id = p\_id
4. where pro\_type = 1

使用union all求上述三张连接后的表的包含重复元素的并集作为表P。

1. (... union all ... union all ...) as P

将client表与表P进行左外连接，根据题目要求选出c\_name以及c\_id\_card字段，并对amount字段以c\_id作为分组条件进行求和作为total\_amount字段。

1. select c\_name,
2. c\_id\_card，
3. ifnull(sum(amount), 0) as total\_amount
4. from client left outer join (
5. ... union all ... union all ...
6. ) as P
7. on (c\_id = pro\_c\_id)

将最终结果按照total\_amount降序排序。

1. order by total\_amount desc;
   * 1. **客户总资产**

首先通过资产表property分别连接理财产品表、保险表与基金表，并计算出各个投资总额，在使用资产表的pro\_income字段计算出投资总收益，在使用银行卡表计算出储蓄卡余额总和与信用卡余额总额的相反数，最后将上述表进行联合（Union）并将对总额按照客户身份证分组求和，并将最终结果按照用户身份证排序。由于具体代码太长，在此不进行展示。具体代码见“src/实训3 数据查询(Select)之一/第13关 客户总资产.sql”。

* 1. **存储过程与事务**
     1. **使用流程控制语句的存储过程**
        1. 思路分析

本关卡要求使用存储过程向斐波那契表中添加斐波那契数列的前m项。当m=1时，直接插入第一项（0, 0）；当m>=2时，首先插入斐波那契数列的前两项，再使用循环计算并插入斐波那契数列剩下的项。

* + - 1. 实验过程

使用if语句判断m是否为1，并m=1时插入第一项。

当m不为1时，先插入斐波那契数列的前两项，并使用循环插入剩下的项。循环部分代码如下所示：

1. set @n = 2;
2. while @n < m do
3. select fibn into @a from fibonacci where n = @n - 1;
4. select fibn into @b from fibonacci where n = @n - 2;
5. insert into fibonacci values(@n, @a + @b);
6. set @n = @n + 1;
7. end while;
   1. **触发器**
      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**
         1. 思路分析

首先判断新加入的元组中pro\_type是否为1，2或3。如果不是，则输出“type x is illegal!”。接着分别使用三个if语句判断pro\_type的值，并在相应的投资产品表中查找是否存在与新插入的元组中pro\_pif\_id值相同pif\_id，若没有相同的值，则说明新插入的元组在投资产品表中没有引用的元组，此时根据要求输出相应的错误信息。

* + - 1. 实验过程

使用if语句和判断新插入的元组中pro\_type的值是否为1，2或3。

1. if new.pro\_type not in (1, 2, 3) then
2. set msg = concat('type ',new.pro\_type,
3. ' is illegal!');
4. SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = msg;

判断新加入的元组中pro\_type的值，并查找相应的表是否存在被引用的元组，如果没有则输出相应的错误信息。例如当新插入的元组代表金融产品时：

1. elseif new.pro\_type = 1
2. and not exists(
3. select \* from finances\_product
4. where new.pro\_pif\_id = p\_id
5. ) then
6. set msg = concat('finances product #',
7. new.pro\_pif\_id,' not found!');
8. SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = msg;
   1. **用户自定义函数**
      1. **创建函数并在语句中使用它**
         1. 思路分析

在get\_deposit函数中，从bank\_card表中选出client\_id客户编号的相关信息，并选取银行卡类型为储蓄卡的元组，再对储蓄卡的余额进行求和，并将求和的结果存入一个临时变量并返回。

在查询语句中，将client表与bank\_card表链接，并使用get\_deposit函数获取每个客户的存款总额，并筛选出存款总额在100万以上的客户信息，打印其银行卡号，姓名以及存款总额，并最终按照存款总额降序排序。

* + - 1. 实验过程

在get\_deposit函数中选出bank\_card表中客户编号为client\_id的储蓄卡信息，并对余额求和后返回。

1. select sum(b\_balance) into @ret
2. from bank\_card
3. where client\_id = b\_c\_id and b\_type = '储蓄卡'
4. group by b\_c\_id;
5. return @ret;

使用get\_deposit函数求出每个客户的存款总额，筛选出存款总额在100万以上的客户信息降序输出。

1. select
2. distinct c\_id\_card,
3. c\_name,
4. get\_deposit(c\_id) as total\_deposit
5. from client join bank\_card
6. where c\_id=b\_c\_id and get\_deposit(c\_id) >= 1000000
7. order by total\_deposit desc;
   1. **并发控制与事务的隔离级别**
      1. **可串行化**
         1. 思路分析

将两个事务设计成可串行化的方法有很多种，最简单的方法即是在其中一个事务延迟到另一个事务结束后再开始。在本关卡中我将事务1延迟到事务二结束后才开始。在事务二中，事务开始前需要先休眠一秒，因此我们只需要将在事务1的开始处设置休眠两秒即可。

* + - 1. 实验过程

在事务1的开始处设置休眠两秒：

1. start transaction;
2. set @n = sleep(2);
   1. **数据库应用开发(JAVA篇)**
      1. **JDBC体系结构和简单的查询**
         1. 思路分析

本关卡要求使用JDBC查询client表中邮箱不为空的客户信息，并列出客户的姓名，邮箱和电话。首先使用Class.forName方法注册JDBC驱动程序，接着使用DriverManager.getConnection方法连接数据库，接着使用connection.createStatement方法创建一个sql查询语句，再执行查询即可得到查询的结果。使用循环遍历输出查询结果即可。

* + - 1. 实验过程

注册驱动程序。

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

建立连接。

1. connection = DriverManager.getConnection(
2. "jdbc:mysql://localhost:3306/finance?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC","root","123123")*;*

创建查询语句并进行查询。

1. statement = connection.createStatement();
2. String sqlstr = "
3. select c\_name,c\_mail,c\_phone
4. from client
5. where c\_mail is not null";
6. ResultSet rs = statement.executeQuery(sqlstr);

循环遍历查询结果并输出。

1. while(rs.next()){
2. System.out.print(rs.getString("c\_name")+"\t");
3. System.out.print(rs.getString("c\_mail")+"\t\t");
4. System.out.println(rs.getString("c\_phone"));
5. }
   1. **数据库设计与实现**
      1. **从概念模型到MySQL实现**
         1. 思路分析

本关卡要求为机场订票系统设计一个数据库，数据库中应包含表2-1中所示的实体。

表2-1 机场订票系统数据库

|  |  |
| --- | --- |
| 实体 | 描述 |
| 用户(user) | 分为普通用户与管理用户 |
| 旅客(passenger) | 用于存储旅客信息 |
| 机场(airport) | 用于表示机场 |
| 航空公司(airline) | 用于描述航空公司 |
| 民航飞机(airplane) | 用于记录飞机的信息 |
| 航班常规调度表(flightschedule) | 用于表示航班的安排情况 |
| 航班表(flight) | 用于记录航班实际的起飞、到达时间 |
| 机票(ticket) | 用于描述的价格，座位等信息 |

根据关卡的要求，本关需要建立一个数据数据库并创建一系列描述上述实体的表，各个实体包含的属性以及实体之间的关系如图2-1所示。

图2-1 实体属性以及实体之间的关系



首先根据上图中描述的实体属性创建相应的表，并根据要求将各个属性设计为其对应的类型，并根据要求为属性添加相应的约束条件。接着我们需要根据图2-1中描述的各个实体之间的关系，为相应的表项建立外码约束即可。

* + - 1. 实验过程

根据要求为实体创建表，并添加相应的业务约束。

根据图2-1中各个实体之间的联系创建外码外码约束。

* + 1. **制约因素分析与设计**

实验过程中，在设计数据库表项时需要考虑实体的属性的数据类型以及取值范围。例如用户编号不能为空，并且为自动递增的整数；用户的生日取值范围不能超过当前日期等。

对于用户实体而言，用户编号唯一的表示一个用户，因此将其设计为主码，为了方便，还将其设置为自动递增的属性。考虑到每个用户登陆的时候需要使用用户名以及密码进行登陆，因此这两个属性不能为空，并且用户名不可重复。

对于旅客实体而言，机场可能会广播通知某位旅客某种信息，或者需要通过电话告知旅客某种信息，因此旅客的姓名以及电话不能为空。

考虑到航班可能存在早点或晚点的情况，因此需要在航班常规调度表之外再建立一个航班表用于记录航班真实的起飞时间与到达时间。

除上述具体因素之外，工程师还需要考虑安全性问题。例如用户在进行登陆时，输入的密码不能显示在交互界面中。在进行密码匹配时，密码的存储需要使用加密算法进行加密后存储。信息安全作为总体国家安全观的一部分，需要数据库工程师在设计数据库时时刻注意。

* + 1. **工程师责任及其分析**

对于大型数据库系统而言，工程师的责任表现在许多方面，例如工程师需要在设计数据库系统的过程中考虑数据库的易用性，并且数据还需要对现实世界的实体做出真实的反应。同时，工程师还需要保证数据库系统的安全性，并能够防止数据库系统遭到攻击。

在本实验中，机场订票系统中各个实体之间的联系并使用外码约束在数据库中将各个表项联系在一起，尽最大的努力反应了实际的机场订票系统中各个实体之间的关系。

在设计用户表项的过程中，我将用户区分为普通用户与管理用户，是为了防止没有管理权限的用户恶意修改数据库并对数据库系统进行攻击。

工程师的责任需要在设计数据库系统的过程中被充分地履行，只有这样才能保证设计出来的数据库系统具有较高的可靠性。

# 3 课程总结

我在本次数据库系统实验中从十五个实训中选择了部分最具有代表性的关卡进行实验，完成了头歌实践教学平台的要求。在实践的过程中，我将数据库系统原理理论课上学到的知识转化为实践能力，掌握了mysql的语法以及用法，同时还学习了使用Java语言进行数据库应用开发，并在这个过程中深入了解了数据库管理系统。

具体而言，在本次实验课程中，我完成了数据库、表与完整性约束的定义以及表结构与完整性约束的修改的实训，掌握了使用mysql创建数据库、创建表的基本操作；完成了数据查询之一的所有关卡以及数据查询之二的部分关卡，掌握了使用mysql进行复杂查询的方法；完成了数据插入、修改与删除实训的所有关卡，掌握了对数据库进行更新的方法；完成了视图实训的所有关卡，掌握了有关视图的操作；完成了存储过程与事务的大部分关卡，对存储过程与事务调度有了更深入的了解；完成了用户自定义函数的所有关卡，掌握了mysql函数定义与调用的方法；完成了安全性控制的所有关卡，对数据库系统的安全性有了进一步的理解；完成了并发控制与事务隔离级别实训的所有关卡，对于事务的并发控制更加熟悉；完成了备份+日志：介质故障与数据库恢复的所有关卡，掌握了使用备份和日志恢复数据库的方法；完成了数据库设计与实现的大部分关卡，掌握了数据库的设计与实现；完成了数据库应用开发的绝大部分关卡，掌握了使用Java语言进行数据库应用开发的方法。

经历了上述实训后，我使用数据库的能力得到了很大的提升。同时，我对于数据库以及数据库系统的理解更加深入，这将为我以后使用数据库进行开发打下坚实的基础。