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：	CS2104
学　　号：	U202115424
姓　　名：	张森磊
指导教师：	

分数	
教师签名	

2023 年 6 月 26 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，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

```
select c_id,c_name,c_id_card,c_phone from client where
```

2.3.16 持有完全相同基金组合的客户

本关涉及到的主要知识有派生表，等值连接等。对于客户基金组合派生表的生成，具体操作为对 property 表查询资金类型为基金（ pro _ type =3）的元组，按用户 id 分组，使用 group _ concat 函数对组内的基金号合并，生成一

个用户基金的集合，集合内按照基金号排序。对于该两个生成表定义为 pl 和 p2。

对上述产生的 pl 和 p2 进行基金组合集合的等值连接，产生基金组合相同的用户对，并且为了减少数据冗余，仅选取前用户 id 小于后用户 id 的元组，按前用户 id 升序排序。具体代码省略。

2.3.17 购买基金的高峰期

本关涉及的主要知识有等值连接，派生表，集合操作，枚举查找等。关键点在于连续三个交易日的选取。首先对 property 表和 fund 表进行等值连接，选取交易日期在 2022 年 2 月且在交易日的元组组成派生表。然后进行枚举查找 3 个连续的交易日内金额大于 100 万的元组，由于 2 月连续 3 个交易日的组合较少，故该方法具有可行性。对与查找到满足要求的元组，进行 UNION 操作，排除重复元组。具体代码省略。

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

本关涉及的主要知识有等值连接，派生表，分组统计等。首先生成至少有一张信用卡余额超过 5000 元的用户 id 的派生表，对 bank_card 表查询类型为信用卡且余额大于 5000 的元组，投影该用 id 。其次将 bank_card 表与该派生表进行用户 id 的等值连接，选取信用卡元组，按用户 id 分组统计信用卡金额的总和，并将其按用户 id 升序排序。具体代码省略。

2.3.19 以日历表格式显示每日基金购买总金额

此查询的关键点在于将列数据筛选变成行数据，即进行行列数据转换。查询过程如下，首先将 property 表和 fund 表进行基金 id 的等值连接，计算基金购买金额。再将该派生表中选取 2022 年 2 月交易日的元组按购买日期分组统计，由此产生周号、星期号和当日基金金额的派生表，该表为列数据型表。对此进行行列数据转换，将同周的数据组成个元组。具体做法则是将该派生表按周号分组统计，在周号相同的一组中，统计星期号为周一的总金额，周二的总金额，…，以此产生每周的元组数据。具体代码省略。

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 $a_n = a_{n-1} + a_{n-2}$ 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=U
TF8&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

该任务关卡跳过。

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

该任务关卡跳过。

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，除了实验四数据查询之二的后两个关卡、实验十三数据库设计与实现的第三个关卡、实验十五数据库的索引 B+树实现的后五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。