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：	CS2011
学　　号：	U202014774
姓　　名：	王逸
指导教师：	潘鹏

分数	
教师签名	

2023 年 01 月 02 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，实践课程 url 见相关课堂教师发布链接及其邀请码。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

2.1 数据库、表与完整性约束的定义(Create)

本子任务中的 6 个关卡主要围绕数据库和表的创建进行，依次完成了数据库的创建、表的创建以及表中主码、外码、各种约束的建立。

2.1.1 创建数据库

该关卡任务已完成，实施情况本报告略过。

2.1.2 创建表及表的主码约束

本关任务：在指定的数据库中创建一个表，并为表指定主码。

过程：先创建数据库 TestDb，在 TestDb 下用 create table 创建表 t_emp，并对 id 增加 primary key 约束。

```
create database TestDb;
use TestDb;
create table if not exists t_emp(id int primary key,name varchar(32),deptId int,salary float);
```

2.1.3 创建外码约束(foreign key)

本关任务：创建外码约束（参照完整性约束）。

过程：创建 MyDb 数据库，并在其中创建 dept 表和 staff 表在表 staff 创建外键。

```
create table if not exists dept(deptNo int primary key, deptName varchar(32));
create table if not exists 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));
```

2.1.4 CHECK 约束

本关任务：为表创建 CHECK 约束。

过程：创建表的时候根据要求在相应的列名后面添加 check 约束。

```
create table if not exists products(
        pid char(10) primary key,
        name varchar(32),
        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

该关卡任务已完成，实施情况本报告略过。

2.1.6 UNIQUE 约束

该关卡任务已完成，实施情况本报告略过。

2.2 表结构与完整性约束的修改(ALTER)

本子任务中的 4 个关卡主要围绕表的基本操作进行, 使用 alter 语句对表中的内容及约束进行修改，包括更换/修改表名、列名、列类型、列约束及表约束。

2.2.1 修改表名

该关卡任务已完成，实施情况本报告略过。

2.2.2 添加与删除字段

本关任务：为表添加和删除字段。一是删除表 orderDetail 中的列 orderDate；二是添加列 unitPrice。

过程: 根据具体任务分别使用“DROP [COLUMN] 列名”和“ADD [COLUMN] 列名 数据类型 [列约束]”来增删列项。

```
alter table orderDetail drop orderDate;
alter table orderDetail; add unitPrice numeric(10,2);
```

2.2.3 修改字段

本关任务：编程任务是对表 addressBook 作以下修改：将 QQ 号的数据类型改为 char(12);将列名 weixin 改为 wechat。

过程：根据具体任务分别使用“MODIFY [COLUMN] 列名 数据类型 [列约束]”和“RENAME COLUMN 列名 TO 新列名”来实现。

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

2.2.4 添加、删除与修改约束

本关任务：添加、删除及修改约束。

过程：按提示和要求选择合适的修改命令即可。

```
#(1) 为表 Staff 添加主码
alter table Staff add primary key(staffNo);
#(2) 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:
alter table Dept add constraint FK_Dept_mgrStaffNo foreign key(mgrStaffNo) references Staff(staffNo);
#(3) 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:
alter table Staff add constraint FK_Staff_dept foreign key(dept) references Dept(deptNo);
#(4) 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:
alter table Staff add constraint CK_Staff_gender check(gender in ('F','M'));
#(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

本子任务中的 19 个关卡主要围绕 select 语句进行，了解掌握各种查询方法。

2.3.1 查询客户主要信息

该关卡任务已完成，实施情况本报告略过。

2.3.2 邮箱为 null 的客户

该关卡任务已完成，实施情况本报告略过。

2.3.3 既买了保险又买了基金的客户

本关任务：查询既买了保险又买了基金的客户的名称和邮箱。

过程：使用嵌套查询即可，最里层查询购买了基金的客户，接着查询在这些客户里购买了保险的客户。

```
select c_name, c_mail, c_phone from client
```

```
where c_id in (
        select distinct pro_c_id from property
        where pro_type = 2 and pro_c_id in (select distinct pro_c_id from property where pro_type=3)
)   order by c_id;
```

2.3.4 办理了储蓄卡的客户信息

该关卡任务已完成，实施情况本报告略过。

2.3.5 每份金额在 30000～50000 之间的理财产品

该关卡任务已完成，实施情况本报告略过。

2.3.6 商品收益的众数

本关任务：查询资产表中所有资产记录里商品收益的众数和它出现的次数。

过程：使用 property 表，基于 pro_income 分组，利用 count 统计出现的次数，若某个分组的计数大于或等于所有分组（all 关键词），则为我们需要求的众数。

```
select pro_income, count(*) as presence
from property
group by pro_income
having presence >= 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

本关任务：查询未购买任何理财产品的武汉居民的信息。

过程：按照提示，武汉市居民身份证前 4 位为“4201”，首先用 like 筛选出武汉居民，再用 exist 语句查询是否购买理财产品。

```
select c_name, c_phone, c_mail from client
where c_id_card like '4201%' AND not exists(select * from property where pro_c_id = client.c_id AND pro_type = 1)
order by c_id;
```

2.3.8 持有两张信用卡的用户

该关卡任务已完成，实施情况本报告略过。

2.3.9 购买了货币型基金的客户信息

该关卡任务已完成，实施情况本报告略过。

2.3.10 投资总收益前三名的客户

本关任务：查询投资总收益前三名的客户。

过程：以 property 中的 pro_c_id 分组，将 client 和 property 两张表通过 c_id 和 pro_c_id 连接起来，查询 status 为“可用”的客户，使用 sum 计算总收益按从高到低排列并输出前三名。

```
select c.c_name, c.c_id_card, sum(p.pro_income) as total_income
from client c, property p
where c.c_id = p.pro_c_id AND p.pro_status = '可用' # 这里直接使用 and 作为判断条件
group by p.pro_c_id
order by total_income desc limit 0,3; # 0 为起始行，3 为偏移量
```

2.3.11 黄姓客户持卡数量

本关任务：查询黄姓用户的编号、名称、办理的银行卡的数量。

过程：利用 left outer join 将 client 和 bank_card 在 c_id=b.b_c_id 基础上作做链接，按 c_id 分组，使用 like 查询黄姓客户，最后利用 sum 统计 b_number 的数目，为了输出 0 使用 ifnull 函数作填充，最后按降序输出。下面仅给出 select 语句和连接部分语句，其余语句参考上述任务实现即可。测试结果见图 2.1 11 关测试结果。

```
select c.c_id, c.c_name, ifnull(count(b.b_number), 0) as number_of_cards
from client c left outer join bank_card b on (c.c_id = b.b_c_id)
```

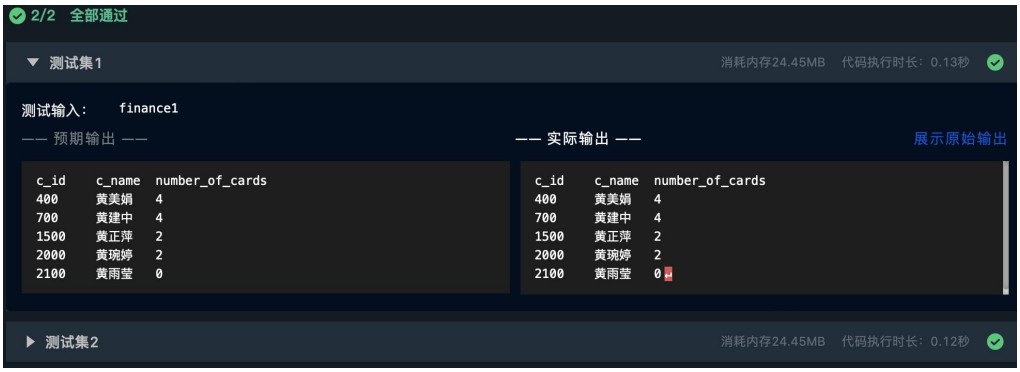

图 2.1 11 关测试结果

2.3.12 客户理财、保险与基金投资总额

该关卡任务已完成，实施情况本报告略过。

2.3.13 客户总资产

上一关实现思路与本关相同，故选择第 13 关介绍。

本关任务：查询客户在本行的总资产。

过程：由于资产有三种类别分别对应三张表，所以需要将每个客户每种类型的资产总和利用一条 select 语句获取，并利用 union all 取可重并集得到一张新的 ps 表，列分别为 id，quantity，income 和 amount；从 bank_card 中选取 b_c_id 和卡的总余额（其中信用卡统一乘以-1）形成子表，再将该子表与 client 在 cid 相等的基础上进行右连接（client right outer join bank_card*）；最后将 ps 表与连接后的 client 表进行右连接，按照 cid 分组以及给出的公式计算总资产。

```
select c.c_id, c.c_name, ifnull(sum(ps.pro_quantity*ps.amount) + sum(ps.pro_income) + cb.tt,0) as total_property
from(
    (select p.pro_c_id, p.pro_quantity, p.pro_income, amount from property p inner join ( select f_id, f_amount from fund) as f(id,amount) on (p.pro_pif_id=f.id) where p.pro_type=3)
    union all(··· pro_type = 1)  union all(··· pro_type = 2)) as ps right outer join (
    client c left outer join (select b_c_id, sum(case b_type when '储蓄卡' then b_balance else -1*b_balance end) as card_tt from bank_card group by b_c_id) as cb(id,tt) on (c.c_id=cb.id)) on (ps.pro_c_id=c.c_id)
group by c.c_id
order by c.c_id;
```

2.3.14 第 N 高问题

本关任务：查询每份保险金额第 4 高保险产品的编号和保险金额。

过程：使用 insurance 表，首先找到第四高的金额，利用 distinct 关键词选出不同金额并按降序排列，利用 limit 1 offset 3 找到第四个金额；再套一层外部查询，判断保险金额是否等于这个第四高金额选择输出信息即可。

```
select i_id, i_amount
```

```
from insurance
where i_amount = (select ifnull((select distinct i_amount from insurance order by i_amount desc limit 1 offset 3),null));
```

2.3.15 基金收益两种方式排名

本关任务：对客户基金投资收益实现连续排名和不连续排名两种方式的排次。

过程：以基金投资收益总额为关键字调用排名函数即可。其中 rank()是不连续的排名，dense_rank()是排名连续的排名函数。代码只给出两个任务的 select 语句部分，需要注意的是 over()里的关键字不能直接使用 total_revenue。

```
select p.pro_c_id, sum(p.pro_income) as total_revenue, rank() over (order by sum(p.pro_income) desc) as 'rank'
select p.pro_c_id, sum(p.pro_income) as total_revenue, dense_rank() over (order by sum(p.pro_income) desc) as 'rank'
```

2.3.16 持有完全相同基金组合的客户

本关任务：查询持有完全相同基金组合的客户。

过程：利用 with as 创建一个临时公用表，按照客户 id 分组，利用 group_concat()将去重排序后的客户购买的基金 id 拼接成一个字符串 f_id，分隔符默认为‘，’，无需担心(1, 12)和(11, 2)连接造成错误计算的情况。若表中两个客户的 f_id 相等，则说明这两个客户的基金组合完全相同。

```
with pro(c_id, f_id) as (
    select pro_c_id c_id, group_concat(distinct pro_pif_id order by pro_pif_id ) f_id
    from property
    where pro_type=3
    group by pro_c_id
)
select p1.c_id c_id1, p2.c_id c_id2
from pro p1, pro p2
where p1.c_id < p2.c_id and p1.f_id = p2.f_id;
```

2.3.17 购买基金的高峰期

本关任务：查询 2022 年 2 月购买基金的高峰期，如果连续三个交易日投资者购买基金的总金额超过 100 万，则称这连续的几日为投资者购买基金的高峰期。

过程：由于每周有两天不是交易日，首先通过 datediff(pro_purchase_time, "2021 - 12 - 31")获取交易日是2022年的第几天a, 再调用 week()获取该天在2022年的第几周 b，由 a-2*b 得到该天是 2022 年的第几个交易日，记为 workday。然后利用交易时间和交易类型筛选得到一张新表。

从得到的新表中选择交易总金额超过 100 万的行，通过 row_number 得到筛选后的行号，记为 rownum，如果两行的 workday-rownum 结果相同，说明这两行在一个连续段中，再调用 count 函数获取连续的天数。结果见图 2.2 查询通关结果所示。

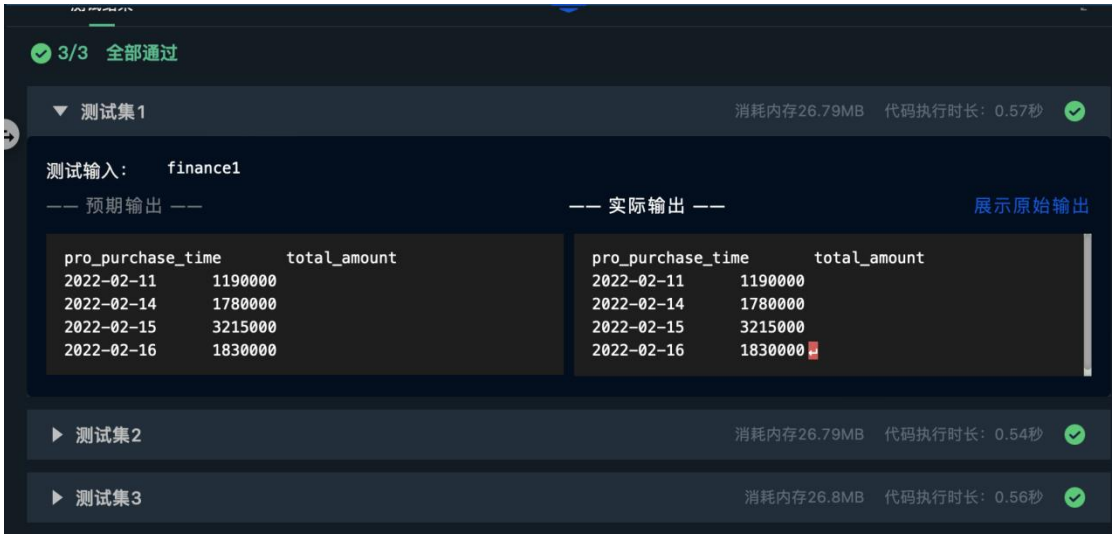

图 2.2 查询通关结果

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

该关卡任务已完成，实施情况本报告略过。

2.3.19 以日历表格式显示每日基金购买总金额

本关任务：以日历表格式显示 2022 年 2 月每周每日基金购买总金额。

过程：对 property 和 fund 表在 pro_pif_id=f_id 的基础上做自然连接，按交易时间分组计算基金购买总金额，用 weekday()函数得到该天是当年的周几，使用 if 判断该将金额放在哪一列计算。

```
select wk week_of_trading, sum(if(dayId = 0, amount, null)) Monday, sum(if(dayId = 1, amount, null)) Tuesday, sum(if(dayId = 2, amount, null)) Wednesday, sum(if(dayId = 3, amount, null)) Thursday, sum(if(dayId = 4, amount, null)) Friday
from (
    select week(pro_purchase_time) - 5 wk, weekday(pro_purchase_time) dayId, sum(pro_quantity * f_amount) amount
    from property join fund on pro_pif_id = f_id
    where pro_purchase_time like "2022-02-%" and pro_type = 3
    group by pro_purchase_time) t    group by wk;
```

2.4 数据查询(Select)之二

本小节子任务中的 6 个关卡仍然以第 2.3 子任务的数据库内容为背景，同样主要围绕 select 语句进行，了解掌握各种查询方法。

2.4.1 查询销售总额前三的理财产品

该任务关卡跳过。

2.4.2 投资积极且偏好理财类产品的客户

本关任务：找到投资积极且偏好理财类产品的客户

过程：用两个嵌套的子查询分别查找用户购买的理财产品数目和基金数目，用客户编号连接两个表，找出理财产品数目大于基金数目的客户 id 并按升序排列即可。

```
select p1.pro_c_id
from(select count(distinct pro_pif_id) as fp_num, pro_c_id from property, finances_product where pro_pif_id=p_id and pro_type=1group by pro_c_id having count(distinct pro_pif_id)>3)p1,
(select count(distinct pro_pif_id) as f_num,pro_c_id from property,fund where pro_pif_id=f_id and pro_type=3 group by pro_c_id)p2
where  p1.pro_c_id=p2.pro_c_id and fp_num>f_num
order by p1.pro_c_id
```

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)

本小节子任务中的 6 个关卡主要围绕 insert，update，delete 语句的应用进行。

2.5.1 插入多条完整的客户信息

该关卡任务已完成，实施情况本报告略过。

2.5.2 插入不完整的客户信息

本关任务：向客户表 client 插入一条数据不全的记录。

过程：可以在 values 前指定列，也可以在插入信息时将空值标明 null。

```
insert into client (c_id, c_name, c_phone, c_id_card, c_password)
values (33, "蔡依婷", "18820762130", "350972199204227621", "MKwEuc1sc6");
```

2.5.3 批量插入数据

本关任务：向客户表 client 批量插入数据。用一条 SQL 语句将 new_client 表的全部客户信息插入到客户表(client)。

过程：将 insert 中的 values 更换为 select 查询语句。

```
insert into client select *    from new_client;
```

2.5.4 删除没有银行卡的客户信息

本关任务：删除在本行没有银行卡的客户信息。

使用 delete 语句删除信息，查询筛选时使用 not exists 语句，注意的是 delete 语句中的 from 不能删掉。

```
delete from client
```

```
where not exists (
    select b_c_id from bank_card where client.c_id = bank_card.b_c_id
);
```

2.5.5 冻结客户资产

该关卡任务已完成，实施情况本报告略过。

2.5.6 连接更新

本关任务：根据客户表的内容修改资产表的内容。

过程：根据要求使用 update 语句。

```
update property,client
set property.pro_id_card = client.c_id_card
where property.pro_c_id = client.c_id
```

2.6 视图

本小节子任务中的 2 个关卡主要围绕视图的创建与使用进行。

2.6.1 创建所有保险资产的详细记录视图

本关任务：基于视图 v_insurance_detail 查询每位客户保险资产的总额和保险总收益。

过程：使用 create view name as 创建视图，后面使用 select 语句即可。

```
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 client, property, insurance
where c_id = pro_c_id and pro_type = 2 and pro_pif_id = i_id;
```

2.6.2 基于视图的查询

该关卡任务已完成，实施情况本报告略过。

2.7 存储过程与事务

本小节子任务中的 3 个关卡主要围绕存储过程进行。

2.7.1 使用流程控制语句的存储过程

该任务关卡跳过。

2.7.2 使用游标的存储过程

该任务关卡跳过。

2.7.3 使用事务的存储过程

该任务关卡跳过。

2.8 触发器

本小节子任务中的 1 个关卡主要围绕触发器的使用进行。

2.8.1 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

本关任务：为表 property(资产表)编写一个触发器，以实现要求的完整性业务规则。

过程：声明一个 before insert on property 的行触发器，按照要求编写函数体部分，若发现不满足条件的情况调用 concat 函数编辑报错信息 msg，之后通过 raise exception 抛出错误信息。函数体部分见图 2.3 触发器函数体部分。

```
if tp =1 then
    if id not in (select p_id from finances_product) then
        set msg = concat("finances product #", id, " not found!");
    end if;
elseif tp =2 then
    if id not in (select i_id from insurance) then
        set msg = concat("insurance #", id, " not found!");
    end if;
elseif tp =3 then
    if id not in (select f_id from fund) then
        set msg = concat("fund #", id, " not found!");
    end if;
else
    set msg = concat("type ", tp, " is illegal!");
end if;
if msg is not null then
    signal sqlstate "45000" set message_text = msg;
end if;
```

图 2.3 触发器函数体部分

2.9 用户自定义函数

本小节子任务中的 1 个关卡主要围绕用户自定义函数的定义和使用进行。

2.9.1 创建函数并在语句中使用它

本关任务：编写一个依据客户编号计算其在本金融机构的存储总额的函数，并在 SELECT 语句使用这个函数。

过程：按照要求编写函数部分。

```
delimiter $$
create function get_deposit(client_id int)
returns numeric(10,2)
begin
    return (select sum(b_balance) from bank_card where b_type="储蓄卡" and b_c_id = client_id group by b_c_id);
end$$
delimiter ;
# 调用部分：
select c_id_card,c_name,get_deposit(c_id) as total_deposit
```

2.10 安全性控制

本小节子任务中的 2 个关卡主要围绕数据库的安全性控制进行。

2.10.1 用户和权限

该关卡任务已完成，实施情况本报告略过。

2.10.2 用户、角色与权限

本关任务：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。

过程：按照要求使用对应命令即可。

```
# (1) 创建角色 client_manager 和 fund_manager;
create user client_manager;
create user fund_manager;
# (2) 授予 client_manager 对 client 表拥有 select,insert,update 的权限;
```

```
grant select, insert, update on table 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_c_id, b_number, b_type) on table bank_card to client_manager;
# (4) 授予 fund_manager 对 fund 表的 select,insert,update 权限；
grant select, insert, update on table 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

本小节子任务中的6个关卡主要涉及数据库中并发控制与事务的歌丽姬别相关内容。

2.11.1 并发控制与事务的隔离级别

该关卡任务已完成，实施情况本报告略过。

2.11.2 读脏

本关任务：选择合适的事务隔离级别，构造两个事务并发执行时，发生“读脏”现象。

过程：根据任务要求将隔离级别设置成 read uncommitted，然后根据注释设置事务休眠时间。

```
-- 事务 1:
## 请设置适当的事务隔离级别
set session transaction isolation level read uncommitted;
-- 时刻 2 - 事务 1 读航班余票,发生在事务 2 修改之后
## 添加等待代码，确保读脏
set @n = sleep(1);
```

```
-- 事务 2
-- 请设置适当的事务隔离级别
set session transaction isolation level read uncommitted;
```

```
-- 时刻 3 - 事务 2 取消本次修改
-- 请添加代码，使事务 1 在事务 2 撤销前读脏;
set @n = sleep(2);
```

2.11.3 不可重复读

该关卡任务已完成，实施情况本报告略过。

2.11.4 幻读

该关卡任务已完成，实施情况本报告略过。

2.11.5 主动加锁保证可重复读

本关任务：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。

过程：本实验中事务 2 尝试在事务 1 两次读的动作之间取出一次票，但由于事务 1 的第一次查询航班加了写锁, 所以事务 2 的取票更新动作无法在锁释放前执行，从而实现可重复读。

```
-- 事务 1:
set session transaction isolation level read uncommitted;
# 第 1 次查询航班'MU2455'的余票
select tickets from ticket where flight_no = 'MU2455' for update;
set @n = sleep(5);
# 第 2 次查询航班'MU2455'的余票
select tickets from ticket where flight_no = 'MU2455' for update;
commit;
-- 第 3 次查询所有航班的余票，发生在事务 2 提交后
set @n = sleep(1);
```

```
-- 事务 2:
set session transaction isolation level read uncommitted;
set @n = sleep(1);
# 在事务 1 的第 1，2 次查询之间，试图出票 1 张(航班 MU2455)：
update ticket …
```

```
commit;
```

2.11.6 可串行化

本关任务：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。

过程：比较简单的方案是直接让事务 1 休眠较长时间，直到事务 2 执行完后再执行即可。

```
-- 事务 1:
use testdb1; start transaction;
set @n = sleep(5);
…
commit;
```

2.12 备份+日志：介质故障与数据库恢复

本小节子任务中的 2 个关卡主要围绕数据库的备份与恢复进行。

2.12.1 备份与恢复

该关卡任务已完成，实施情况本报告略过。

2.12.2 备份+日志：介质故障的发生与数据库的恢复

本关任务：模拟介质故障的发生，以及如何利用备份和备份之后的日志恢复数据库。

过程：对数据库 train 作一次海量备份，同时新开日志：加入--flush-logs 即可；

```
mysqldump -h127.0.0.1 -uroot --flush-logs --databases train > train_bak.sql;
```

利用逻辑备份和日志恢复数据库的命令：按照提示即可，注意加入 --no-defaults 参数，不然会有编码格式冲突的问题。

```
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

本小节子任务中的 3 个关卡主要涉及数据库的涉及与设计相关内容。

2.13.1 从概念模型到 MySQL 实现

该关卡任务已完成，实施情况本报告略过。

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

该任务关卡跳过。

2.13.4 制约因素分析与设计

从实际问题的建模到数据库的概念模型和逻辑模型的构建过程中，需要考虑若干因素。以本次的订票系统为例，我们需要考虑到旅客的实际情况，例如由于旅客可以多次乘坐飞机，那么一张机票是由某个人为某个旅客购买的特定航班的机票，所以机票信息需要记录乘坐人与购买人信息（虽然两者有时是同一人）。

2.13.5 工程师责任及其分析

社会方面，工程师应该能够基于工程相关背景知识进行合理分析，评价专业工程实践和复杂工程问题解决方案对社会、健康、安全、法律以及文化的影响，并理解应承担的责任。安全方面，工程师应该尽可能考虑系统中存在的安全漏洞，安全性是所有系统用户关心的重要命题；科学发展方面，工程师应该能够基于科学原理并采用科学方法对复杂工程问题进行研究，包括设计实验、分析与解释数据、并通过信息综合得出合理有效的结论。

2.14 数据库应用开发(JAVA 篇)

本小节子任务中的 7 个关卡从 JDBC 体系结构出发，涉及使用 JAVA 开发数据库应用的基本知识。

2.14.1 JDBC 体系结构和简单的查询

该关卡任务已完成，实施情况本报告略过。

2.14.2 用户登录

本关任务：编写客户登录程序，提示用户输入邮箱和密码，并判断正确性，给出适当的提示信息。

过程：用 PreparedStatement 类执行 SQL 语句，并且把 SQL 语句中变化的部分当作参数用来防御 SQL 的注入攻击。

```
statement = connection.createStatement();
String sql = "select * from client where c_mail = ? and c_password = ?";
PreparedStatement ps = connection.prepareStatement(sql);
ps.setString(1, loginName);
ps.setString(2, loginPass);
resultSet = ps.executeQuery();
if (resultSet.next())
        System.out.println("登录成功。");
else
        System.out.println("用户名或密码错误!  ");
```

2.14.3 添加新客户

本关任务：编程完成向 client(客户表)插入记录的方法。

过程：使用 PreparedStatement 类执行即可。

```
String sql = "insert into client values (?, ?, ?, ?, ?, ?)";
try {
        PreparedStatement ps = connection.prepareStatement(sql);
        ps.setInt(1, c_id);
        ps.setString(2, c_name);
        ps.setString(3, c_mail);
        ps.setString(4, c_id_card);
        ps.setString(5, c_phone);
        ps.setString(6, c_password);
        return ps.executeUpdate();
} catch (SQLException e){
        e.printStackTrace();
}
```

2.14.4 银行卡销户

该关卡任务已完成，实施情况本报告略过。

2.14.5 客户修改密码

本关任务：编写修改客户登录密码的的方法。

过程：首先判断该用户是否存在，若存在再判断输入的旧密码是否正确，最后用 update 修改密码。

```
String sql = "select * from client where c_mail = ?";
try {
    PreparedStatement ps = connection.prepareStatement(sql);
    ps.setString(1, mail);
    ResultSet resultSet = ps.executeQuery();
    if(resultSet.next()){
        if(password.equals(resultSet.getString("c_password"))){
            sql = "update client set c_password = ? where c_mail = ? and c_password = ?";
            ps = connection.prepareStatement(sql);
            ps.setString(1, newPass);
            ps.setString(2, mail);
            ps.setString(3, password);
            ps.executeUpdate();
            return 1;
        }
        else return 3;
    }
    else return 2;
} catch (SQLException e){
    e.printStackTrace();
}
return -1;
```

2.14.6 事务与转账操作

该关卡任务已完成，实施情况本报告略过。

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

本小节子任务中的 7 个关卡主要围绕存储过程进行。

2.15.1 BPlusTreePage 的设计

本关任务：作为 B+树索引结点类型的数据结构设计的第一部分：实现 BPlusTreePage 类，该类是 B+树叶结点类型和内部结点类型的父类，提供 B+树结点的基本功能。BPlusTreePage 作为 BPlusTreeInternalPage 与 BPlusTreeInternalPage 父类，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，当前结点 id。

过程：按照要求的完成的函数及功能完成函数的编写即可，没有什么太困难的地方。

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

本次实验从最基础的数据库、表的创建开始，依次完成了表的约束、数据查询、数据修改相关命令、视图的创建与使用、并发事务、触发器、自定义函数、安全性控制、备份与日志、数据库应用、数据库的设计以及 B+树等 14 个任务，共计 58 个关卡。

我在这次课程实验里耗时最长的部分主要是数据查询与数据库 JAVA 应用两大部分。第一次接触 mysql 相关指令，不像其他使用较多的语言，每次进入新的关卡时总会在某些小地方卡住，例如排名、第 N 高问题等，需要自己所搜资料后才能找到解决办法，另外在网上搜来的 limit 相关使用，有的形式在本次实验中或是因为版本原因会有出错提示，需要更换 limit offset 搭配使用；而越到后面对于查询的先后以及表连接相关考察的越深，需要根据表项选择合适的列组成新表，在这之中也对 union、outer join 等连接方式以及多表连接有了进一步的认识。其余任务的关卡虽然看起来很多，但提示已经把最需要的部分给出，完全只需要按照左侧栏所给的资料就可以做出，并且往往只需要一两行语句就足够。同时在这次实验中也有一个数据库的设计任务，是对前面所学的知识与命令的综合，需要我们去考虑在实际情况下一个数据库系统设计在数据之间可能会有哪些联系以及限制。

最后感谢各位老师在本次课程实验中对我的帮助与指导!