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：	CS2208
学　　号：	U202215642
姓　　名：	田清林
指导教师：	谢美意

分数	
教师签名	

2024 年 7 月 3 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”是为配合“数据库系统原理”课程独立开设的实践课程，旨在通过 MySQL 编程语言进行数据库的各项操作和开发任务。课程依托头歌实践教学平台，实验环境在 Linux 操作系统下，主要编程语言为 MySQL 8.0.28，并在数据库应用开发环节使用 JAVA 1.8。

课程任务分为五个主要部分：

1. 数据对象的管理与编程，包括数据库、表、索引、视图、约束、存储过程、函数、触发器、游标等。

2. 数据操作任务，包括数据查询，数据插入、删除与修改等。

3. 数据库系统内核实验。包括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。

4. 数据库的设计与实现；

5. 数据库应用系统的开发 (JAVA 篇)，主要包括 JAVA 编程与数据库的交互。

本课程依托头歌实践教学平台，实验环境在 Linux 操作系统下，编程语言主要为 MySQL 8.0.28。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课程平台中的第 1~14 实训的所有任务。下面将重点针对其中的数据查询、存储过程与事务、触发器、用户自定义函数、并发控制与事务的隔离级别、数据库设计与实现内容，阐述完成过程中的具体工作。

2.1　数据查询(Select)之一

本节的主要任务是对若干个表的数据进行查询操作，主要考察了 select 语句的各种用法，在具体的情境之中设计了单表和多表的直接、嵌套等查询，还使用了必要的函数。

本任务所有关卡均已完成，选取其中部分较复杂关卡进行重点阐述。

2.1.1　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。

实现方法： 该关卡可分两步进行，第一步算出每个客户的投资总收益，第二步，选出投资总收益前三名的客户。

首先对于总收益，即将 property 表与 client 表按照 c_id = pro_c_id 连接后，按照 c_id 分组并使用聚集函数 SUM()来对 pro_income 求和，总收益命名为 total_income。

第二步，由于题中给出不考虑并列排名情形，因此对于客户 A 的排名就是投资总收益大于客户 A 的客户个数加 1，将第一步得到的查询结果作为派生表 A，进行嵌套子查询。父查询和子查询均在表 A 上进行，依次分别命名为表 a 和表 b，子查询中对于表 a 中的一个元组，得到表 b 中满足 total_income 大于该元组 total_income 的元组个数，若个数小于 2 则表明该元组为前三名，在父查询中选择该元组。

完整代码如下所示：

```
1. select c_name,c_id_card,total_income from
2. (select c_name,c_id_card,SUM(pro_income)
3. from property,client
```

```
4. where c_id = pro_c_id and pro_status != '冻结'
5. group by c_id) a(c_name,c_id_card,total_income)
6. where (
7.         select count(1)
8.         from
9.         (select c_name,c_id_card,SUM(pro_income)
10.        from property,client
11.        where c_id = pro_c_id and pro_status != '冻结'
12.        group by c_id) b(c_name,c_id_card,total_income)
13.        where a.total_income < b.total_income
14.        )<=2
15.order by total_income desc;
```

该题更简洁的方法是使用 limit 子句，只需要求和后按照 total_income 降序排序并使用 limit 3 只输出前三行。

2.1.2 客户总资产

本关任务：查询客户理财、保险、基金投资金额的总和，并排序。

实现方法：查询客户在本行的总资产，综合客户表(client)、资产表(property)、理财产品表(finances_product)、保险表(insurance)、基金表(fund)，列出所有客户的编号、名称和总资产，总资产命名为 total_property。总资产为储蓄卡总余额，投资总额，投资总收益的和，再扣除信用卡透支的总金额(信用卡余额即为透支金额)。客户总资产包括被冻结的资产。

此处需要综合各个表的内容并根据种类不同计算出合适的金额并相加，此处采用 union all 子句，对于每一条资产信息，统一算出各资产的收益/透支金额统一命名为 pro_property，统一格式为（c_id，pro_property）的元组，之后将 client 和 union 后的表进行外连接，进行求和即可。

注意 client 表可能存在不存在资产记录的客户，此时使用左外连接可以保留悬浮元组，使用 ifnull 函数，若该客户不存在资产记录，sum(pro_property)为 null，则结果为 0。

```
1. select c_id, c_name, ifnull(sum(pro_property), 0) as total_proper
   ty
2. from client left join
3. (
4.     select pro_c_id, pro_quantity * p_amount as pro_property
5.     from property, finances_product
```

```
    where pro_type = 1 and pro_pif_id = p_id
    union all
    select pro_c_id, pro_quantity * i_amount as pro_property
    from property, insurance
    where pro_type = 2 and pro_pif_id = i_id
    union all
    select pro_c_id, pro_quantity * f_amount as pro_property
    from property, fund
    where pro_type = 3 and pro_pif_id = f_id
    union all
    select b_c_id, b_balance as pro_property
    from bank_card
    where b_type = "储蓄卡"
    union all
    select b_c_id, -b_balance as pro_property
    from bank_card
    where b_type = "信用卡"
    union all
    select pro_c_id, pro_income as pro_property
    from property
) a on c_id = pro_c_id
group by c_id
order by c_id;
```

2.1.3 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。

实现方法：此处内层查询使用 limit 子句，将 insurance 中的元组按 i_amount 排序，使用 distinct 去重，limit 3，1 表示跳过前三行取一行。外层查询中查询第四高保险金额对应的保险编号，输出保险编号和保险金额。

```
select i_id, i_amount
from insurance
where i_amount =
(
    select distinct i_amount
    from insurance
    order by i_amount desc
    limit 3,1
    -- 跳过三行取一行
)
```

2.1.4 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两种方式的排名次。

实现方法：第一条 SQL 语句实现全局名次不连续的排名，第二条 SQL 语句实现全局名次连续的排名。使用 MySQL 中内置的两个函数：rank()进行并列名次相同且跳过的标号，dense_rank()进行并列名次相同且标号连续的标号。两个 SQL 语句仅是使用函数不同，下仅以 dense_rank()为例。

```
select pro_c_id, sum(pro_income) as total_revenue,
dense_rank() over(order by sum(pro_income) desc) as `rank`
from property
where pro_type = 3
group by pro_c_id
order by sum(pro_income) desc, pro_c_id;
```

2.1.5 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。

实现方法： 这里实际上需要比较两个客户对应基金组合的元组的集合是否相等，SQL 中没有全称量词，但可以将全称量词转化成存在量词。对于客户 a 和客户 b，要求不存在 a 持有而 b 不持有的基金，且不存在 b 持有而 a 不持有的基金。使用两个 not exist 子查询来确保两个客户的基金组合完全相同。

```
select distinct a.pro_c_id c_id1,b.pro_c_id c_id2
from property a,property b
where a.pro_c_id < b.pro_c_id and a.pro_type = 3 and b.pro_type =
 3 and not exists
(
    select *
    from property c
    where c.pro_type = 3  and c.pro_c_id = b.pro_c_id and not exi
sts
    (
        select *
        from property d
        where d.pro_type = 3 and d.pro_c_id = a.pro_c_id and d.pr
o_pif_id = c.pro_pif_id
    )
)
and not exists
```

```
(
    select *
    from property c
    where c.pro_type = 3  and c.pro_c_id = a.pro_c_id and not exi
sts
    (
        select *
        from property d
        where d.pro_type = 3 and d.pro_c_id = b.pro_c_id and d.pr
o_pif_id = c.pro_pif_id
    )
);
```

2.1.6 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，则称这连续几日为投资者购买基金的高峰期。

实现方法：高峰期为连续三个交易日内基金购买总金额超过 100 万元。

具体实现方法如下：

a) 筛选超过 100 万元的交易日：首先，将单日购买金额超过 100 万元的天数筛选出来，并对这些交易日进行编号（记为 rownum）。

b) 计算交易天数：通过 MySQL 内置的 DATEDIFF 函数计算每个交易日距离 2022 年 1 月 1 日的天数（记为 daycnt）。

c) 判定连续交易日：如果是购买高峰期，那么该高峰期内的交易天数和编号必然同时连续。因此，交易日天数与编号的差值必然相同。根据这一差值进行分组，如果某一组的值个数大于等于三，则可认为是高峰期。

d) 注意到 2022 年 2 月只有第一周因为春节假期而休市，因此统计自 2022 年 1 月 1 日的交易日可以直接通过 DATEDIFF 函数快速计算出 daycnt 的值。

```
select daycnt.pro_purchase_time as pro_purchase_time, daycnt.amou
nt as total_amount
from (
    select *, count(*) over(partition by orderday.workday - order
day.rownum) cnt
    from (
        select *, row_number() over(order by workday) rownum //使
用内置函数计算 rownum
```

```
        from (
            select pro_purchase_time, sum(pro_quantity * f_amount
) as amount,
datediff(pro_purchase_time, "2021-12-31") - 2 * week(pro_purchase
_time) as workday
            from property, fund
            where pro_pif_id = f_id and pro_type = 3
and pro_purchase_time like "2022-02-%"
            group by pro_purchase_time
        )amount_by_day where amount_by_day.amount > 1000000
    ) orderday
) daycnt where daycnt.cnt >= 3;
```

2.1.7 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。

实现方法：为了以日历表的格式显示 2022 年 2 月每周每日基金购买的总金额，我们需要将资产表 (property) 和基金表 (fund) 进行自然连接，并按交易时间分组计算购买总金额。我们还需要利用 WEEK() 函数和 WEEKDAY() 函数获取交易时间所在的周次和星期几。最终通过 IF 语句将金额分配到相应的列中，并按周次分组输出。

```
SELECT week_of_trading, SUM(IF(days=0,amount,NULL)) AS Monday, SU
M(IF(days=1,amount,NULL)) AS Tuesday, SUM(IF(days=2,amount,NULL))
 AS Wednesday, SUM(IF(days=3,amount,NULL)) AS Thursday, SUM(IF(da
ys=4,amount,NULL)) AS Friday
FROM
    (SELECT week(pro_purchase_time)-5 AS week_of_trading, SUM(f_a
mount*pro_quantity) AS amount, weekday(pro_purchase_time) AS days
    FROM fund, property
    WHERE f_id = pro_pif_id AND
    pro_type = 3 AND
    pro_purchase_time >= "2022-2-7" AND
    pro_purchase_time <= "2022-2-28"
    GROUP BY pro_purchase_time) AS a
GROUP BY week_of_trading
ORDER BY week_of_trading ASC;
```

2.2 数据查询(Select)-新增

本节内容在 Select 之一的背景和基础之上难度有所提高，仍然关注于数据的

查询工作。

该任务的关卡 1~6 已全部完成，选取第 1，2，5，6 关进行阐述。

2.2.1 查询销售总额前三的理财产品

本关任务：查询 2010 年和 2011 年这两年每年销售总额前 3 名（如果有并列排名，则后续排名号跳过之前的并列排名个数，例如 1、1、3）的统计年份（pyear）、销售总额排名值(rk)、理财产品编号(p_id)、销售总额(sumamount)。

实现方法：

本关采取复杂的派生查询。从内到外，首先在 property 表中查询出年份在 2010 和 2011 的理财产品记录，再将结果作为派生表 a 与 finance_product 表根据 pro_pif_id = p_id 进行内连接，根据购买年份和理财产品编号分组，这样可以用聚集函数算出不同年份的各理财产品销售总额。最后再根据销售总额使用 rank () 函数排名记为 rk，并选出 rk 小于等于 3 的元组即为前三名。

```
select * from
(
    select pyear,rank() over(partition by pyear order by sumamount desc) as rk,p_id, sumamount
    from
    (
        select year(pro_purchase_time) as pyear, sum(pro_quantity * p_amount) as sumamount, p_id
        from
        (
            select *
            from property
            where pro_type = 1 and year(pro_purchase_time) in (2010,2011)
        ) as a inner join finances_product on pro_pif_id = p_id
        group by p_id, pyear
    ) as b
) as c
where rk <= 3;
```

2.2.2 投资积极且偏好理财类产品的客户

本关任务：购买了 3 种（同一编号的理财产品记为一种）以上理财产品的客户被认为投资积极的客户，若该客户持有基金产品种类数（同一基金编号记为相

同的基金产品种类）小于其持有的理财产品种类数，则认为该客户为投资积极且偏好理财产品的客户。查询所有此类客户的编号(pro_c_id)。

实现方法： 在表 property 上，按照 pro_c_id 分组，并采用 sum()统计基金产品(pro_type = 1)和理财产品种类数(pro_type = 3)，使用 case 子句，当 pro_type 为对应值时计数 1，否则为 0，按照题意在 having 内加入存在聚集函数的条件表达式即可。

由于同一编号的理财产品记为一种，查询时采用 distinct 去重。

```
1. select distinct pro_c_id
2. from property
3. group by pro_c_id
4. having sum(case when pro_type = 1 then 1 else 0 end)>=3 and sum(c
   ase when pro_type = 1 then 1 else 0 end)>sum(case when pro_type =
    3 then 1 else 0 end)
5. order by pro_c_id asc;
```

2.2.3 查询任意两个客户的相同理财产品数

本关任务：查询任意两个客户之间持有的相同理财产品种数，并且结果仅保留相同理财产品数至少 2 种的用户对。

实现方法：将 property 表设置别名 a 和 b，直接枚举 a、b 表中的元组代表资产信息，两个客户记为 A、B，按照两人的 pro_c_id 的值进行分组，组内元组个数即为两个客户相同的理财产品数。

注意需要给出两客户 pro_c_id 不相等的条件，计算同一个人相同理财产品数没有意义。

```
1. select a.pro_c_id pro_c_id,b.pro_c_id pro_c_id,count(1) total_cou
   nt
2. from property a,property b
3. where a.pro_pif_id = b.pro_pif_id and a.pro_type = 1 and b.pro_ty
   pe  = 1 and a.pro_c_id != b.pro_c_id
4. group by a.pro_c_id,b.pro_c_id
5. having count(1)>=2
6. order by a.pro_c_id asc;
```

2.2.4 查找相似的理财客户

本关任务：查找相似的理财客户。在某些推荐方法中，需要查找某位客户在

理财行为上相似的其他客户，不妨设其定义为：对于 A 客户，其购买的理财产品集合为{P}，另所有买过{P}中至少一款产品的其他客户集合为{B}，则{B}中每位用户购买的{P}中产品的数量为他与 A 客户的相似度值。将{B}中客户按照相似度值降序排列，得到 A 客户的相同相似度值则按照客户编号升序排列，这样取前两位客户即为 A 客户的相似理财客户列表。

查询每位客户(列名：pac)的相似度排名值小于 3 的相似客户(列名：pbc)列表，以及该每位客户和他的每位相似客户的共同持有的理财产品数(列名：common)、相似度排名值(列名：crank)。

实现方法：此处选择不断地嵌套查询和派生查询，从内到外分别，选出 A 客户购买过的理财产品，选出买过至少一款上述理财产品的其他客户并顺便计算出购买的产品数，再使用 rank over 函数来为其排序，排序值记作 crank，并选出 crank 小于 3（即前两名客户）。

```
1. select *
2. from
3. (
4. select pac,pbc,cnt common,rank() over(partition by pac order by c
   nt desc,pbc asc)as crank
5. from(
6. select a.c_id pac,b.pro_c_id pbc,count(b.pro_pif_id) cnt#买过至少
   一款理财产品的客户b,与购买的产品数
7. from client a,property b
8. where a.c_id != b.pro_c_id  and b.pro_type = 1 and b.pro_pif_id i
   n
9. (
10.select c.pro_pif_id#a 买过的理财产品
11.from property c
12.where c.pro_c_id = a.c_id and c.pro_type = 1
13.)
14.group by a.c_id,b.pro_c_id
15.)as x
16.)as y
17.where crank<3
18.order by pac asc,crank asc;
```

2.3 存储过程与事务

这一实训任务涉及到存储过程的使用，分别涉及使用流程控制语句的存储过

程、使用游标的存储过程和使用事务的存储过程。

该任务的全部关卡均已完成，选择其中较为复杂的第 1，2 关进行

2.3.1 使用流程控制语句的存储过程

本关任务：创建一个存储过程，向表 fibonacci 插入斐波拉契数列的前 n 项。

实现方法：使用一条 with 子句编写递归过程，通过此递归过程产生连续的三项 fibonacci 数，通过此语句计算 fibonacci 数列的前 n 项。注意 fibonacci 的边界条件第一项为 1。

```
1. delimiter $$
2. create procedure sp_fibonacci(in m int)
3. begin
4.     set m = m - 1;
5.     with recursive cte(id, cur, pre) as (
6.         select 0, 0, 0
7.         union all
8.         select id + 1, if (id < 2, 1, cur + pre), cur from cte wh
   ere id < m
9.     )
10.    select id n, cur fibn from cte;
11.end $$
12.delimiter ;
```

2.3.2 使用游标的存储过程

本关任务：医院的某科室有科室主任 1 名（亦为医生），医生若干（至少 2 名，不含主任），护士若干（至少 4 人）。编写一存储过程，自动安排某个连续期间的大夜班的值班表。

实现方法：护士不存在周末换班操作，因而直接使用一个游标进行不断地循环遍历，当遍历到表尾后（done 标记）再重置该游标即可。

对于医生，需要判断当天是周一还是周末。如果是周末，并且排到主任的班，将 head 标记为主任，然后游标继续遍历；如果当前是周一，并且 head 不为空，表示主任跳班到周一了，需要将主任安排到周一。其余情况与护士相同。

```
1. delimiter $$
2. create procedure sp_night_shift_arrange(in start_date date, in en
   d_date date)
3. begin
```

```
4.      declare done, tp, wk int default false;
5.      declare doc, nur1, nur2, head char(30);
6.      declare cur1 cursor for select e_name from employee where e_t
   ype = 3;
7.      declare cur2 cursor for select e_type, e_name from employee w
   here e_type < 3;
8.      declare continue handler for not found set done = true;
9.      open cur1; open cur2;
10.     while start_date <= end_date do
11.         fetch cur1 into nur1;
12.         if done then
13.             close cur1; open cur1; set done = false; fetch cur1 i
   nto nur1;
14.         end if;
15.         fetch cur1 into nur2;
16.         if done then
17.             close cur1; open cur1; set done = false; fetch cur1 i
   nto nur2;
18.         end if;
19.         set wk = weekday(start_date);
20.         if wk = 0 and head is not null then //当天周一，主任跳班
21.             set doc = head; set head = null;
22.         else
23.             fetch cur2 into tp, doc;
24.             if done then //重置游标
25.                 close cur2; open cur2; set done = false; fetch cu
   r2 into tp, doc;
26.             end if;
27.             if wk > 4 and tp = 1 then //当天周末，排到主任，将主任移
   动到周一
28.                 set head = doc; fetch cur2 into tp, doc;
29.                 if done then //重置游标
30.                     close cur2; open cur2; set done = false; fetc
   h cur2 into tp, doc;
31.                 end if;
32.             end if;
33.         end if;
34.         insert into night_shift_schedule values (start_date, doc,
    nur1, nur2);
35.         set start_date = date_add(start_date, interval 1 day);
36.     end while;
37. end$$
38. delimiter ;
```

2.4 触发器

本节主要内容为触发器的应用。

该实训只有一个关卡，已全部完成，下面对该关卡进行分析。

2.4.1 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：为资产表 property 编写一个触发器，以实现任务所要求的完整性业务规则。完整性业务规则：

1) 如果 pro_type=1，则 pro_pif_id 只能引用 finances_product 表的 p_id 。

2) 如果 pro_type = 2，则 pro_pif_id 只能引用 insurance 表的 i_id 。

3) 如果 pro_type = 3，则 pro_pif_id 只能引用 fund 表的 f_id。

实现方法：声明 BEFORE INSERT ON property 类型的触发器，使用 if 语句，首先判断是否存在该类型的投资产品，然后判断该 p_id 是否为上述规则对应类型的投资产品，如果报错信息为空则说明没有错误。

```
delimiter $$
CREATE TRIGGER before_property_inserted BEFORE INSERT ON property
 FOR EACH ROW
BEGIN
    declare message varchar(50);
    if (new.pro_type = 1) then
        if (new.pro_pif_id not in (select p_id from finances_prod
uct)) then
            set message=concat("finances product #",new.pro_pif_i
d," not found!");
        end if;
    elseif(new.pro_type = 2) then
        if (new.pro_pif_id not in (select i_id from insurance)) t
hen
            set message = concat("insurance #", new.pro_pif_id, "
 not found!");
        end if;
    elseif(new.pro_type = 3) then
        if (new.pro_pif_id not in (select f_id from fund)) then
            set message = concat("fund #", new.pro_pif_id, " not
found!");
        end if;
    else
```

```
18.         set message = concat("type ", new.pro_type, " is illegal!
    ");
19.     end if;
20.     if (message is not null) then
21.         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
22.     end if;
23. END$$
24. delimiter ;
```

2.5 用户自定义函数

本节的主要应用为用户自定义函数的创建和使用。用户自定义函数能够将特定操作模块化，且方便后续的多次调用。通过本实验，我们可以了解用户自定义函数的编写方法。

该任务全部关卡已完成，由于只含一个关卡，下面即对该关卡进行分析

2.5.1 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。

实现方法：使用 create function 语句创建统计给定客户编号，统计存储总额的函数（函数内部使用 select 语句实现）。第 11 行与第 12 行是使用 SELECT 语句使用该函数的代码。

```
1. delimiter $$
2. create function get_deposit(client_id int) returns numeric(10,2)
3. begin
4.     return (
5.         select sum(b_balance) from bank_card
6.         where b_type = "储蓄卡"
7.          group by b_c_id having b_c_id = client_id
8.     );
9. end$$
10. delimiter ;
11. select c_id_card, c_name, get_deposit(c_id) as total_deposit from
    client
12. where get_deposit(c_id) >= 1000000 order by total_deposit desc;
```

2.6 安全性控制

MySQL 采用自主存取控制机制进行安全性管理。通过用户，数据对象，权

限，授权，收回权限等要素进行存取控制。另外，为了方便批量授权给同一类用户，引入了角色。本实训就是对上述要素加以利用，实现数据库的安全性控制。

本实验共两关，现已全部完成，下面选取第二关来进行分析。

2.6.1 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。

实现方法：根据 create 和 grant 语句按要求实现即可。

```
# (1) 创建角色 client_manager 和 fund_manager;
create role client_manager, 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_c_id, b_number, b_type)
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;
```

2.7 并发控制与事务的隔离级别

数据库在运行过程中，需要保持其数据一致性，本实验可体现出事务并发访问数据时产生的不一致性。本节的 6 个关卡涉及数据库中并发控制与事务的隔离级别的内容，包括隔离级别的设置，事务的开启、提交和回滚等，还通过在合适的位置和时机添加等待代码以实现在隔离级别不够的各种场景下产生读脏、不可重复读、幻读等出错情况。

该实训一共六个关卡，已经全部完成，下面选取部分重要关卡（第 2、5 关）进行分析。

2.7.1 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。

实现方法：读脏，即读到“脏数据”，是多事务并发执行导致的读错误。首先事务 2 修改数据，之后事务 1 读数据，最后事务 2 回滚，此时事务 1 读到的数据就和数据库中内容不一致，是脏数据。模拟上述过程，用 sleep()来调整他们的运行顺序。注意 sleep()函数的返回值必须要赋值给一个变量，否则会报错。

```
事务 1：
set @n = sleep(1);
select tickets from ticket where flight_no = 'CA8213';
事务 2：
update ticket set tickets = tickets - 1 where flight_no = 'CA8213
';
set @n = sleep(2);
rollback;
```

2.7.2 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。

实现方法：使用 for share 和 for update 分别对数据加 S 锁和 X 锁。由于事务 2 尝试在事务 1 的两次操作之间进行修改，因而当事务 1 加上 X 锁后，事务 2 无法在事务 1 进行过程中打断进行修改，因而保证了事务 1 的可重复读。

```
事务 1
use testdb1;
set session transaction isolation level read uncommitted;
start transaction;
-- 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);
select * from ticket;

事务 2
use testdb1;
set session transaction isolation level read uncommitted;
start transaction;
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 set tickets = tickets - 1 where flight_no = 'MU2455
';
commit;
```

2.8 备份+日志：介质故障与数据库恢复

2.8.1 备份与恢复

本关任务：设有居民人口登记数据库 residents,请为该数据库做一次静态的(你一个人独享服务器)海量逻辑备份，备份文件命名为 residents_bak.sql。 然后再用该逻辑备份文件恢复数据库。

实现方法：首先使用 mysqldump 命令将 residents 备份到 residents_bak.sql 中：

```
mysqldump -h127.0.0.1 -uroot --databases residents > residents_bak.sql
```

再使用该备份恢复：

```
mysql -h127.0.0.1 -uroot < residents_bak.sql
```

2.8.2 备份+日志：介质故障的发生与数据库的恢复

本关任务：摸拟介质故障的发生与数据库的恢复。

实现方法：由于需要对数据库 train 作逻辑备份并新开日志文件，所以需要在上一关的基础上 mysqldump 指令中加入--flush-logs 参数来新开日志文件：

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

系统故障前的日志文件保存在 log/binlog.000018 中，因而除了 checkpoint 时刻产生的 train_bak.sql 文件中的数据，还需要根据日志文件对部分事务进行重做和撤销操作。由于 mysql 的默认字符集为 utf-8，与日志文件不兼容，因而需要使用--no-defaults 参数来取消 MySQL 默认参数。

```
mysql -h127.0.0.1 -uroot < train_bak.sql;
mysqlbinlog --no-defaults log/binlog.000018 | mysql -uroot;
```

2.9 数据库设计与实现

数据库设计一般包括以下几个主要环节：需求分析、概念结构设计、逻辑结构设计、物理结构设计，以及实施和运维。由于物理结构设计和实施运维过程较

为复杂，并且需求已经确定，因此本次实训将主要关注概念结构设计和逻辑结构设计。任务的核心是评估数据库设计的过程，即从需求（概念模型）到具体的关系模式（E-R 图）的转变，并在 MySQL 中使用 DDL 来实现这些关系。

2.9.1 从概念模型到 MySQL 实现

本关任务：根据一个已建好的逻辑模型，完成 MySQL 的实现。

实现方法：根据给出的 E-R 图（E-R 图如图 2.1 所示），对于每个实体和联系，仔细分析 E-R 图中的联系，发现全都是一对多的联系，可以将“一”端关系模式的码作为外码加到“多”端的关系模式中，这样可以减少表的数目且不造成存储空间的浪费。

由于代码较冗长，此处不予展示。

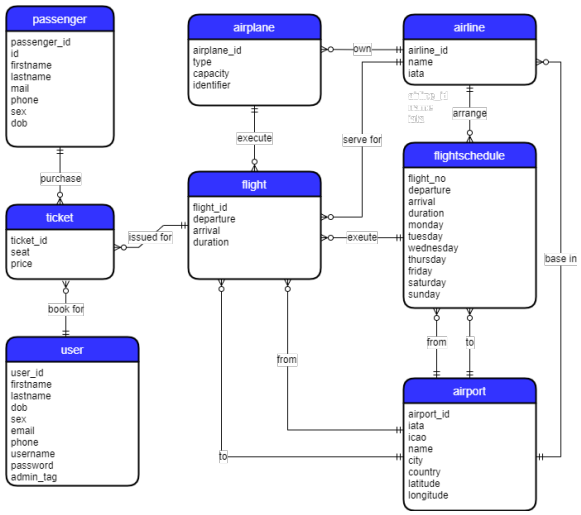

图 2.1 机票订票系统概念模型 ER 图

2.9.2 从需求分析到逻辑模型

本关任务：按影院管理系统要求画出 E-R 图，并给出对应的关系模式。

实现方法：首先找出实体，该问题中实体一共五个，分别是 movie，customer，hall，schedule 和 ticket，并给出了各自的属性。之后分析实体之间的联系，再将这些联系转换成对应的关系，最终设计出我们的 ER 图，如图 2.2 所示。

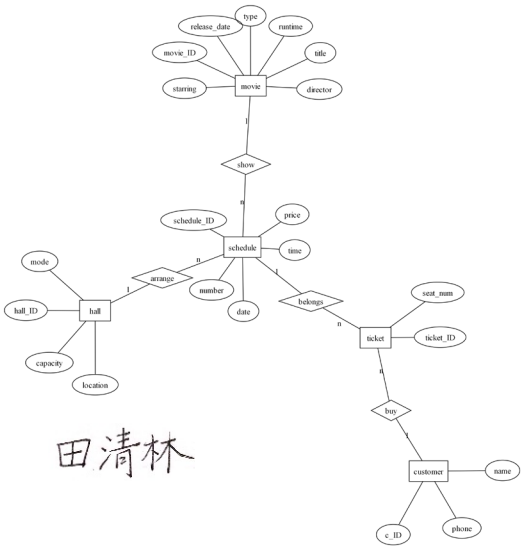

图 2.2 影院管理系统 ER 图

并有如下的关系模式：

```
1. movie(movie_ID, title, type, runtime, release_date, director, sta
   rring),
2. primary key:(movie_ID);
3. customer(c_ID, name, type, phone), primary key:(c_ID);
4. hall(hall_ID, mode, capacity, location), primary key:(hall_ID);
5. schedule(schedule_ID, date, time, price, number, movie_ID, hall_I
   D),
6. primary key:(schedule_ID), foreign key(movie_ID, hall_ID);
7. ticket(ticket_ID, seat_num, schedule_ID), primary key(ticket_ID),
8. foreign key(schedule_ID);
```

2.9.3 建模工具的使用

本关任务：将一个建好的模型文件，利用 MySQLWorkbench 的 forward engineering 功能，自动转换成 SQL 脚本。

实现方法：可以使用软件自动生成相应的 SQL 脚本，例如 erwin Data Modeler。由于头歌平台已经提供了一个已建立的模型文件 rbac.mwb，我们可以直接使用该文件，通过 MySQL Workbench 中的建模模块中的 Forward Engineering 功能，自动导出 SQL 脚本即可。

2.9.4 制约因素分析与设计

受限于课程复杂度，我们机票订票系统简化了很多。

首先，在概念模型设计中，我们考虑得较为宽泛，仅将用户类型分为普通用

户和管理用户，但未进行更细致的划分。例如，普通用户中可能因为年龄不同而分为大人、老人和小孩。

其次，对于购票用户来说，一张机票仅可以让一人乘坐，而由一人代为其购买，因而在购买机票信息不仅需要记录乘机人的信息，还需要记录购票人的信息。

另外，为了简化设计，我们未显示实际飞行信息。这样做一方面帮助我们设计出更加简洁、表达含义更清晰的概念模型，但另一方面，由于描述过于模糊，在将其转换为 ER 图时会遇到困难。

对于用户这个实体，与其设计为两种类型，不如将其分成两个实体，因为购票人与飞机管理员的属性没有特殊联系。普通用户可能根本不需要查看机密信息，但却占用了权限的存储空间，这不仅造成了空间浪费，还使得概念的表达更加模糊。

2.9.5 工程师责任及其分析

作为工程师，解决问题时，应当全面考虑各种可能的影响因素，理解业务需求，并转化为数据库设计需求，以确保解决方案全面、协调、可持续。同时要并注重数据安全的保护，管理数据库用户和权限，确保只有授权用户能够访问和操作数据，定期审计数据库活动，发现和解决潜在的安全隐患。同时也应持续学习与改进，不断分析和改进数据库性能，提升系统的整体效率和用户体验。

在社会责任方面，工程师应当努力满足社会需求，并致力于解决环境问题和保护用户健康、安全和隐私。同时也应学习相关法律法规，并在解决任务时遵守法律法规，杜绝违法违纪行为。

3 课程总结

在这次实验课程中，我们主要通过 MySQL 语言实现了数据库的建立、修改、查询、安全性、并发性等实际应用中的数据库操作训练。此外，还涉及数据库设计、数据库的 Java 应用开发等数据库高阶内容应用

在实验过程当中，最困扰我的任务就是数据查询，对于较难的关卡百思不得其解，花费了大量时间 debug，最后也是通过和其他同学进行交流后得到了大致的解题思路，才得以写出代码。这一过程锻炼了我的思维能力和对查询语句的熟练程度。

实验过程中我学习了不少拓展内容，。例如在事务存储过程中，由于课内未涉及该部分，因而在实现该节的时候查阅了大量的资料、付出了大量的时间后才得以完成。

本实验最有意思的地方在于并发控制的读脏、幻读、不可重复读现象的设计，让学生通过亲自设置不同事务的等待时间对事务进行调度，来实现不同隔离级别下的错误并行处理情况，以此能更好的帮助学生掌握隔离级别的正确应用。这加深了我对课内知识的理解，激发了我的兴趣。

通过本次实验的训练，让我熟练掌握了 MySQL 语句的使用，并充分感受到了 MySQL 语句的灵活性和使用便捷性，并且掌握了如何正确使用 MySQL 语句实现复杂的查询要求，也提高了我分析用户需求的能力。

在这次实践中，我认为实验在一些拓展内容上的引导不明确，比如没有对 MySQL 中一些广泛使用且便捷的函数（例如 ifnull()，group_concat()）和提供简明扼要的介绍，使得在一些关卡中本人花费了大量时间尝试使用局限的课本知识解决问题，给出的答案都较为复杂。当然，深入思考的过程也让我受益匪浅，加深了对课本知识的理解。

总之，本次数据库系统原理实践课程让我受益匪浅。我深刻体会到了理论与实践相结合的重要性。通过实际操作，不仅巩固了课堂上的知识，还发现并解决了许多实际问题， 从而反过来加深对课堂知识的理解，正如王阳明所说： “知者行之始，行者知之成。”本次实验经历对于我未来的学习和工作都有很大的帮助。