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 2001 班
学　　号：	U201914858
姓　　名：	王怡彬
指导教师：	谢美意

分数	
教师签名	

2023　年 1　月 2　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

该课程是为了配合"数据库系统原理"课程而开设的实践课，旨在帮助学生在实践中加深对数据库系统原理的理解。该课程以 MySQL 语言为例，并重视理论与实践的结合。课程内容涵盖了 MySQL 的数据对象管理与编程、数据处理任务、数据库的系统内核（如安全性控制、完整性控制、恢复机制、并发控制机制）、数据库的设计与实现、以及数据库应用系统的开发。

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)

使用 MySQL 语言实现数据库、表与完整性约束的定义。掌握 create 及其相关语句的使用。

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

该关卡任务已完成，实施情况本报告略过

2.1.3 创建外码约束(foreign key)

该关卡任务已完成，实施情况本报告略过

2.1.4 CHECK 约束

该关卡任务已完成，实施情况本报告略过

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)

使用 MySQL 语言实现表结构与完整性约束的修改。掌握 alter 及其相关语句的使用。

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

该关卡任务已完成，实施情况本报告略过

2.2.3 修改字段

该关卡任务已完成，实施情况本报告略过

2.2.4 添加、删除与修改约束

该关卡任务已完成，实施情况本报告略过

2.3 数据查询(Select)之一

使用 MySQL 语言实现数据查询。掌握 select 及其相关语句的使用。

2.3.1 金融应用场景介绍,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

使用嵌套的子查询完成多表多条件的条目筛选。第二个子查询查询出所有具有 pro_type=2 和在上一个子查询查询出的 pro_c_id 列表中的客户的 pro_c_id。最后，使用最外层的 SELECT 语句查询出所有满足条件的客户的对应信息，并按照 c_id 排序，代码如图 2.1 所示。

```
SELECT c_name,c_mail,c_phone
FROM client
WHERE c_id IN
    (SELECT pro_c_id
    FROM property
    WHERE pro_type = 2 AND c_id IN
                        (SELECT pro_c_id
                        FROM property
                        WHERE pro_type = 3))
ORDER BY c_id;
```

图 2.1 任务 2.3.3 代码

2.3.4 办理了储蓄卡的客户信息

在 select 语句的 from 子句中查询两个表 client 和 bank_card 以实现多表连接，并在 where 子句中使用 and 操作对多条件进行查询，并按照 c_id 排序，代码如图 2.2 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;
```

图 2.2 任务 2.3.4 代码

2.3.5 每份金额在 30000～50000 之间的理财产品

该任务较为简单，此处不再赘述，代码如图 2.3 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;
```

图 2.3 任务 2.3.5 代码

2.3.6 商品收益的众数

具体来说，首先使用 GROUP BY 子句按照 pro_income 列分组，然后使用 COUNT 函数统计每组的行数。接下来使用 HAVING 子句过滤出出现次数最多的

组，这里使用了 ALL 关键字和一个子查询，以实现只保留每组的行数都大于等于子查询中每组的行数的各组，即众数的筛选。最后，使用 SELECT 语句显示 pro_income 列和每组的行数，代码如图 2.4 所示。

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

图 2.4 任务 2.3.6 代码

2.3.7 未购买任何理财产品的武汉居民

使用 like 子句和 not exists 谓词排除掉存在 pro_type=1 的记录的客户，代码如图 2.5 所示。

```
select c_name, c_phone, c_mail
from client
where c_id_card like '4201%' and not exists
    (SELECT *
    FROM property
    WHERE pro_type = 1 and c_id = pro_c_id)
ORDER BY c_id;
```

图 2.5 任务 2.3.7 代码

2.3.8 持有两张信用卡的用户

首先使用一个子查询查询出拥有至少两张信用卡的客户，然后使用 WHERE 子句和 IN 运算符将查询结果用作条件，查询出满足条件的客户的对应信息，代码如图 2.6 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;
```

图 2.6 任务 2.3.8 代码

2.3.9 购买了货币型基金的客户信息

与上一关类似，使用子查询查询出拥有货币型基金且 pro_type=3 的客户，然后使用 WHERE 子句和 IN 运算符将查询结果用作条件，查询出满足条件的客户的对应信息。代码如图 2.7 所示。

```
SELECT c_name,c_phone,c_mail
FROM client
WHERE c_id IN
    (SELECT pro_c_id
    FROM fund,property
    WHERE pro_pif_id = f_id AND f_type = '货币型' and pro_type = 3)
ORDER BY c_id;
```

图 2.7 任务 2.3.9 代码

2.3.10 投资总收益前三名的客户

首先使用子查询和派生表查询出查询出每个客户的 id 和投资总收益，并将结果进行排序，最后使用 LIMIT 子句限制只显示前三条记录。代码如图 2.8 所示。

```
SELECT c_name, c_id_card, Sum_pro.sum_income AS total_income
FROM (SELECT pro_c_id, SUM(pro_income)
    FROM
        (SELECT *
        FROM property
        where pro_status = '可用') as temp1
    GROUP BY pro_c_id
   ) AS Sum_pro(id, sum_income), client
WHERE c_id = id
ORDER BY Sum_pro.sum_income DESC limit 3;
```

图 2.8 任务 2.3.10 代码

2.3.11 黄姓客户持卡数量

使用外连接 left outer join 实现在两表连接时将无记录项置为 0 的操作(没有办卡的卡数量计为 0)，使用 like 语句对客户名进行筛选以找出黄姓客户，并利用 group by 子句对其进行分组统计。代码如图 2.9 所示。

```
select c_id, c_name, count(b_number) as
number_of_cards
from client left outer join bank_card on
(client.c_id = bank_card.b_c_id)
where c_name like '黄%'
group by c_id
order by number_of_cards desc, c_id
```

图 2.9 任务 2.3.11 代码

2.3.12 客户理财、保险与基金投资总额

使用外连接 left outer join 和 ifnull 函数实现在两表连接时将无记录项置为 0 的操作，并多次使用外连接和子查询统计各类(理财,保险,基金)资产投资金额，最

后在 select 语句中对各项投资金额进行加和以计算投资总额。代码如图 2.10 所示。

```
select c_name, c_id_card, ifnull(ii_n, 0) + ifnull(ff_n, 0) + ifnull(pp_n, 0) as total_amount
from (((client left outer join
(
    select pro_c_id, sum(pro_quantity * i_amount)
    from property join insurance
    on(property.pro_pif_id = insurance.i_id)
    where pro_type = 2
    group by pro_c_id
) as i_n(ii_id, ii_n) on (c_id = i_n.ii_id))
left outer join
(
    select pro_c_id, sum(pro_quantity * f_amount)
    from property join fund
    on(property.pro_pif_id = fund.f_id)
    where pro_type = 3
    group by pro_c_id
) as f_n(ff_id, ff_n) on (c_id = f_n.ff_id))
left outer join
(
    select pro_c_id, sum(pro_quantity * p_amount)
    from property join finances_product
    on(property.pro_pif_id = finances_product.p_id)
    where pro_type = 1
    group by pro_c_id
) as p_n(pp_id, pp_n) on (c_id = p_n.pp_id))
order by total_amount desc;
```

图 2.10 任务 2.3.12 代码

2.3.13 客户总资产

与上一关类似，使用外连接 left outer join 和 ifnull 函数实现在两表连接时将无记录项置为 0 的操作，并多次使用外连接和子查询统计多表项的数据和，最后在 select 语句中对多项数据进行加减操作运算以计算得出客户总资产。由于代码过长且内容与上一关大体重复，此处不再展示代码。

2.3.14 第 N 高问题

首先使用子查询查询出保险账户金额排名第四的保险的 i_amount。然后使用 WHERE 子句将查询结果用作条件，查询出满足条件的数据项。

在子查询中使用 GROUP BY 子句以实现消除重复数据项；使用 ORDER BY 子句进行排序；使用 LIMIT 子句，并将其参数设为 3，1。以跳过前三条记录，取第四条记录。代码如图 2.11 所示。

```
SELECT i_id, i_amount
FROM insurance
WHERE i_amount = (
    SELECT i_amount
    FROM insurance
    GROUP BY i_amount
    ORDER BY i_amount DESC
    LIMIT 3, 1
);
```

图 2.11 任务 2.3.14 代码

2.3.15 基金收益两种方式排名

第一部分根据总收益进行倒序排序并使用 rank 函数为每个 id 分配一个名次（rank），利用 rank 函数特性实现总收益相同时名次也相同但不连续。第二部分与第一部分主体相同，但利用 dense_rank 函数特性实现总收益相同时名次也相同且连续。代码如图 2.12 所示。

```sql
-- (1) 基金总收益排名(名次不连续)

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

-- (2) 基金总收益排名(名次连续)

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 total_revenue desc, pro_c_id;
```

图 2.12 任务 2.3.15 代码

2.3.16 持有完全相同基金组合的客户

该任务关卡跳过

2.3.17 购买基金的高峰期

该任务关卡跳过

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

该任务关卡跳过

2.3.19 以日历表格式显示每日基金购买总金额

该任务关卡跳过

2.4 数据查询(Select)之二

使用 MySQL 语言实现数据查询。掌握 select 及其相关语句的使用。

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

该关卡任务已完成，实施情况本报告略过

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)

使用 MySQL 语言实现表数据的插入、修改与删除。掌握 Insert,Update,Delete 及其相关语句的使用。

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

该任务较为简单，此处不再赘述，代码如图 2.13 所示。

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

图 2.13 任务 2.5.2 代码

2.5.3 批量插入数据

该关卡任务已完成，实施情况本报告略过

2.5.4 删除没有银行卡的客户信息

该任务较为简单，此处不再赘述，代码如图 2.14 所示。

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

图 2.14 任务 2.5.4 代码

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

该任务较为简单，此处不再赘述，代码如图 2.15 所示。

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

图 2.15 任务 2.5.6 代码

2.6 视图

使用 MySQL 语言实现视图 view 的创建和基于视图的查询。掌握 create view 及其相关语句的使用。

2.6.1 创建所有保险资产的详细记录视图

该关卡任务已完成，实施情况本报告略过

2.6.2 基于视图的查询

该任务较为简单，此处不再赘述，代码如图 2.16 所示。

```
select
    c_name,
    c_id_card,
    sum(i_amount * pro_quantity) 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;
```

图 2.16 任务 2.6.2 代码

2.7 存储过程与事务

使用 MySQL 语言实现存储过程的创建和使用。掌握使用流程控制语句、游标和事务三种不同的存储过程。

2.7.1 使用流程控制语句的存储过程

在存储过程中，使用了一个 WITH RECURSIVE 子句和 CTE（递归查询）来生成斐波拉契数列。首先定义了 CTE 的初始值为(0, 0, 0)，三个数值分别表示数列索引、当前值和前一个值，然后使用 UNION ALL 语句将当前的 CTE 和其他行结合起来。每一行的值都通过使用 IF 函数来计算：如果 id 小于 2，则当前值为 1；否则，当前值等于前一个值（pre）加上当前值（cur）。最后进行更新，将当前值赋值给前一个值。代码如图 2.17 所示。

```
drop procedure if exists sp_fibonacci;
delimiter $$
create procedure sp_fibonacci(in m int)
begin
    set m = m - 1;
    -- 定义递归查询初始值为 (0, 0, 0)，id 表示数列索引，cur 和 pre 分别表示当前值和前一个值
    with recursive cte (id, cur, pre) as
    (
        select
            0, 0, 0
        union all
        select
            id + 1,
            -- 当 id 小于 2，当前值等于 1；否则，等于 cur + pre
            if (id < 2, 1, cur + pre),
            cur -- 前一个值等于当前值
        from cte
        where id < m
    )
    select
        id n,
        cur fibn
    from cte;
end $$

delimiter ;
```

图 2.17 任务 2.7.1 代码

2.7.2 使用游标的存储过程

在循环中，它将每天的值班医生、护士分别从员工表中按照类型分类查询，并将每天的值班信息插入到 night_shift_schedule 表中。循环会继续直到开始日期超过结束日期为止。

首先定义变量：光标 cur1 和 cur2、医生 doctor、护士 nurse1 和 nurse2、标志变量 done 和 typ、以及一个字符变量 h。

然后分别打开光标 cur1 和 cur2。接下来，使用 while 循环，循环条件是 start_date 小于等于 end_date，每次循环都将 start_date 增加 1 天。在循环中，首先从 cur1 中获取两个护士的信息，并将这些信息赋值给 nurse1 和 nurse2。然后计算当前日期的星期数，如果是周日且 h 不为空，则将 h 的值赋给 doctor，并将 h 置空；否则，从 cur2 中获取医生的信息，并将这些信息赋值给 doctor 和 typ。如果当前日期是周五、周六、周日，并且 typ 等于 1，则将 doctor 的值赋给 h，并再次从 cur2 中获取医生的信息，然后将这些信息赋值给 doctor 和 typ。最后，将某天的日期、doctor、nurse1、nurse2 的信息插入 night_shift_schedule 表中。在循环结束后，关闭光标 cur1 和 cur2。

由于代码较长，此处不贴出代码。

2.7.3 使用事务的存储过程

该关卡任务已完成，实施情况本报告略过

2.8 触发器

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

为了避免插入数据不合法导致的错误，我们需要在插入事件执行之前触发触发器，并对每一条要插入的数据进行检查。为了更好地反馈出错信息，可以先定义用于输出错误信息的字符串，然后依次检查数据。

2.9 用户自定义函数

掌握 MySQL 语言中函数的定义并调用函数。

2.9.1 创建函数并在语句中使用它

要计算某个客户所有储蓄卡的存款总数，需要先把客户表和银行卡表连接起来。连接的条件是客户的 id 和银行卡的 id 相同，并且银行卡类型是储蓄卡。然后用 sum 聚集函数统计存款总数并返回。调用较为简单，此处不再赘述，函数体代码如图 2.18 所示。

```
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"
        group by b_c_id
        having b_c_id = client_id
    );
end$$
delimiter ;
```

图 2.18 任务 2.9.1 代码

2.10 安全性控制

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

该关卡任务已完成，实施情况本报告略过

2.11 并发控制与事务的隔离级别

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

将事务的隔离级别设为 READ COMMITTED 以实现读脏。在事务 2 修改数据之后，使用 sleep 函数等待指定量的时间以确保事务 1 能够在事务 2 将修改撤销之前读取到修改后的数据。

2.11.3 不可重复读

为了在事务之间实现不可重复读，需要设置事务的隔离级别为 READ COMMITTED，然后先在事务 1 中修改数据并使用 sleep 函数等待一段时间，使得事务 2 在修改之前能够读到一次数据；提交事务 1，之后令事务 2 使用 sleep 函数等待指定量的时间后再次读取该数据，即可使得事务 2 的两次读取到的数据不一致。

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

该关卡任务已完成，实施情况本报告略过

2.11.6 可串行化

该关卡任务已完成，实施情况本报告略过

2.12 备份+日志：介质故障与数据库恢复

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

该关卡任务已完成，实施情况本报告略过

2.13 数据库设计与实现

2.13.1 从概念模型到 MySQL 实现

根据任务要求，按需对各个主体创建表，并按需指定各个主体的主码、外码和其他码的缺省值、不可空性、不可重性等。由于代码过长且任务较为简单，此处不再赘述，也不贴出代码。

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

在解决方案设计中，应当考虑以下制约因素：

社会因素：用户需求；

健康因素：任务对用户健康无影响；

安全因素：数据安全和网络安全；

法律因素：符合本地法律法规，遵守用户隐私权；

文化因素：不同文化背景下的用户使用习惯；

环境因素：不会对环境造成负面影响。

2.13.5 工程师责任及其分析

解决问题时，应当全面考虑各种可能的影响因素，以确保解决方案全面、可持续，能够满足用户需求，并保护用户健康、安全和隐私。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。

2.14 数据库应用开发(JAVA 篇)

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

2.12.2 用户登录

该关卡任务已完成，实施情况本报告略过

2.12.3 添加新客户

该关卡任务已完成，实施情况本报告略过

2.12.4 银行卡销户

该关卡任务已完成，实施情况本报告略过

2.12.5 客户修改密码

该关卡任务已完成，实施情况本报告略过

2.12.6 事务与转账操作

首先使用 PreparedStatement 对象执行一个 SQL 查询，检查转出账号是否存在，以及该账号是否是信用卡或者账户余额是否足够支付转账金额。如果满足任意一种情况，则会返回 false；然后，程序再次使用 Prepared Statement 对象执行一个 SQL 查询，检查转入账号是否存在。如果不存在，则会返回 false；最后，判断转入账号的类型是否为信用卡，如果是，则将转入账号的余额减去转账金额；如果不是，则将转入账号的余额加上转账金额。最后，将转出账号的余额减去转账金额并更新数据库中的信息。由于代码较长，此处不贴出代码。

2.12.7 把稀疏表格转为键值对存储

先通过 JDBC 驱动连接到 MySQL 数据库，然后执行一个查询语句，获取 entrance_exam 表中的数据。接下来遍历结果集，对于每一行数据，检查其中的每一个成绩列，如果该成绩不为空，就执行一个插入操作，将该成绩插入到另一张表 sc 中。由于代码较长，此处不贴出代码。

2.15 数据库的索引 B+树实现

完成 B+树索引的数据结构，并实现它的内部节点和叶子节点类型；然后分别实现索引的插入和删除功能，以及 B+树迭代器。

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

3 课程总结

在本次课程实践任务涵盖了MySQL的数据对象管理与编程、数据处理任务、数据库的系统内核（如安全性控制、完整性控制、恢复机制、并发控制机制）、数据库的设计与实现、以及数据库应用系统的开发。我根据课程任务要求按需完成了对应的任务量，达到了本次课程实践任务的要求。

各个部分的主要工作情况如下：

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

该部分是数据库管理的基础，任务较为简单但重要，全部完成。

2. 数据查询，数据插入、删除与修改等数据处理相关任务

数据的增删改查是数据库管理的核心，除数据查询少量任务外其他均全部完成。

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

除个别关卡跳过外，基本上全部完成

4. 数据库的设计与实现

根据任务要求初步完成了从概念模型到MySQL的实现，跳过了从需求分析到逻辑模型以及建模工具的使用。

5. 数据库应用系统的开发

全部完成，实现了用java语言完成一个基础数据库应用系统的开发。

6. 数据库的索引B+树实现

该部分任务选择了跳过。

在本次课程实践任务中，我通过实验将课本上的理论知识转化为了实践结果，在实践中进一步增进了对理论知识的认知。例如，在数据查询这一任务系列中，通过不断练习数据查询这个最常用的功能，也不断深化了我对MySQL数据查询逻辑的认知；而通过使用 java 语言开发一个基础的数据库应用系统，也进一步加深了我对数据库应用系统的理解。

在这次实践中，我认为不足的地方是实验没有为MySQL一些便捷的函数（例如ifnull()）提供简明扼要的介绍，使得在一些关卡中花费了大量时间查阅资料以学习MySQL语言在课本和实验指导中都尚未提及的语法和函数。但总体上本次实验任务内容灵活，可以主动选择自己感兴趣的方向进行学习和实践，这一点我非常感谢。

附录