華中科技大學

数据库系统原理实践报告

专　　业：	计算机科学与技术
班　　级：	CS2108
学　　号：	I202120037
姓　　名：	郑澍频
指导教师：	袁平鹏教授

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

本此课程“数据库系统原理实践课”是以 MySQL 为例，设计了一系列的实训任务，其中包括：

1) 数据库、表、完整性约束的创造与修改

   对应实训：实训 1、实训 2

2) 数据查询，数据插入、删除与修改等数据处理相关任务

   对应实训：实训 3、实训 4、实训 5

   实验目的：

3) 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；

   对应实训为：实训 6、实训 7、实训 8

4) 数据库的安全性控制，恢复机制，并发控制机制等系统内核的实验；

   对应实训为：实训 9、实训 10、实训 11、实训 12

5) 数据库的设计与实现

   对应实训为：实训 13

6) 数据库应用系统的开发(JAVA 篇)。

   对应实训为：实训 14

通过本课程，熟练掌握 SQL 语言一步步完成任务。

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 2.1~2.14 任务阐述其完成过程中的具体工作。

2.1 数据库、表与完整性约束的定义(Create)

熟练使用 SQL 语句对数据库和表进行创建，添加，修改，删除等操作，以及了解掌握一些常用的约束语句。本任务已完成了所有关卡

2.1.1 创建数据库

本关任务：创建用于 2022 年北京冬奥会信息系统的数据库: beijing2022。

开始本关卡之前需要连接到数据库，在命令行输入：

Mysql -h host -u user -ppassword dbname

然后创建数据库：create database beijing2022;

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。

```
mysql> create database TestDb;
mysql> use TestDb;
mysql> create table t_user
(
    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）。

```
mysql> use MyDb;
mysql> create table dept(
    deptNo int primary key,
    deptName varchar(32)
);

mysql> create table 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

```
mysql> create database MyDb;
mysql> use MyDb;
mysql> create table 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

熟练使用 SQL 语句对表结构和完整性约束进行修改操作。本任务已完成所有关卡。

2.2.1 修改表名

本关任务：修改表的名称。

```
ALTER TABLE your_table rename to my_table;
```

2.2.2 添加与删除字段

本关任务：为表添加和删除字段。

```
#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

本关任务：修改字段。

```
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、删除与修改约束。

```
#(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

本任务目标是学会使用各种查询方式。本任务已完成所有关卡。

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

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。

错误：一开始直接用 AND 语句写了条件 WHERE property.pro_type=2 AND property.pro_type=3; 是错误的。因为在 property 关系里的一个数据元组不可能同时有 pro_type=2 和 pro_type=3 的出现，而是分成两个元组记录。

纠错：如图所示。

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

该关卡任务已完成，实施情况本报告略过。

2.3.7 未购买任何理财产品的武汉居民

该关卡任务已完成，实施情况本报告略过。

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

该关卡任务已完成，实施情况本报告略过。

2.3.11 黄姓客户持卡数量

该关卡任务已完成，实施情况本报告略过。

2.3.12 客户理财、保险与基金投资总额

本关任务：查询客户理财、保险、基金投资金额的总和，并排序。

使用左连接把 client 和 property、finance_product、insurans、fund 连接起来组成一个新的 TABLE union_table。再 union_table 中查询所需信息并根据 c_id 分组以及根据总额降序排序。

```
-- output listafter sum is total_amount = amount per property type * property quantity
SELECT c_name, c_id_card, SUM(ifnull(p_amount, IFNULL(i_amount, IFNULL(f_amount, 0)))*IFNULL(pro_quantity,0)) AS total_amount
FROM(
-- SELECT output list FROM union_table
-- which union_table include table client, property, finance_property, insurans, fund
-- SELECT c_id to left join property
-- SELECT c_name and c_id_card to OUTPUT
-- SELECT every table's amount and property quantity from each table and left join amount table with client table using pro_pif_id and pro type
    SELECT client.c_id, client.c_name, client.c_id_card, finances_product.p_amount, insurance.i_amount, fund.f_amount, property.pro_quantity
    FROM client
    LEFT JOIN property ON client.c_id=property.pro_c_id
    LEFT JOIN finances_product ON property.pro_pif_id=finances_product.p_id AND pro_type=1
    LEFT JOIN insurance ON property.pro_pif_id=insurance.i_id AND pro_type=2
    LEFT JOIN fund ON property.pro_pif_id=fund.f_id AND pro_type=3
    )AS union_table
GROUP BY c_id
ORDER BY total_amount DESC;
```

2.3.13 客户总资产

本关任务：查询客户在本行的总资产。

创造了两个新 TABLE 分别是 bank_table 和 property_table。再对 client 和两个 table 进行左连接。

```
SELECT c_id, c_name, IFNULL(bank_balance,0.0)+IFNULL(property_balance,0.0) AS total_property
FROM client
-- get bank balance; 储蓄卡is balance, 信用卡is liability; so need to minus liability
LEFT JOIN (SELECT b_c_id, sum(if(b_type='储蓄卡',b_balance,0.0-b_balance))AS bank_balance
FROM bank_card
GROUP BY b_c_id)bank_table ON c_id=b_c_id

-- get property balance; property balance=property_amount++
-- property_amount=property quantity*(finance property amount + insurance amount + fund amount +property income)
-- first sub table to get property_amount++
LEFT JOIN
    (
    SELECT pro_c_id, sum(property_amount) AS property_balance
-- second sub table to get each property amount
    FROM(
        SELECT DISTINCT pro_id, pro_c_id, pro_quantity*if(pro_type=1, p_amount, if(pro_type=2, i_amount, f_amount))+pro_income AS property_amount
-- from those table to get each amount where pro_type and pro_pif_id=each_id is true
        FROM property, fund, insurance, finances_product
        WHERE IF(pro_type=1, pro_pif_id=p_id, true)
            AND IF(pro_type=2, pro_pif_id=i_id, true)
            AND IF(pro_type=3, pro_pif_id=f_id, true)
        )property_type_table
        GROUP BY pro_c_id
    ) property_table ON c_id=pro_c_id;
```

2.3.14 第 N 高问题

该关卡任务已完成，实施情况本报告略过。

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两种方式的排名次。

以降序的基金投资收益和为关键字，直接调用 MySQL 库函数即可：函数 rank() 是名次不连续的排名，函数 dense_rank() 是名次连续的排名。

```
-- (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 ;
```

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。

使用 group_concat 函数将分组中的 pro_pif_id 找到，使用 DISTINCT 子句用于在连接分组之前消除组中的重复值。

```
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 t1.c_id c_id1, t2.c_id c_id2
FROM pro t1, pro t2
WHERE t1.c_id<t2.c_id AND t1.f_id=t2.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。

使用 datediff 查询购买日期减去 2021 年 12 月 31 日找到当天是 2022 年的第几天（简称天数 1），然后使用 2 乘以购买日期的周数为休息日的天数（简称天数 2），再用天数 1 减去天数 2 就能得到工作日的天数。最后再使用几次子查询把所有信息找到就可以了。

```sql
SELECT t3.t AS pro_purchase_time, t3.amount AS total_amount
FROM(
    -- * and 计算同一 workday - rownum 值的行数，并将结果作为新的字段 cnt
    SELECT *, COUNT(*) over(partition by t2.workday-t2.rownum)cnt
    FROM(
        -- * and 对结果集按照 workday 的升序进行编号，并将编号结果作为新的字段 rownum。
        SELECT *, row_number() over(ORDER BY workday)rownum
        FROM(
            -- to get amount per day and workday
            -- return t, amount, workday
            SELECT pro_purchase_time t, sum(pro_quantity*f_amount)amount, @row:=datediff(pro_purchase_time,"2021-12-31")-2*week(pro_purchase_time)workday
            FROM property, fund, (SELECT @row)a
            WHERE pro_purchase_time LIKE "2022-02-%"
            AND pro_type=3 AND pro_pif_id=f_id
            GROUP BY pro_purchase_time
        )t1
        WHERE amount > 1000000
    )t2
)t3
WHERE t3.cnt>=3;
```

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。

在 TABLE t 内：把购买时间周数减去 5，因为二月是在第五周开始。使用 LIKE 函数找出 2022 年 2 月的信息和 pro_type=3 的一起作为条件找出所需信息；在外层查询中使用 SUM 函数和 IF 函数找到一周各天的有效天数。

```sql
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

熟练使用 SQL 语句进行更高难度的查询操作。本任务已完成所有关卡。

2.4.1 查询销售总额前三的理财产品

本关任务：查询销售总额前三的理财产品。。

在 table1 内：查询购买时间为 2010 年或 2011 年的理财产品的购买年份，产品 id，和产品总额；在 table2 内：查询购买年份和根据年份和总额降序的前三排名的产品排名，id 和总额。

```
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, p_id, SUM(p_amount * pro_quantity) AS sumamount
        FROM property, finances_product
        WHERE pro_pif_id=p_id AND pro_type=1 AND year(pro_purchase_time) IN (2010,2011)
        GROUP BY p_id, pyear
    )AS table1
)AS table2
WHERE rk<=3;
```

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

该关卡任务已完成，实施情况本报告略过。

2.4.4 查找相似的理财产品

该关卡任务已完成，实施情况本报告略过。

2.4.5 查询任意两个客户的相同理财产品数

该关卡任务已完成，实施情况本报告略过。

2.4.6 查找相似的理财客户

本关任务：查找相似的理财客户。

在 table2 内：查询 pro_type=1 的 pro_c_id 和 pro_pif_id（不重复）

在 table3 内：连接 table1 和 table2，查询条件为 pro_type=1 以及 table1 和 table2 的用户 id 不相等（只要对比不同客户的理财）

在 table4 内：查询每位客户(列名：pac)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)。

在最外层查询内：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。

```
SELECT * FROM(
    -- to get crank
    SELECT *, rank() over(partition by pac ORDER BY common DESC, pbc ASC) AS crank
    FROM(
        -- to get pac, pbc and common
        SELECT DISTINCT table1.pro_c_id AS pac, table2.pro_c_id AS pbc, COUNT(table2.pro_c_id) AS common
        FROM property AS table1 JOIN(
            -- to get pbc
            SELECT DISTINCT pro_c_id, pro_pif_id
            FROM property
            WHERE pro_type=1
        )AS table2
        ON table2.pro_pif_id = table1.pro_pif_id
        WHERE pro_type=1 AND table2.pro_c_id != table1.pro_c_id
        GROUP BY table1.pro_c_id, table2.pro_c_id
    )AS table3
)AS table4
WHERE crank <=2;
```

2.5 数据的插入、修改与删除(Insert,Update,Delete)

熟练掌握数据的插入、修改和删除操作。本任务已完成所有关卡。

2.5.1 插入多条完整的客户信息

本关任务：向客户表 client 插入数据。

```
insert into client values
    (1,'林惠雯', "960323053@qq.com", "411014196712130323", "15609032348", "Mop5UPkl"),
    (2, "吴婉瑜", "1613230826@gmail.com", "420152196802131323", "17605132307", "QUTPhxgVNlXtMxN"),
    (3, "蔡贞仪", "252323341@foxmail.com", "160347199005222323", "17763232321", "Bwe3gyhEErJ7");
```

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。

```
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。

```
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。

```
DELETE FROM client WHERE NOT EXISTS (SELECT * FROM bank_card WHERE client.c_id=bank_card.b_c_id);
```

2.5.5 冻结客户资产

本关任务：冻结客户的投资资产。

```
UPDATE property
SET pro_status="冻结"
WHERE EXISTS(SELECT * FROM client WHERE c_phone="13686431238" AND c_id=pro_c_id);
```

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。

```
UPDATE property, client SET property.pro_id_card=client.c_id_card
WHERE property.pro_c_id=client.c_id;
```

2.6 视图

熟练使用 SQL 语对视图进行创建，修改，查询等操作。本任务已完成所有关卡。

2.6.1 创建所有保险资产的详细记录视图

本关任务：创建所有保险资产的详细记录视图。

```
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, insurance, property
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

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。

```
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

熟练使用 SQL 语句进行存储和事务的相关操作。本任务已完成 2.7.1 关卡。

2.7.1 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。

```
drop procedure if exists sp_fibonacci;
delimiter $$
create procedure sp_fibonacci(in m int)
begin
######## 请补充代码完成存储过程体 ########
    set m=m-1;
    with recursive cte (id, cur, pre) AS(
        SELECT 0,0,0
        UNION ALL
        SELECT id+1, if(id<2,1,cur+pre), cur
        FROM cte
        WHERE id<m
    )
    SELECT id n, cur fibn
    FROM cte;

end $$

delimiter ;
```

2.8 触发器

熟练使用 SQL 语句创建与删除触发器。本任务已完成所有关卡。

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，以实现以下完整性业务规则：

如果 pro_type = 1，则 pro_pif_id 只能引用 finances_product 表的 p_id；
如果 pro_type = 2，则 pro_pif_id 只能引用 insurance 表的 i_id；
如果 pro_type = 3，则 pro_pif_id 只能引用 fund 表的 f_id；
pro_type 不接受(1,2,3)以外的值。
各投资品种一经销售，不会再改变；
也不需考虑 finances_product，insurance，fund 的业务规则(一经销售的理财、保险和基金产品信息会永久保存，不会被删除或修改，即使不再销售该类产品)。

```
use finance1;
drop trigger if exists before_property_inserted;
-- 请在适当的地方补充代码，完成任务要求：
delimiter $$
CREATE TRIGGER before_property_inserted BEFORE INSERT ON property
FOR EACH ROW
BEGIN
    DECLARE new_pro_type int default new.pro_type;
    DECLARE id int default new.pro_pif_id;
    DECLARE msg varchar(50);
    IF new_pro_type = 1
    THEN
        IF id NOT IN(SELECT p_id FROM finances_product)
        THEN SET msg=concat("finances product #", id, " not found!");
        END IF;
    ELSEIF new_pro_type = 2
    THEN
        IF id NOT IN(SELECT i_id FROM insurance)
        THEN SET msg=concat("insurance #", id, " not found!");
        END IF;
    ELSEIF new_pro_type = 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 ", new_pro_type, " is illegal!");
    END IF;

    IF msg IS NOT NULL
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
    END IF;
END$$
delimiter ;
```

2.9 用户自定义函数

熟练使用 SQL 语句完成用户自定义函数，本任务已完成所有关卡。

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。

```
use finance1;
set global log_bin_trust_function_creators=1;
drop function IF EXISTS get_deposit;
delimiter $$
create function get_deposit(client_id int)
returns numeric(10,2)
begin
    return(
        SELECT SUm(b_balance)
        FROM bank_card
        WHERE b_type = '储蓄卡'
        GROUP BY b_c_id
        HAVING b_c_id=client_id
    );
end$$
delimiter ;

select *
from(
    SELECT c_id_card, c_name, get_deposit(c_id) AS total_deposit
    FROM client
)a
where total_deposit >= 1000000
order by total_deposit DESC
```

2.10. 安全性控制

熟练使用 SQL 语句进行用户，角色，权限等进行操作。本任务已完成所有关卡。

2.10.1 用户和权限

本关任务：在金融应用场景数据库环境中，创建用户，并给用户授予指定的权限

```
#(1) 创建用户tom和jerry，初始密码均为'123456';
CREATE USER tom IDENTIFIED BY '123456',
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 TABLE 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 TABLE bank_card
TO jerry;
#(4) 收回用户Cindy查询银行卡信息的权限。
REVOKE SELECT
ON TABLE bank_card
FROM Cindy;
```

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。

```
# (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 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_number, b_c_id, 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

熟练对并发控制与事物的隔离级别，本任务已完成 2.11.1~2.11.3 关卡。

2.11.1 并发控制与书屋的隔离级别

本关任务：设置事务的隔离级别

```
use testdb1;
# 设置事务的隔离级别为 read uncommitted
set session transaction isolation level read uncommitted;
-- 开启事务
start transaction;
insert into dept(name) values('运维部');
# 回滚事务:
rollback;
/* 结束 */
```

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。

要发生“读脏” 需要将隔离级别设置为 read uncommitted 。同时，确保事务 1 读航班余票发生在在事务 2 修改之后，事务 2 撤销发生在事务 1 读取之后，据此设置事务休眠时间，即可发生“读脏”。

事务 1：

```
use testdb1;
## 请设置适当的事务隔离级别
set session transaction isolation level read uncommitted;

start transaction;

-- 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);
select tickets from ticket where flight_no = 'CA8213';
commit;
```

事务 2：

```
use testdb1;
## 请设置适当的事务隔离级别
set session transaction isolation level read uncommitted;
start transaction;
-- 时刻1 - 事务2修改航班余票
update ticket set tickets = tickets - 1 where flight_no = 'CA8213';

-- 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);

rollback;
```

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。

2.12 备份+日志：介质故障与数据库恢复

熟练掌握备份和日志的操作。本任务已完成所有关卡。

2.12.1 备份与恢复

本关任务：备份数据库，然后再恢复它。

使用 mysqldump 指令将服务器上的数据库 residents 备份至文件 residents_bak.sql 中：

```
mysqldump -h127.0.0.1 -uroot --databases residents > 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。

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在 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;
```

2.13 数据库设计与实现

将已建好的概念模型，变成 MySQL 物理实现。本任务已完成 2.13.1 关卡。

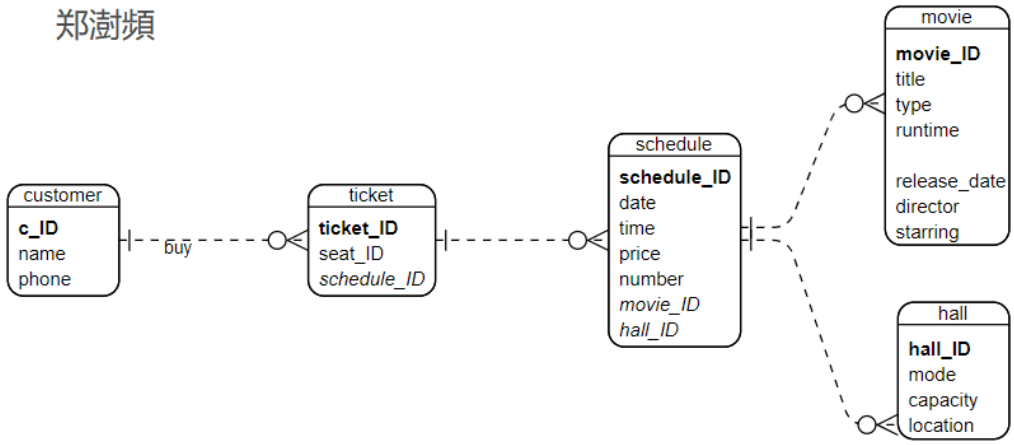
2.12.1 从概念模型到 MySQL 实现

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

本关任务：根据应用场景业务需求描述，完成 ER 图，并转换成关系模式。提交设计文档

ER 图：



关系模式：

以下给出关系模式:
movie(movie_ID, title, type, runtime, release_date, director, starring), primary key:(movie_ID);

customer(c_ID, name, type, phone), primary key:(c_ID);

hall(hall_ID, mode, capacity, location), primary key:(hall_ID);

schedule(schedule_ID, date, time, price, number, movie_ID, hall_ID), primary key:(schedule_ID), foreign key(movie_ID, hall_ID);

ticket(ticket_ID, seat_num, schedule_ID), primary key(ticket_ID), foreign key(schedule_ID);

2.13.4 制约因素分析与设计

在现实中，在设计和构建数据库的过程中需要考虑很多制约因素。

以任务 2.13.1 中的例子展开，系统需要考虑到团体旅客的存在。一个旅客可能需要同时够买多张相同航班的机票（乘坐人不同）。所以可以考虑在系统中建立两个相关实体：乘客和购买人。这样，每个机票实例可以与一个乘客和一个购买人相关联，允许旅客和购买人是同一人的情况。

另外，系统也需要考虑到权限问题。系统需要区分普通用户和管理用户的权限。这可以通过为用户实体添加一个"权限"属性或者将用户分为不同的角色来实现。可以将用户的权限属性设置为普通用户或管理用户，并在系统中定义相应的访问和操作权限。

2.13.5 工程师责任及其分析

在社会方面，工程师具有基于他们的专业知识和背景进行合理分析的能力。他们应该能够评估专业工程实践和解决复杂工程问题的方案对社会、健康、安全、法律以及文化的影响，并且理解他们应该承担的责任。这意味着工程师需要考虑他们的工程实践如何影响社会的各个方面，包括其对人们的健康和安全的影响，以及与法律和文化价值观的一致性。

在安全方面，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都关心的重要问题。工程师在设计和实施工程解决方案时，应该考虑到潜在的威胁和漏洞，并采取适当的措施来减轻风险。这包括对系统进行安全性评估、采取安全措施和使用安全性最佳实践。

在科学发展方面，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。这包括设计实验来测试假设、收集和分析数据，并通过综合信息得出合理有效的结论。工程师需要了解并运用科学方法，以确保他们的研究和工程实践是可靠、准确和可重复的。

综上所述，工程师在社会方面应具备评估工程实践对社会各个方面的影响的能力，并承担相应的责任；在安全方面应重视系统的安全性，并采取适当的措施来减轻风险；在科学发展方面应运用科学原理和方法进行研究，以获得可靠有效的结论。这些能力和职责对于工程师来说都是非常重要的。

2.14 数据库应用开发(JAVA 篇)

熟练使用嵌入式 SQL，使用 JAVA。本任务已完成所有关卡。本任务报告中因全代码过长所以均以部分代码形式展示。

2.14.1 JDBC 体系结构和简单的查询

本关任务：查询 client 表中邮箱非空的客户信息，列出客户姓名，邮箱和电话。

使用 Java 的 Class.forName() 方法，将驱动程序的类文件动态加载到内存中，并将其自动注册。加载驱动程序后，使用 DriverManager.getConnection(String url, String user, String password) 方法建立连接。在使用 Statement 对象执行 SQL 语句之前，需要使用 Connection 对象的 createStatement()方法创建 Statement 的一个实例。

以下是连接部分代码：

```
static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://127.0.0.1:3306/finance?useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC";
static final String USER = "root";
static final String PASS = "123123";
```

以下是连接设定好的信息，查询所需信息，处理输出的代码：

```
try {
    //编写部分
    Class.forName(JDBC_DRIVER);
    connection = DriverManager.getConnection(DB_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.println("姓名\t邮箱\t\t\t\t电话");
    while (resultSet.next()) {
        System.out.print(resultSet.getString("c_name") + "\t");
        System.out.print(resultSet.getString("c_mail") + "\t\t");
        System.out.println(resultSet.getString("c_phone"));
    }
```

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。

```
//申明下文中的resultSet, statement
Statement statement = null;
ResultSet resultSet = null;
```

从 SQL 中查询客户邮件和密码，对比成功后显示“登入成功”，失败则显示“用

户名或密码错误！”

```
// 补充实现代码:
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。设置好格式后开始接收数据并按顺序分到各个属性中。

```
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();
}
return 0;
```

2.14.4 银行卡销户

本关任务：编写一个能删除指定客户编号的指定银行卡号的方法。

把得到的用户 ID 和银行卡号嵌入到写好的 SQL 语句中，执行删除操作。

```
String sql = "delete from bank_card where b_c_id = ? and b_number = ?";
try {
    PreparedStatement ps = connection.prepareStatement(sql);
    ps.setInt(1, b_c_id);
    ps.setString(2, b_number);
    return ps.executeUpdate();
} catch (SQLException e) {
    e.printStackTrace();
}
return 0;
```

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。

查询用户邮箱是否存在，接收到新密码后，利用 SQL 语句进行修改操作。

```
String sql = "select * from client where c_mail = ?";
try {
    PreparedStatement ps = connection.prepareStatement(sql);
    ps.setString(1, mail);
    ResultSet res = ps.executeQuery();
    if (res.next()) {
        if (password.equals(res.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

本关任务：编写一个银行卡转账的方法。

从 SQL 中查询对应银行卡号的用户信息，设置条件如卡类型为信用卡或卡余额小于转账数额的情况下，返回 false，无法操作。如符合转账条件，则进行转账后，再对 SQL 进行修改操作，把信息修改成最新的余额状况。

```
    try {
    String sql = "select * from bank_card where b_number = ?";
    PreparedStatement ps = connection.prepareStatement(sql);
    ps.setString(1, sourceCard);
    ResultSet res = ps.executeQuery();
    if (!res.next() || res.getString("b_type").equals("信用卡") || res.getDouble("b_balance") < amount)
        return false;
    ps = connection.prepareStatement(sql);
    ps.setString(1, destCard);
    res = ps.executeQuery();
    if (!res.next())
        return false;
    double rcv_amount = res.getString("b_type").equals("信用卡") ? -amount : amount;
    sql = "update bank_card set b_balance = b_balance + ? where b_number = ?";
    ps = connection.prepareStatement(sql);
    ps.setDouble(1, -amount);
    ps.setString(2, sourceCard);
    ps.executeUpdate();
    ps = connection.prepareStatement(sql);
    ps.setDouble(1, rcv_amount);
    ps.setString(2, destCard);
    ps.executeUpdate();
    return true;
} catch (SQLException e) {
    e.printStackTrace();
}
return false;
```

2.14.7 把稀疏表格转为键值对存储

本关任务：将一个稀疏的表中有保存数据的列值，以键值对(列名，列值)的形式转存到另一个表中，这样可以直接丢失没有值列。

输入学生学号，科目名称，分数到 SC 表。

```java
public static void insertSC(Connection con, int sno, String col_name, int col_value) {
    try {
        String sql = "insert into sc values (?, ?, ?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, sno);
        ps.setString(2, col_name);
        ps.setInt(3, col_value);
        ps.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```java
public static void main(String[] args) throws Exception {
    Class.forName(JDBC_DRIVER);
    Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
    String[] subject = {"chinese", "math", "english", "physics", "chemistry", "biology", "history", "geography", "politics"};
    try {
        ResultSet res = con.createStatement().executeQuery("select * from entrance_exam");
        while (res.next()) {
            int sno = res.getInt("sno"), score;
            for (String sub : subject) {
                score = res.getInt(sub);
                if (!res.wasNull())
                    insertSC(con, sno, sub, score);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

3 课程总结

本次数据库课程实践的总体任务是学习和应用数据库的概念、原理、设计和应用。在完成这一任务的过程中，我学习了很多：

我学习了实体-关系模型和关系模型的理论知识，并应用这些知识进行了数据库的建模和设计。我使用了 ER 图工具，绘制了数据库的实体关系图，并根据实际需求设计了合适的关系模式和数据表结构。

我学习了 SQL 语言的基本语法和常见操作，并通过实践练习掌握了 SQL 查询的技巧。我能够独立编写和执行 SQL 查询语句，实现数据的插入、更新、删除和查询操作。

我了解了数据库管理的基本概念和技术，并学习了如何进行数据库备份和恢复、安全性管理和性能调优。我掌握了常见的数据库管理工具和技术，能够有效地管理和优化数据库系统。

我学习了使用编程语言与数据库进行交互，编写数据库应用程序和 Web 应用程序的方法。我通过实践项目，开发了一些简单的数据库应用程序，实现了数据的增删改查功能。

在本次课程实践中，我收获了很多。首先，我对数据库的概念和原理有了更深入的理解，能够更好地设计和管理数据库系统。其次，我提高了 SQL 查询的能力，能够灵活运用 SQL 语言解决实际问题。此外，我还学会了使用编程语言开发与数据库交互的应用程序，提升了自己的实际开发能力。

然而，在本次课程实践中，我也发现了一些可以改进和完善的地方。首先，我希望能够在实践项目中应用更复杂的数据库设计和查询技巧，以挑战自己的能力并提高实际应用水平。其次，我认识到数据库安全和隐私保护的重要性，希望在未来的实践中更加注重这方面的学习和实践。

总而言之，通过本次数据库课程实践，我获得了宝贵的知识和经验，并提升了自己在数据库领域的能力。我相信这些学习将对我的未来职业发展产生积极的影响，并希望能够在实践中不断改进和完善自己的数据库技能。