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 2301 班
学　　号：	U202315387
姓　　名：	施诚
指导教师：	谢美意

分数	
教师签名	

2025 年　6 月 29　日

教师评分页

子目标	子目标评分
1	
2	
3	
4	
5	

总分	

目录

1 课程任务概述

本实践课程基于头歌平台，以 MySQL 8.0.28 和 Java 1.8 为环境，注重理论与实践结合，涵盖数据库对象管理、数据处理、安全控制、设计开发等内容。 总体任务要求，任务分解为 16 个实训：

1. 基础操作：包括数据库、表等对象的创建与修改，数据的增删改查。

2. 高级功能：视图、存储过程、触发器等的使用，安全性与并发控制，备份恢复。

3. 设计开发：数据库设计与实现，Java 应用开发。

4. 存储管理：缓冲池管理器等的实现。

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 1～11，13～15 实训任务，下面将重点针对其中的一些任务阐述其完成过程中的具体工作。

2.1 MYSQL-数据库、表与完整性约束的定义

本小节的六个小关卡主要围绕如何正确使用 SQL 语法，如何正确创建数据库和表，以及如何正确添加主码约束，外码约束，CHECK 约束，UNIQUE 约束，DEFAULT 约束，本小节的六个关卡全部做完。

2.1.1 创建数据库

本关全部完成，完成过程略。

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。由于在测试过程中会多次创建表，故先 DROP 表，再创建表，见下图 2. 1.1 所示。

```
DROP DATABASE TestDb;
CREATE DATABASE TestDb;
use TestDb;

CREATE TABLE t_emp
(
    id INT PRIMARY KEY,
    name VARCHAR(32),
    deptId INT,
    salary FLOAT
);
```

图 2. 1.1 创建表和主码约束

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）。

思路方法：首先要创建 dept 部门表，再根据 dept 表来创建 staff 员工表，代码如下图 2. 1.2 所示：

```sql
DROP DATABASE MyDb;
CREATE DATABASE MyDb;
USE MyDb;

CREATE TABLE dept (
  deptNo INT PRIMARY KEY,
  deptName VARCHAR(32)
);

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)
);
```

图 2. 1.2 创建外码约束

2.1.4 CHECK 约束

本关任务：为表创建 CHECK 约束

思路方法: 根据题目要求创建 products 表并添加 CHECK 约束, 代码如下图 2. 1.3 所示：

```sql
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)
);
```

图 2.1.3 CHECK 约束

2.1.5 DEFAULT 约束

本关任务：创建默认值约束(Default 约束),为字段指定默认值。

思路方法：为 hr 表创建 DEFAULT 约束并注意 NOT NULL 约束，代码如下图 2. 1.4 所示：

```sql
CREATE TABLE hr (
    id CHAR(10) PRIMARY KEY,
    name VARCHAR(32) NOT NULL,
    mz CHAR(16) DEFAULT '汉族'
);
```

图 2.1.4 DEFAULT 约束

2.1.6 UNIQUE 约束

本关任务：创建唯一性约束(UNIQUE 约束),以保证字段取值的唯一性。

思路方法：创建 s 表并设置 UNIQUE 约束，代码略。

2.2 表结构与完整性约束的修改(ALTER)

本小节的四个小关卡主要围绕如何正确使用 ALTER 语法，如何正确修改表名，添加和删除字段，修改字段以及添加、删除和修改约束。本小节的四个关卡全部做完。

2.2.1 修改表名

本关任务：修改表的名称。

思路方法：使用 ALTER 关键字将表名 your_name 改为 my_name,代码略。

2.2.2 添加和删除字段

本关任务：为表添加和删除字段。

思路方法：格式为 ALTER TABLE [NAME] ADD/DROP,按照格式即可完成本关，代码略。

2.2.3 修改字段

本关任务：修改字段

思路方法：和上述一样，略。

2.2.4 添加，删除和修改约束

本关任务：添加、删除与修改约束

思路方法：结合上一小节的添加，删除各种约束的知识，可以完成本关，代码截图如下图 2.2.1 所示：

```
-- (1) 为表Staff添加主码
ALTER TABLE Staff
ADD CONSTRAINT PK_Staff PRIMARY KEY (staffNo);

-- (2) Dept.mgrStaffNo是外码，对应Staff.staffNo，添加外码约束
ALTER TABLE Dept
ADD CONSTRAINT FK_Dept_mgrStaffNo FOREIGN KEY (mgrStaffNo) REFERENCES
Staff(staffNo);

-- (3) Staff.dept是外码，对应Dept.deptNo，添加外码约束
ALTER TABLE Staff
ADD CONSTRAINT FK_Staff_dept FOREIGN KEY (dept) REFERENCES Dept(deptNo);

-- (4) 为表Staff添加check约束，gender只能是F或M
ALTER TABLE Staff
ADD CONSTRAINT CK_Staff_gender CHECK (gender IN ('F', 'M'));

-- (5) 为表Dept添加unique约束，deptName不能重复
ALTER TABLE Dept
ADD CONSTRAINT UN_Dept_deptName UNIQUE (deptName);
```

图 2.2.1 添加，删除和修改约束

2.3 基于金融应用的数据查询（SELECT）

本小节围绕 SELECT 语句在不同场景的应用展开，包含 19 个关卡，涉及金融场景客户信息查询、空值处理、复杂条件查询、金额范围筛选、收益统计、排名、高峰期分析等，还实现了日历表格式展示等复杂查询操作。本小节的 19 个关卡均已完成，挑选其中具有代表性的关卡介绍。

2.3.1 查询邮箱为 NULL 的客户

本关任务：查询客户表(client)中没有填写邮箱的客户的编号、名称、身份证号、手机号。

思路方法：使用 SELECT 语句查询邮箱为 NULL 的客户，需要注意的是，WHERE 语句应当写成“WHERE c_mail IS NULL”而不是“WHERE c_mail = NULL”，因为 NULL 代表未知，并非一个具体数据，=用于比较具体值，无法准确判断 NULL 状态。代码如下图 2.1 所示：

```
SELECT c_id,c_name,c_id_card,c_phone
FROM client
WHERE c_mail is NULL;
```

图 2.3.1 查询邮箱为 NULL 的客户

2.3.2 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。

思路方法：本关需要使用嵌套查询以及多条件查询的方法并最后要注意排序，需要注意的是嵌套查询的语法和格式，代码如下图 2. 3.2 所示：

```
select c_name, c_mail, c_phone
from client
where exists(
        select *
        from property
        where
            client.c_id = pro_c_id
            and pro_type = '3'
    )
    and exists(
        select *
        from property
        where
            client.c_id = pro_c_id
            and pro_type = '2'
    )
order by c_id;
```

图 2.3.2 既买了保险又买了基金的客户

2.3.3 每分金额在 30000～50000 之间的产品

本关任务：查询每份金额在 30000～50000 之间的理财产品，并对结果进行排序。

思路方法：本关需要注意的是 BETWEEN AND 以及正序排序，倒序排序的语法。代码如下图 2.3 .3 所示：

```
SELECT
    p_id,
    p_amount,
    p_year
FROM finances_product
WHERE p_amount BETWEEN 30000 AND 50000
Order BY p_amount ASC ,p_year DESC;
```

图 2.3.3 每分金额在 30000～50000 之间的产品

2.3.4 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。

思路方法：本关需要注意的是分组统计与 COUNT()函数，ALL 关键词或者 MAX()函数。代码如下图 2.3.4 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
    );
```

图 2.3.4 商品收益的众数

2.3.5 未购买任何产品的武汉居民

本关任务：查询未购买任何理财产品的武汉居民的信息。

思路方法：本关需要注意的是 LIKE 的用法，其用法为 <列名> LIKE ‘<匹配串>’[ESCAPE ‘<转义字符>’]，通配符%代表任意长度的字符串；通配符_代表任意单个字符，代码如下图 2. 3.5 所示：

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

图 2.3.5 未购买任何产品的武汉居民

2.3.6 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。

思路方法：该 SQL 语句通过连接 client 和 property 表，筛选出可用资产的客户，按客户分组计算总收益，降序排列后取前三名，输出名称、身份证号和总收益。代码如下图 2. 3.6 所示：

```sql
SELECT c_name,c_id_card,sum(pro_income) as total_income
FROM client,property
WHERE
    client.c_id = property.pro_c_id and
    pro_status = "可用"
GROUP BY c_id
ORDER BY total_income DESC
LIMIT 3;
```

图 2.3.6 投资总收益前三名的客户

2.3.7 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。

思路方法：该代码使用`RankedData`，通过`DENSE_RANK()`窗口函数对`insurance`表中保险金额`i_amount`降序排名，保留排名为 4 的保险产品编号`i_id`和金额，最后按`i_id`升序输出。代码如下图 2.3.7 所示：

```sql
WITH RankedData AS (
    SELECT
        i_id,
        i_amount,
        DENSE_RANK() OVER (ORDER BY i_amount DESC) AS rank_num
    FROM insurance
)
SELECT i_id, i_amount
FROM RankedData
WHERE rank_num = 4
ORDER BY i_id ASC;
```

图 2.3.7 第 N 高问题

2.3.8 基金收益的两种方式

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。

思路方法：两种方式均从`property`表筛选`pro_type=3`的基金投资记录，按客户编号`pro_c_id`分组计算总收益`total_revenue`，前者用`RANK()`实现不连续排名（相同收益占同一名次，后续名次跳跃），后者用`DENSE_RANK()`实现连续排名（相同收益占同一名次，后续名次连续），均按总收益降序和客户编号升序输出。代码如下图 2.3.8 所示：

```sql
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;

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 2.3.8 基金收益的两种方式

2.3.9 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。

思路方法：该 SQL 通过多层子查询实现高峰期查询：首先连接`property`和`fund`表，筛选 2022 年 2 月基金记录并计算每日总金额`amount`，通过`datediff`与`week`函数组合生成连续工作日标识`workday`；接着筛选`amount>100 万`的日期，用`row_number`生成行号`rownum`；然后通过`workday-rownum`分组计数`cnt`，识别连续工作日；最后筛选`cnt≥3`的日期，输出高峰期日期和总金额。代码如下图 2.3.9 所示：

```
select
daycnt.pro_purchase_time as pro_purchase_time,
daycnt.amount as total_amount
from (
select*,count(*) over(
partition by orderday.workday - orderday.rownum
) cnt
from (select*,row_number()
over(order by workday) 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;
```

图 2.3.9 购买基金的高峰期

2.4 数据查询（SELECT2）

本小节在上一小节的基础上进行一些更复杂的查找方式，本小节的完成题目为 1～3，5 题。

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。

思路方法：该 SQL 通过 CTE 临时表 yearly_ranked_products，先将 property 与 finances_product 表连接，筛选出 2010 至 2011 年间 pro_type 为 1 的理财产品记录，按年份和产品 ID 分组，计算各产品销售总额（pro_quantity 乘以 p_amount），再用 RANK()窗口函数按年份分区对总额降序排名，最后从临时表中筛选出各年份排名前三的产品，按年份和排名排序输出。代码如下图 2.4.1 所示：

```sql
WITH yearly_ranked_products AS (
    SELECT
        YEAR(pro_purchase_time) AS pyear,
        pro_pif_id AS p_id,
        SUM(property.pro_quantity * p_amount) AS sumamount,
        RANK() OVER (
            PARTITION BY YEAR(pro_purchase_time)
            ORDER BY SUM(property.pro_quantity * p_amount) DESC
        ) AS rk
    FROM
        property
    JOIN
        finances_product ON property.pro_pif_id = finances_product.p_id
    WHERE
        pro_type = 1
        AND pro_purchase_time BETWEEN '2010-01-01' AND '2011-12-31'
    GROUP BY
        YEAR(pro_purchase_time), pro_pif_id
)
SELECT pyear, rk, p_id, sumamount
FROM yearly_ranked_products
WHERE rk <= 3
ORDER BY pyear, rk;
```

图 2.4.1 查询销售总金额前三的理财产品

2.4.2 投资积极且偏好理财类产品的客户

本关任务：投资积极且偏好理财类产品的客户。

思路方法：先分别统计每个客户购买理财产品（pro_type=1）和基金（pro_type=3）的不同产品数量，再筛选出理财产品数量多于基金且购买至少 3 种不同产品的客户，最后按客户 ID 排序，以此确定投资积极且偏好理财类产品的客户，代码略。

2.4.3 查询购买了所有畅销理财产品的客户

本关任务：查询购买了所有畅销理财产品的客户。

思路方法：首先通过内层子查询确定“畅销理财产品”（`pro_type=1`且被超过 2 个客户购买的`pro_pif_id`），然后对于每个客户`x`，检查是否存在未购买的畅销产品`y`，若不存在（即客户购买了所有畅销产品），则返回其`pro_c_id`，最后按客户 ID 升序排列。代码略。

2.4.4 查询任意两个客户的相同理财产品数

本关任务：查询任意两个客户的相同理财产品数。

思路方法：该 SQL 语句通过自连接`property`表为`A`和`B`，筛选出`pro_type=1`的理财产品记录，匹配不同客户（`A.pro_c_id != B.pro_c_id`）且购买相同产品（`A.pro_pif_id = B.pro_pif_id`）的情况，按客户对分组后统计相同产品数量，筛选出至少有 2 个相同产品的客户对，最后按客户 ID 升序输出。代码略。

2.5 数据的插入、删除和修改

本小节围绕对于表中数据的插入删除和修改等操作展开，整体难度不大，六个小题均已完成。

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。

思路方法：略。

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。

思路方法：本关需要注意的点是在插入不完整数据的时候需要将对应的属性列设置为 NULL。相关代码如下图 2.5.1 所示：

```
INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password)
VALUES
(33,'蔡依婷',NULL,'350972199204227621','18820762130','MKwEuc1sc6');
```

图 2.5.1 投资积极且偏好理财类产品的客户

2.5.3 批量插入数据

本关任务：向客户表 client 批量插入数据。

思路方法：将 INSERT 语句中的 VALUE 改成 SELECT*即可，相关代码如下图 2.5.2 所示：

```
INSERT INTO client
SELECT *
FROM new_client
```

图 2.5.2 批量插入数据

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。

思路方法：本关需要注意的是 MySQL 的 delete 语句中的 FROM 不能省略，代码略。

2.5.5 冻结客户资产

本关任务：冻结客户的投资资产。

思路方法：简单地使用嵌套查询将客户的资金状态设计为‘冻结’即可，代码略。

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。

思路方法：简单的 UPDATE SET 语句，代码略。

2.6 视图

SQL 中的视图（View）是一个虚拟表，它是基于一个或多个表的查询结果创建的。视图本身不存储数据，而是保存了一条 SQL 查询语句，查询时会动态执行。本小节两关均已做完。

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。

思路方法：使用创建视图的语法：CREATE VIEW v_insurance_detail AS...具体代码略。

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。

思路方法：视图的查询和基本表的查询并无区别，代码如下图 2.6. 1 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;
```

图 2.6. 1 基于视图的查询

2.7 存储过程与事务

本节的 3 个关卡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。

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。

思路方法：使用创建视图的语法：CREATE VIEW v_insurance_detail AS...具体代码略。

2.7.2 使用游标的存储过程

本关任务：使用游标编程存储过程为医院的某科室排夜班值班表。

思路方法：这段存储过程的设计思路是通过游标遍历员工名单，实现轮换排班，并结合日期特性动态调整排班逻辑。它将医生（含主任）和护士分别用两个游标依次取出，每天安排一名医生和两名护士值班；如果游标到头则重新打开，实现循环排班。同时，利用星期信息设置“主任连值星期一”的规则，通过保存上一周末值班主任的信息，在周一继续安排其值班，保障排班公平与连续性。具体代码略。

2.7.3 使用事务的存储过程

本关任务：编写实现转账功能的存储过程。

思路方法：该存储过程通过关闭自动提交并启动事务，先尝试从付款方储蓄卡扣款、向收款方账户加款，再通过条件判断验证卡号和余额的合法性，若有任一异常则回滚，否则提交操作，确保转账过程具备原子性和数据一致性，从而避免因中途出错导致资金异常。同时通过输出参数 return_code 返回转账结果，便于外部系统判断操作是否成功。具体代码如下图 2.7.1 使用事务的存储过程图 2.7.1 所示：

```
create procedure sp_transfer(IN applicant_id int,
                             IN source_card_id char(30),
                             IN receiver_id int,
                             IN dest_card_id char(30),
                             IN amount numeric(10,2),
                             OUT return_code int)
as
begin
    update bank_card set b_balance = b_balance - amount where b_type = '储蓄卡'
and b_number = source_card_id and b_c_id = applicant_id;
    update bank_card set b_balance = b_balance + amount where b_type = '储蓄卡'
and
b_number = dest_card_id and b_c_id = receiver_id;
    update bank_card set b_balance = b_balance - amount where b_type = '信用卡'
and
b_number = dest_card_id and b_c_id = receiver_id;
    if not exists(select * from bank_card where b_type = '储蓄卡' and b_number =
source_card_id and b_c_id = applicant_id and b_balance >= 0) then
        return_code := 0;
        rollback;
    elseif not exists(select * from bank_card where b_number = dest_card_id and
b_c_id = receiver_id) then
        return_code := 0;
        rollback;
    else
        return_code := 1;
        commit;
    end if;
end;
```

图 2.7.1 使用事务的存储过程

2.8 触发器

2.8.1 为投资表 property 实现业务约束规则

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。

思路方法：这段代码定义了一个 BEFORE INSERT 类型的触发器，用于在向 property 表插入新记录前对数据合法性进行校验。它根据 pro_type 字段的值，判断关联的产品 ID 是否在对应的业务表中存在（如 finances_product、insurance、fund）。如果未找到对应记录，或 pro_type 不合法（不在 1、2、3 范围内），则通过 SIGNAL 语句抛出自定义错误，阻止插入操作。整体思路是通过触发器实现前置约束校验，从源头防止无效或错误的产品数据写入系统，确保数据一致性与业务准确性。具体代码如下图 2.8. 1 所示：

```
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 msg VARCHAR(50);
IF new.pro_type = 1 AND NOT EXISTS (SELECT * FROM finances_product WHERE p_id =
new.pro_pif_id) THEN
    SET msg = CONCAT("finances product #",new.pro_pif_id," not found!");
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
ELSEIF new.pro_type = 2 AND NOT EXISTS (SELECT * FROM insurance WHERE i_id = new.
pro_pif_id) THEN
    SET msg = CONCAT("insurance #",new.pro_pif_id," not found!");
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
ELSEIF new.pro_type = 3 AND NOT EXISTS (SELECT * FROM fund WHERE f_id = new.
pro_pif_id) THEN
    SET msg = CONCAT("fund #",new.pro_pif_id," not found!");
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
ELSEIF new.pro_type NOT IN (1,2,3) THEN
    SET msg = CONCAT("type ",new.pro_type," is illegal!");
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
END IF;

END$$

delimiter ;
```

图 2.8. 1 触发器

2.9 用户自定义函数

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。

思路方法：本关的核心思路是先通过自定义函数 get_deposit(client_id) 封装“按客户编号统计其所有储蓄卡余额总和”的逻辑，使其成为可复用的查询工具；然后在主查询中调用该函数，筛选出所有存款总额大于等于 100 万的客户，并输出其身份证号、姓名及存款总额，按金额降序排序。具体代码如下图 2.9. 1 图 2.9. 2 所示：

```
use finance1;
set global log_bin_trust_function_creators=1;
drop function IF EXISTS get_deposit;
/*
   用create function语句创建符合以下要求的函数:
   依据客户编号计算该客户所有储蓄卡的存款总额。
   函数名为: get_Records。函数的参数名可以自己命名:*/

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 ;
```

图 2.9. 1 用户创建函数

```
/*  应用该函数查询存款总额在100万以上的客户身份证号，姓名和存储总额(total_deposit),
    结果依存款总额从高到代排序  */
select
    c_id_card,
    c_name,
    get_deposit(c_id) as total_deposit
from client
where
    get_deposit(c_id) >= 1000000
order by total_deposit desc;
```

图 2.9. 2 函数应用

2.10 安全性控制

本节的两个关卡主要涉及数据库中用户，角色和权限的内容

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。

思路方法：这段代码通过一系列 SQL 语句实现对数据库用户的创建、授权与权限管理。首先使用 CREATE USER 创建两个用户 tom 和 jerry，并设置初始密码为 '123456'；接着使用 GRANT 语句分别赋予 tom 查询 client 表中客户姓名、邮箱和电话字段的权限，并允许其转授该权限（WITH GRANT OPTION），

以及赋予 jerry 修改 bank_card 表中余额字段的权限;最后用 REVOKE 语句收回用户 Cindy 对银行卡信息的查询权限。具体代码如下图 2.10.1 所示：

```
# 请填写语句，完成以下功能：
use finance;
#(1) 创建用户tom和jerry，初始密码均为'123456';
DROP USER IF EXISTS 'tom';
CREATE USER 'tom' IDENTIFIED BY '123456';
DROP USER IF EXISTS 'jerry';
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 finance.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 finance.bank_card TO 'jerry';
#(4) 收回用户Cindy查询银行卡信息的权限。
REVOKE SELECT ON finance.bank_card FROM 'Cindy';
```

图 2.10.1 用户和权限

2.10.2 用户，角色和权限

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。

思路方法：相比于前一关，本关新增了角色，只需要使用 GRANT 语句给角色赋予权限并将角色赋给具体的用户即可，代码略。

2.11 数据库设计与实现

本关的主要任务包括从概念模型到 SQL 实现、从需求分析到逻辑模型、建模工具的使用、制约因素分析与设计以及工程师责任及其分析。需将概念模型转换为 SQL 关系模式，依据需求绘制 ER 图并转为关系模式，使用 SQL 建模工具，分析约束条件，遵循设计原则考虑社会影响、安全性等。

2.11.1 从概念模型到 MySQL 实现

本关任务：将建好的模型转化为 SQL 的物理实现

思路方法：本关比较简单，主要是建立表格等等，代码略。

2.11.2 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。

思路方法：首先写好标准的 SQL 语句，再将标准 SQL 语句输出为 DBML 语句，然后使用 dbdiagram.io 网站工具画出 E-R 图，画出的 E-R 图如下图 2.11.1 所示：

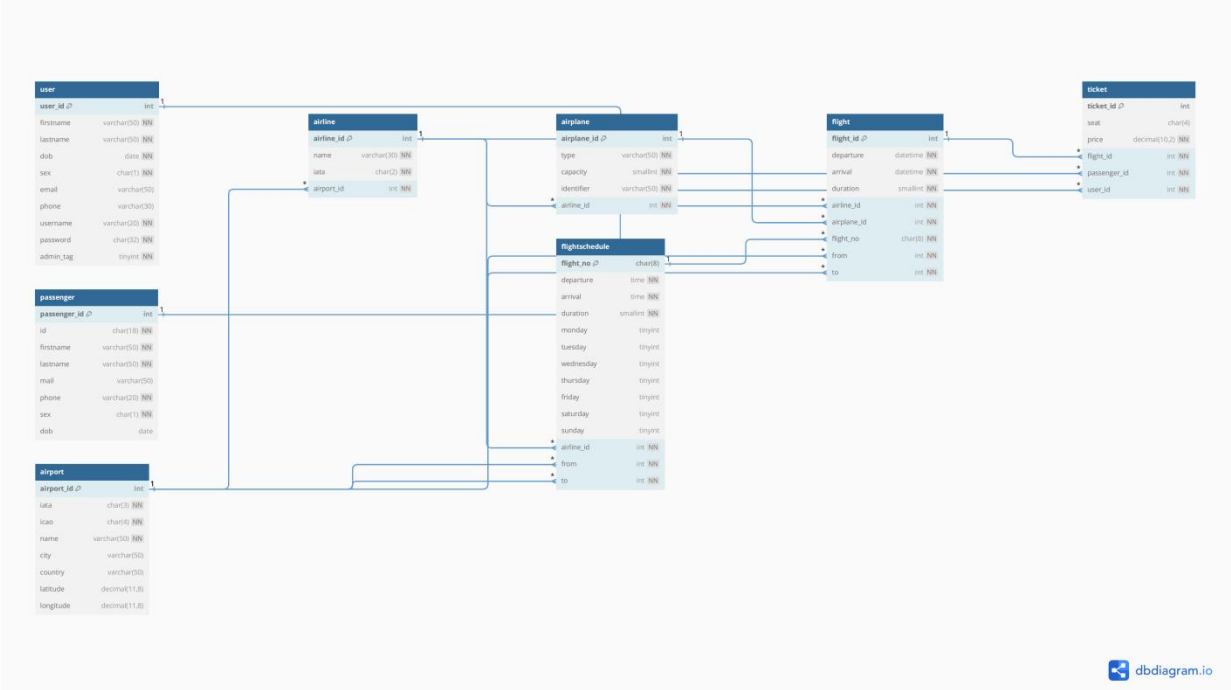

图 2.11.1 E-R 图

2.11.3 建模工具的使用

本关任务：使用 MySQL Workbench 的 forward engineering 功能，将建好的模型自动转换成 SQL 脚本。

思路方法：学会使用改功能即可，具体代码略。

2.11.4 制约因素分析与设计

在设计这个系统的时候，我们考虑了很多实际生活中的限制因素：

社会和用户需求：比如坐飞机的人可能多次订票，系统得能记录每个人的多次购票情况；买票的人可能是帮别人买的，所以要分开记录“购票人”和“乘机人”信息，这样出错时能清楚查到是谁操作的。

安全和权限：账号分了“普通用户”和“管理员”，普通用户只能订票，管理员才能管理整个系统，就像现实中不同岗位的人有不同职责，防止乱操作。

法律和隐私：收集身份证、电话这些信息时，系统会规定必须填哪些内容，不能随便泄露，就像平时签合同一样，得保护每个人的隐私。

使用便利性：虽然没特别设计文化相关的功能，但系统操作界面和流程尽量简单。

系统效率：通过优化数据存储方式（比如用 B+树索引），让查询和订票速度更快，减少大家等待的时间，节省资源。

2.11.5 工程师责任及其分析

在数据库系统原理实践中，任务解决方案与社会、安全、法律等因素紧密关联。例如，在安全性控制实验中，通过创建角色分配权限（如 client_manager 角色对客户表的操作权限），防止未授权访问，这直接关系到用户数据安全和隐私保护，符合数据安全法规要求。

工程师需承担多重社会责任：在社会层面，设计需考虑系统对金融行业稳定性的影响，如客户总资产查询功能确保数据准确，支撑金融决策；安全层面，通过触发器和权限控制防止数据泄露，守护用户信息安全；法律层面，遵循数据保护法规，如权限分配严格限制数据访问范围。此外，工程师还应兼顾技术先进性与社会适用性，确保技术应用符合伦理和法律规范，推动行业健康发展。

2.12 数据库应用开发(JAVA 篇)

本小节要求基于 JDBC 技术实现数据库与 Java 的交互，完成用户登录、新客户添加、银行卡销户、密码修改、转账等功能。任务需实现数据查询（如邮箱非空客户信息查询）和事务操作（如转账时设置自动提交、回滚机制），同时要处理异常并正确释放资源，还涉及将稀疏表格转化为键值对存储等数据处理需求。

2.12.1 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。

思路方法：使用 Class.forName 加载 PostgreSQL 驱动。通过 DriverManager.getConnection 建立数据库连接，参数包括 URL、用户名、密码。创建 Statement 对象执行 SQL 查询。执行 select 语句查询 client 表中邮箱非空的客户信息。用 ResultSet.next 遍历结果集，输出客户姓名、邮箱、电话。在 finally 块中按顺序关闭 ResultSet、Statement、Connection。通过 try-catch 处理 ClassNotFoundException 和 SQLException 异常。具体代码如下图 2.12. 1 所示：

```
        Statement statement = null;
        ResultSet resultSet = null;
        try {
            Class.forName("org.postgresql.Driver");
            String URL = "jdbc:postgresql://localhost:5432/postgres";
            String USER = "gaussdb";
            String PASS = "Passwd123@123";
            connection = DriverManager.getConnection(URL, USER, PASS);
            statement = connection.createStatement();
            String SQL = "select * from client where c_mail is not null;";
            resultSet = statement.executeQuery(SQL);
            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"));
            }

        } catch (ClassNotFoundException e) {
            System.out.println("Sorry,can`t find the JDBC Driver!");
            e.printStackTrace();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            try {
                if (resultSet != null) {
                    resultSet.close();
                }
                if (statement != null) {
                    statement.close();
                }

                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
    }
}
```

图 2.12. 1 JDBC 体系结构和简单的查询

2.12.2 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。

思路方法：通过 Scanner 获取用户输入的用户名和密码；使用 Class.forName 加载 MySQL 驱动；通过 DriverManager.getConnection 建立数据库连接；创建 PreparedStatement 对象并绑定 SQL 查询参数，查询 client 表中匹配邮箱和密码的记录；执行 executeQuery 获取结果集；通过 resultSet.next()判断是否存在匹配记录，输出登录结果；最后在 finally 块中按顺序关闭 ResultSet、Statement 和 Connection，处理可能出现的异常。流程图如下图 2.12. 2 所示：

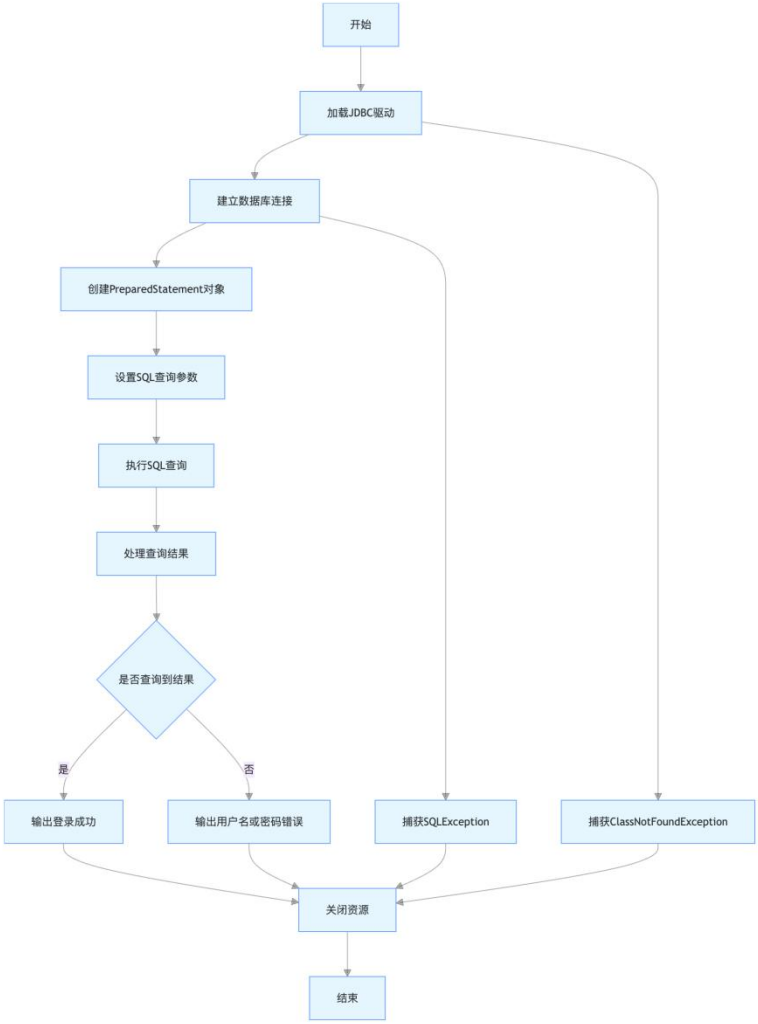

图 2.12.2 用户登录流程图

2.12.3 添加新用户

本关任务：编程完成向 client(客户表)插入记录的方法。

思路方法：和上一关差不多，通过 Scanner 读取用户输入的客户信息，使用 Class.forName 加载 MySQL 驱动，通过 DriverManager.getConnection 建立数据库连接；在 insertClient 方法中，创建 PreparedStatement 对象并绑定 SQL 插入语句的参数，执行 executeUpdate 方法将客户信息插入到 client 表中；最后在 finally 块中关闭 PreparedStatement 对象，确保资源释放，整个过程通过 try-catch 处理可能出现的 SQLException 异常。代码略。

2.12.4 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。

思路方法：在 removeBankCard 方法中，创建 PreparedStatement 对象并绑定 SQL 删除语句的参数, 执行 executeUpdate 方法删除 bank_card 表中匹配客户编号和银行卡号的记录。其他思路和上两关类似不再赘述。

2.12.5 客户修改密码

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。

思路方法：在 passwd 方法中，先查询客户邮箱是否存在，不存在则返回 2，存在则进一步验证原密码是否正确，不正确返回 3，正确则执行更新语句修改密码并返回 1；其他思路和前面类似不再赘述。

2.12.6 事务与转账操作

本关任务：编写一个银行卡转账的方法。

思路方法：通过 Scanner 获取转出卡号、转入卡号和转账金额；在 transferBalance 方法中，先查询转出卡是否存在、是否为信用卡及余额是否充足，再查询转入卡是否存在；若均有效，执行转出卡余额扣除，根据转入卡类型确定转入金额正负并更新转入卡余额；代码略。

2.12.7 把稀疏表格转化为键值对储存

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。

思路方法：首先通过 JDBC 连接 MySQL 数据库，然后从 entrance_exam 表中查询所有学生记录，对于每个学生的每门科目的成绩（存储在多个列中），如果成绩不为空，就将学生编号、科目名称和成绩作为一条记录插入到 sc 表中，从而实现数据从宽表到稀疏表的转换。关键代码如下图 2.12. 3 所示：

```java
    public static int insertSC(Connection connection, int sno, String col_name,
int col_value){
        String sql = "insert into sc values(?, ?, ?)";
        try {
            PreparedStatement pps = connection.prepareStatement(sql);
            pps.setInt(1, sno);
            pps.setString(2, col_name);
            pps.setInt(3, col_value);
            return pps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return 0;
    }

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

图 2.12.3 把稀疏表格转化为键值对储存

3 课程总结

一、课程实践总体任务及完成情况

本次课程实践以 MySQL 为平台，围绕数据库系统原理展开系统性实训，涵盖数据库对象管理、数据处理、安全性控制、设计开发及内核实验等。任务包括定义数据库与表结构、实现完整性约束、完成复杂数据查询、开发 Java 应用、模拟并发控制等。实践中，通过头歌平台在 Linux 环境下完成操作。

二、主要工作内容归纳

1. 数据库对象管理：创建数据库、表，设置主码、外码、CHECK 等约束，修改表结构与约束，建立视图、存储过程、触发器及用户自定义函数。

2. 数据处理：完成复杂查询，包括多表连接、窗口函数、嵌套查询等；实现数据插入、修改、删除操作，处理稀疏表转换。

3. 系统控制：进行安全性控制，创建用户角色并授权；模拟并发场景，验证事务隔离级别，实现备份与恢复。

4. 应用开发：基于 JDBC 实现用户登录、客户管理、转账等功能，结合事务保证数据一致性。

三、心得体会与改进方向

实践中深刻体会到数据库设计需兼顾逻辑严谨性与性能优化，SQL 的灵活性让复杂查询实现更高效，而 Java 与数据库结合开发时，SQL 注入防范等安全规范至关重要。有待改进之处在于：部分高级查询场景的效率优化需进一步研究；并发控制中锁机制的底层原理可深入探索；

补充说明：

课程结束后，在课程平台上（如学院有要求变动或者平台出现提交方面的技术问题，则提交方式将另行通知并相应调整，例如以班为单位提交一个归档压缩文件，内含全班每位同学该提交的各自的压缩文件）**每位同学提交一个压缩文档，文件名为“课程名-学号-姓名.rar”，里面包含实践报告的 doc 和 pdf 版本、存储所有源代码的目录（目录中创建一个 readme.txt 文件对本代码目录的结构和内容做必要的说明，然后自行组织本代码目录中的子目录或文件来提交本此处实践课所完成的全部代码）。**

上述提交资料是按照学院的管理要求待课程结束后提交学院资料室归档保存的资料，以备将来学校或者教育部的学科检查、抽查使用，因此请同学们做好相关的提交工作。