

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：	CS2202
学 号：	U202215378
姓 名：	冯瑞琦
指导教师：	胡贯荣

分数	
教师签名	

2024 年 6 月 30 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	1
2 任务实施过程与分析	2
2.1 数据库、表与完整性约束的定义(CREATE)	2
2.2 基于金融应用的数据查询(SELECT).....	3
2.3 数据查询(SELECT)-新增	4
2.4 视图	5
2.5 存储过程与事务	6
2.6 触发器	7
2.7 用户自定义函数	7
2.8 并发控制与事务的隔离级别	8
2.9 数据库设计与实现	9
2.10 数据库应用开发(JAVA 篇).....	12
3 课程总结	15
3.1 总任务完成情况	15
3.2 主要工作归纳	15
3.3 心得体会	15
附录	16

1 课程任务概述

“数据库系统原理实践”课程是配合“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，基础内容涉及以下几个部分，并可结合实际对 DBMS 原理的掌握情况向内核设计延伸：

- 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，实验环境为 Linux 操作系统下的 MySQL 8.0.28（主要为 8.0.28 版本，部分关卡使用 8.0.22 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。

登录头歌上的本课程平台后，将会看到总体任务由一系列实训任务构成，而进入每个实训后将会看到其由若干关卡组成，关卡依据其难易程度和工作量会有不同的分值。依据每个关卡布置的任务，在头歌界面输入相应答题代码后运行头歌的测评试功能，系统会自动依据运行结果评判该关卡是否通过，并记录相应分值，最终得分不少于 100 分。

任务共分解为 16 个实训，分别为实训 1 数据库、表与完整性约束定义，实训 2 表结构与完整性约束的修改，实训 3 基于金融应用的数据查询，实训 4 数据查询-新增，实训 5 数据的插入、修改与删除，实训 6 视图，实训 7 存储过程与事务，实训 8 触发器，实训 9 用户自定义函数，实训 10 安全性控制，实训 11 并发控制与事务的隔离级别，实训 12 备份+日志：介质故障与数据库恢复，实训 13 数据库设计与实现，实训 14 数据库应用开发（JAVA 篇），实训 15 存储管理，实训 16 索引管理。

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 1~12、第 13 的第一个、第 14 个实训任务，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、数据库应用开发、数据库设计与实现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)

本小节任务主要是数据库初始化时的部分基本操作，例如数据库的创建，表的创建以及表的主码约束，创建外码约束，CHECK 约束，DEFAULT 约束和 UNIQUE 约束。任务要求掌握数据库以及表的创建的基本语句，并且能够给创建的数据库加上不同类型的约束。由于较为基础，只将其中一关呈现在实验报告中。

本任务已完成 1-6 所有关卡。

2.1.1 CHECK 约束

1) 编程要求

在数据库 MyDb 中创建表 products，并分别实现对品牌和价格的约束，两个 CHECK 约束名称分别为 CK_products_brand 和 CK_products_price，主码约束不要显示命名。

2) 代码思路

以防数据库 MyDb 不存在，首先创建它，并将它作为工作数据库。

使用 CREATE TABLE 语句创建一个名为 products 的表。该表包含 pid 类型为 CHAR(10)，作为主键；name 类型为 VARCHAR(32)；brand 类型为 CHAR(10)，并添加名为 CK_products_brand 的 CHECK 约束，确保 brand 列的值只能是 'A' 或 'B'；price 类型为 INT，并添加名为 CK_products_price 的 CHECK 约束，确保 price 列的值必须大于 0。

3) 源代码

SQL 语句如下。经测试结果与预期输出一致。

```
create database MyDb; use MyDb;
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) );
```

2.2 基于金融应用的数据查询(Select)

本实训采用的是某银行的一个金融场景应用的模拟数据库，数据库中有客户表、银行卡表、理财产品表、保险表、基金表和资金表共六个表（见附录）。测试库中有已有相应测试数据，要求依据关卡任务需求完成相应查询动作。

本任务已完成 1-19 所有关卡。

2.2.1 查询投资总收益前三名的客户

1) 编程要求

查询当前总的可用资产收益(被冻结的资产除外)前三名的客户的名称、身份证号及其总收益，按收益降序输出，总收益命名为 `total_income`。不考虑并列排名情形。

2) 代码思路

内连接 `client` 和 `property` 表，筛选出 `pro_status` 为“可用”的资产，按客户 ID 分组并计算每个客户的总收益，按总收益降序排列并限制结果为前三名。

3) 源代码

```
select
    c_name,c_id_card,sum(pro_income) as total_income
from client inner join property on pro_c_id = c_id and pro_status
= "可用"
group by c_id
order by sum(pro_income) desc limit 3;
```

2.2.2 购买基金的高峰期

1) 编程要求

查询 2022 年 2 月购买基金的高峰期。至少连续三个交易日，所有投资者购买基金的总金额超过 100 万(含)，则称这段连续交易日为投资者购买基金的高峰期。只有交易日才能购买基金,但不能保证每个交易日都有投资者购买基金。2022 年春节假期之后的第 1 个交易日为 2 月 7 日，周六和周日是非交易日，其余均为交易日。请列出高峰时段的日期和当日基金的总购买金额，按日期顺序排序。总购买金额命名为 `total_amount`。

2) 代码思路

最内层的查询从 `property` 和 `fund` 表中选取数据，根据采购时间 `pro_purchase_time`、采购数量 `pro_quantity` 和金额 `f_amount` 计算每一天的采购总金额 `amount`，并且生成一个自定义的工作日数 `workday`，该值根据采购日期与 2021 年 12 月 31 日的差值减去周数的两倍计算得出。同时，这一层过滤出采购类型为 3 且采购时间为 2022 年 2 月的记录，并按日期进行分组。中间层查询基于第一层的结果，按日期生成行号 `rownum`，计算每条记录的 `workday` 和 `rownum` 的差值，进行分区，使用窗口函数 `count(*) over(partition by t2.workday - t2.rownum)` 计算连续采购记录的计数 `cnt`。最外层查询基于中间层的结果，筛选出连续采购天数大于等于三天的记录，最后输出这些记录的采购时间 `pro_purchase_time` 和总金额 `total_amount`。

2.3 数据查询(Select)-新增

本小节子任务仍然以第 2.2 子任务的数据库内容为背景，但内容与统计、相似性推荐相关，主要内容为 SQL 语句中数据查询的新增部分。

本实训已完成 1-6 全部关卡。

2.3.1 查找相似的理财产品

1) 编程要求

请用一条 SQL 语句完成以下查询任务：

在某些推荐方法中，需要查找某款理财产品相似的其他理财产品，不妨设其定义为：对于某款理财产品 A，可找到持有 A 数量最多的“3”个（包括所有持有相同数量的客户，因此如有 3 个并列第一、1 个第二、一个第三，则排列结果是 1,1,1,2,3）客户，然后对于这“3”个客户持有的所有理财产品（不包含产品 A 自身），每款产品被全体客户持有总人数被认为是和产品 A 的相似度，若有相似度相同的理财产品，则为了便于后续处理的确定性，则这些相似度相同的理财产品间按照产品编号的升序排列。按照和产品 A 的相似度，最多的“3”款（同上理，前 3 名允许并列的情况，例如排列结果是 1,2,2,2,3）理财产品，就是产品 A 的相似的理财产品。

请查找产品 14 的相似理财产品编号（不包含 14 自身）(`pro_pif_id`)、该编号的理财产品的客购买客户总人数(`cc`)以及该理财产品对于 14 号理财产品的相似度排名值 (`prank`)。

注意结果输出要求：按照相似度值降序排列，相同相似度的理财产品之间则按照产品编号的升序排列。

2) 代码思路

由于条件较为复杂，代码分为 5 层嵌套查询，查询出的集合命名为 t1~t5。

通过嵌套查询找出持有产品 14 数量最多的前 3 名客户，使用 `dense_rank()` 窗口函数对客户按持有数量降序排名，并确保包括所有持有相同数量的客户（即排名结果可能出现并列）。然后，通过这些客户与 `property` 表进行自然连接，筛选出这些客户持有的所有其他理财产品，并按产品编号分组，计算每款产品被这些客户持有的总人数，结果命名为 `cc`。最后，外层查询按持有人数 `cc` 降序排列，并使用 `dense_rank()` 对这些产品进行排名，相同的值会排到同一名次，但遇到下一个值不会跳跃，按连续的顺序排名。将排名结果命名为 `prank`，并输出产品编号、持有总人数及其相似度排名。相似度排名值越高表示与产品 14 的相似度越高，相同相似度的产品按编号升序排列。

2.4 视图

本小节要求掌握视图的创建、对其查询和分组统计。视图是是数据库中的一种虚拟表，它基于数据库中的其他表（或视图）创建，并不直接存储数据。视图通过一个 `SELECT` 查询定义，当访问视图时，这个查询会被动态执行，从而展示查询结果。

本任务已完成 1-2 全部关卡。

2.4.1 基于视图的查询

1) 编程要求

基于上一关创建的视图 `v_insurance_detail` 进行分组统计查询，列出每位客户的姓名，身份证号，保险投资总额(`insurance_total_amount`)和保险投资总收益(`insurance_total_revenue`)，结果依保险投资总额降序排列。

2) 源代码及注释

```
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 #从视图 v_insurance_detail 中选择数据
group by c_id_card #根据身份证号分组
order by insurance_total_amount desc; #根据保险总金额降序排序
```


2.5 存储过程与事务

本小节的任务为基于对存储过程与事务的了解，掌握变量的定义与赋值、复合语句与流程控制语句，掌握存储过程的定义、创建和查询以及删除。任务关卡包括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。

本任务已完成 1-3 所有关卡。

2.5.1 使用事务的存储过程

1) 编程要求

在金融应用场景数据库中，编程实现一个转账操作的存储过程 `sp_transfer`，实现从一个帐户向另一个帐户转账。

2) 代码思路

声明部分已经给出，过程中要用到 5 个输入参数：

`applicant_id` 付款人编号

`source_card_id` 付款卡号

`receiver_card_id` 收款人编号

`dest_card_id` 收款卡号

`amount` 转账金额

还有 1 个整型输出参数：

`return_code` 1：正常转账；0:转账不成功

先查询源账户的客户 ID、余额和卡类型，存入变量 `s_id`、`s_b` 和 `s_type`。
查询目标账户的客户 ID 和卡类型，存入变量 `r_id` 和 `r_type`。

再检查条件是否满足转账要求。检查条件有：

① 源账户客户 ID 是否与申请人 ID 匹配。

② 目标账户客户 ID 是否与接收人 ID 匹配。

③ 是否源账户为信用卡且目标账户为储蓄卡。

④ 源账户是否为储蓄卡且余额不足。

如果以上任何条件不满足，则设置返回码为 0，表示转账失败，并退出存储过程。

如果源账户是信用卡，转账金额变为负值，表示从信用卡借款。如果目标账户是信用卡，接收金额变为负值，表示还款。

完成事务后更新源账户的余额，减去转账金额；更新目标账户的余额，加上接收金额。返回码为 1，表示转账成功。

2.6 触发器

本小节要求知晓触发器的定义和几种常见的触发器触发方式，例如 `after`、`before`、`instead of` 等等，还要掌握触发器的创建、触发器内的特殊表的相关知识。

本任务只有 1 个关卡，已经完成。

2.6.1 为投资表 `property` 实现业务约束规则

1) 编程要求

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)。

2) 代码思路

创建触发器，这里创建的触发器为 `before` 类型，在对 `property` 表进行插入操作之前执行。创建触发器的语句如下：

```
CREATE TRIGGER before_property_inserted BEFORE INSERT ON
property
```

声明三个变量：`tp` 用于存储新记录的 `pro_type`，`id` 用于存储新记录的 `pro_pif_id`，`msg` 用于存储错误信息。

验证产品 ID 是否为 1 金融产品，2 保险产品，3 基金产品，分别用对应的语句在相应表中查询 ID 是否存在，若不存在，则设置错误信息 `not found`。若 `tp` 不是 1, 2, 3 中任何一个，说明类型非法，`type is illegal`。

如果 `msg` 不为空，则触发一个 SQL 异常，错误代码为 45000，错误信息为 `msg` 的内容。这会中断插入操作并返回错误信息。

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。

2.7 用户自定义函数

本小节是关于函数的定义和应用的，函数有很多种，比如标量函数、表函数，本任务中要求创建标量函数。

本任务只有 1 个关卡，已经完成。

2.7.1 创建函数并在语句中使用它

1) 编程要求

- ① 用 `create function` 语句创建符合以下要求的函数 `get_deposit`:
依据客户编号计算其所有储蓄卡余额的总和。
- ② 利用创建的函数，仅用一条 `SQL` 语句查询存款总额在 100 万(含)以上的客户身份证号，姓名和存款总额(`total_deposit`)，结果依存储总额从高到低排序。

2) 代码思路

定义一个名为 `get_deposit` 的函数，接受一个 `int` 类型的参数 `client_id`。指定函数的返回类型为 `numeric(10,2)`，即最多 10 位数字，其中 2 位小数。函数体在 `begin` 与 `end` 之间。

`SQL` 语句为从 `bank_card` 表中选择 `b_balance` 的总和，条件为 `b_type = "储蓄卡"`，即只考虑储蓄卡，按 `c_id` 分组，用了 `group by` 之后不能够用 `where`，故使用 `having` 子句过滤出 `b_c_id` 等于 `client_id` 的记录。

2.8 并发控制与事务的隔离级别

可串行化即多个事务并发执行是正确的，当且仅当其结果与按某一次序串行地执行这些事务时的结果相同。本小节要求理解事务的并发控制和可串行化执行。

本任务已经通过全部 1-6 关。

2.8.1 并发控制与事务的隔离级别

1) 编程要求

在右侧代码文件编辑器里根据提示补充适当的代码，将事务的隔离级别设置为 `read uncommitted`；并以 `rollback` 语句结束事务。

2) 源代码

本关卡中要求自己写的代码只有一行，如下：

```
# 设置事务的隔离级别为 read uncommitted
set session transaction isolation level read uncommitted;
```

2.8.2 并发控制与事务的隔离级别

1) 编程要求

现有两个涉及该表的并发事务 t1 和 t2，分别定义在 t1.sql 和 t2.sql 代码文件中。事务 t1 两次查询航班 MU2455 的余票，事务 t2 修改航班 MU2455 的余票(减 1)。请对两个代码文件进行修改，使得两个事务并发执行的结果与 t2→t1 串行执行的结果相同。

2) 源代码

t1.sql 文件中代码如下：

```
use testdb1;
start transaction;#开始一个事务
select tickets from ticket where flight_no = 'MU2455';#第一次查询航班号为
'MU2455' 的航班剩余票数
set @n = sleep(2); #设置一个 2 秒的延迟，模拟处理时间
select tickets from ticket where flight_no = 'MU2455';
commit;
```

t2.sql 文件中代码如下：

```
-- 事务 2:
use testdb1;
start transaction;
set @n = sleep(1);
update ticket set tickets = tickets - 1 where flight_no = 'MU2455';
commit;
```

2.9 数据库设计与实现

本小节要求掌握数据库设计的阶段和每阶段的任务，了解概念模型，要求会画 E-R 图，并知晓逻辑模型与概念模型之间的关系以及在 DBMS 中的物理实现。

本小节分为关卡 1-3，已完成关卡 1。

2.9.1 从概念模型到 MySQL 实现

1) 编程要求

本关的任务，是根据一个已建好的逻辑模型，完成 MySQL 的实现。

2) 代码思路

在背景介绍中我们已经知道系统需考虑的实体有用户、旅客、机场、航空公司、民航飞机、航班常规调度表、航班表、机票，共 8 个，把每一个实体转换成一个关系，将其属性加入关系模型中，并选择主码，一般为 id。

E-R 图（图 2-1）已经给出，通过图片可以知晓各个关系表之间的关系，以此确定主码和外码。实体间的联系都清楚地标注在 ER 图中：

每个航空公司都有一个母港(机场)，又叫基地。大的机场可能会是多家公司的基地，小型机场可能不是任何航空公司的基地。每个航班属于一家航

航空公司，航空公司可以很多航班。任何一架民航飞机属于一家航空公司，航空公司可以有多架飞机。每架飞机可以执飞多个航班，一个飞行航班由一架飞机执飞。一个航班根据执飞机型可以售出若干机票。一张机票是某个特定航班的机票。

用户可以多次订票，旅客可以多次乘坐飞机。一张机票肯定是某个用户为某个特定的旅客购买的特定航班的机票。即机票信息不仅跟乘坐人有关，同时记录购买人信息(虽然两者有时是同一人)。为简单起见，订购时间没有考虑。无论常规计划的航班，还是实际飞行航班，都是从某个机场出发，到达另一个机场。但一个机场可以是很多个航班的出发地，也是很多航班的到达地。

如 flight 和 airline、airplane、flightschedule、airport 全都有多对多或一对多的关系，故把这些关系的主码全部作为 flight 的外码。

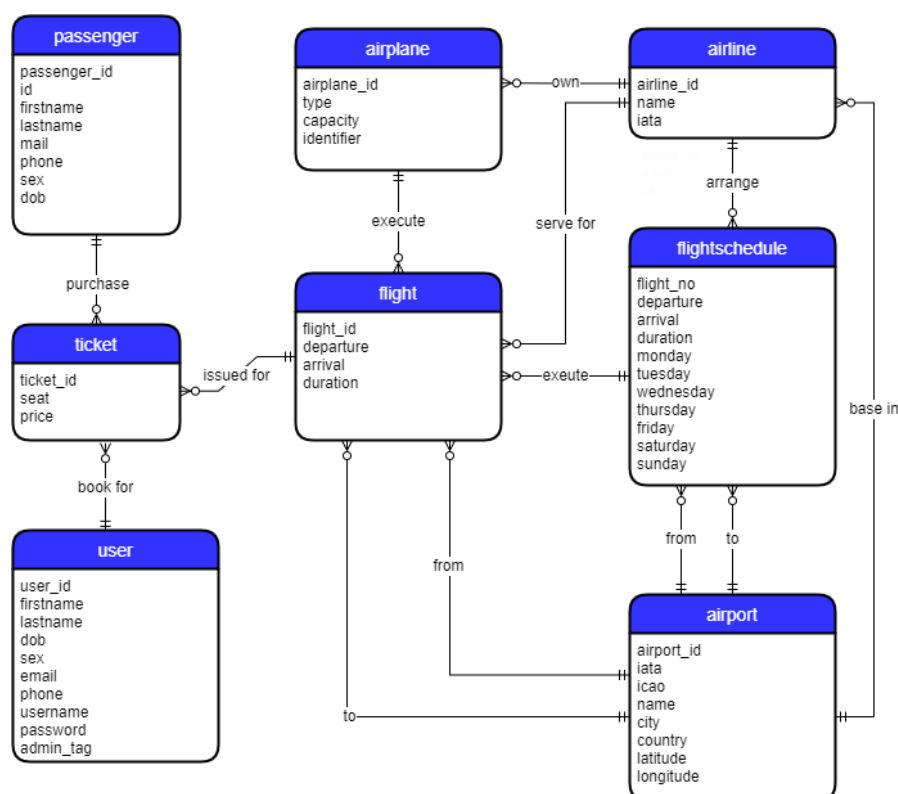


图 2-1 机票订票系统概念模型 ER 图

2.9.2 从需求分析到逻辑模型

设计一个影院管理系统。影院对当前的放映厅和电影进行排片，顾客到来后，可以购买任一排场的电影票，进入对应放映厅观看。系统中有以下实体集：

电影(movie): 属性有标识号(movie_ID)、电影名(title)、类型(type)、时长(runtime)、

首映日期(`release_date`)、导演姓名(`director`)、主演姓名(`starring`)。

顾客(`customer`): 属性有标识号(`c_ID`)、姓名(`name`)、手机号(`phone`)。

放映厅(`hall`): 属性有标识号(`hall_ID`)、放映模式(`mode`)、容纳人数(`capacity`)、位置(`location`)。

排场(`schedule`): 属性有标识号(`schedule_ID`)、日期(`date`)、时间(`time`)、票价(`price`)、票数(`number`)。

电影票(`ticket`): 属性有标识号(`ticket_ID`)、座位号(`seat_num`)。

实体间的关系描述如下：①. 顾客和电影票有一对多的购买关系。每位顾客可以买多张电影票，每张电影票被一位顾客购买。②. 电影票和排场有多对一的属于关系。一张电影票只属于一个排场，一个排场有多张电影票。③. 排场和电影有一对多的放映关系。每个排场放一部电影，每部电影可以在多个排场放映。④. 排场和放映厅有一对多的位于关系。每个排场位于一个放映厅，每个放映厅可以安排多个排场。

总体思路和 2.9.1 相同。

2.9.3 建模工具的使用

根据一个 2.9.1 已建好的逻辑模型，完成 MySQL 的实现。物理结构设计的主要工作，是在逻辑结构的基础上，确定存储结构和存取方法，将所有的数据类型针对具体的 DBMS 确定化等。

本关卡我没有选择完成，在此不多做描述。

2.9.4 制约因素分析与设计

在设计 `flight_booking` 数据库时，我们充分考虑了社会、健康、安全、法律、文化及环境等制约因素。为了保障数据的完整性与安全性，设置了严格的外键约束和非空约束，确保所有引用的数据都是有效的。此外，为防止数据泄露和非法访问，针对用户密码进行了加密存储，并通过 `admin_tag` 字段区分普通用户与管理员，实现权限管理。我们还考虑到法律合规性，确保数据存储和处理符合相关法律法规，特别是在个人信息保护方面。文化和环境因素的考虑体现在对多语言支持的预留，以及数据库设计中的可扩展性，以适应不同地区和业务需求的变化。通过这些设计，保证系统的高效运行与用户数据的安全，提升用户的信任度和满意度。

2.9.5 工程师责任及其分析

在设计和实施 `flight_booking` 数据库解决方案的过程中，工程师需承担多个方面的责任，以确保系统的安全性、可靠性和法律合规性。首先，工程师需确保数据的完整性和安全性，防止数据泄露和篡改，同时遵循相关法律法规，保护用户隐私。其次，工程师需进行性能优化，保障系统在高负载下的稳定运行。工程

师还需考虑文化多样性和环境影响,通过预留多语言支持和优化数据库资源使用,提升系统的可扩展性和环保性。最后,工程师需进行详细的文档编写和团队培训,确保项目的可持续发展和团队成员的有效协作。这些措施不仅提高了系统的用户体验和信任度,也体现了工程师应承担的社会责任和职业道德。

2.10 数据库应用开发(JAVA 篇)

本小节的任务是基于 JAVA 的数据库应用开发,本小节涉及多个关卡,要求掌握 JDBC 的体系结构及核心组件。关卡涉及 JDBC 简单查询,用户登录、更新,把稀疏表格转为键值对存储等等。

本任务已完成 1-7 全部关卡。

2.10.1 用户登录

1) 编程要求

编程体验客户登录功能。程序先后提示客户输用户名和密码。根据客户的输入,输出“登录成功”或“用户名或密码错误!”,如果用户名和密码匹配成功,输出前者,其它情况输出后者(包括该用户不存在)。

2) 代码思路

首先,我们需要了解 JDBC 的体系结构。JDBC (Java DataBase Connectivity,java 数据库连接)是一种用于执行 SQL 语句的 Java API,可以为多种关系数据库提供统一访问,它由用 Java 语言编写的类和接口组成。JDBC 提供了一种基准,据此可以构建更高级的工具和接口,使数据库开发人员能够编写数据库应用程序。其体系结构如图 2-2 所示。

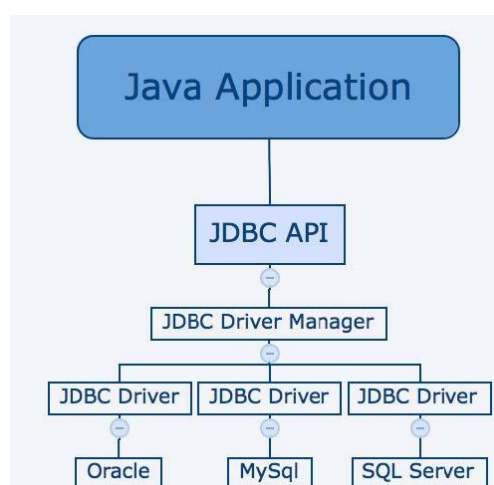


图 2- 2 JDBC 体系结构图

声明 Connection、Statement 和 ResultSet 对象，用于与数据库建立连接和执行 SQL 查询。此外，通过 Scanner 对象接收用户输入的用户名和密码。

接下来，尝试加载 MySQL JDBC 驱动程序，并使用提供的数据库 URL、用户名和密码建立数据库连接。连接成功后，创建一个 Statement 对象，用于执行 SQL 查询。

在 try 块结束后，无论是否抛出异常，都会执行 finally 块中的代码，确保 ResultSet、Statement 和 Connection 对象被正确关闭，以释放数据库资源。主要部分源代码如图 2-3 所示。

```
19      try {
20          //注册驱动程序，将驱动程序的类文件动态加载到内存中，并将其自动注册。
21          Class.forName("com.mysql.cj.jdbc.Driver");
22          //创建数据库连接对象
23          String userName = "root";
24          String passWord = "123123";
25          String url = "jdbc:mysql://127.0.0.1:3306/finance?useUnicode=true&characterEncoding=UTF8&
useSSL=false&serverTimezone=UTC";
26          connection = DriverManager.getConnection(url, userName, passWord);
27
28          statement = connection.createStatement();
29          String SQL = "select * from client where c_mail = '"+loginName+"' ";
30          resultSet = statement.executeQuery(SQL);
31          if(resultSet.next()){
32              if(resultSet.getString("c_password").equals(loginPass))
33                  System.out.println("登录成功。");
34              else System.out.println("用户名或密码错误！");
35          }
36          else System.out.println("用户名或密码错误！");
37
38      } catch (ClassNotFoundException e) {
```

图 2-3 用户登录源代码片段

3) 测试结果

测试通过，如图 2-4 所示。



图 2-4 用户登录测试结果

2.10.2 把稀疏表格转为键值对存储

1) 编程要求

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。sc 表初始为空表，程序依前述规则将 entrance_exam 表的值转写到 sc 表。对每一行，请从左至右依次考察每一列，转存非空列。

2) 代码思路

定义 JDBC 驱动程序的类名、数据库的 URL 以及连接数据库所需的用户名和密码。insertSC 接收一个数据库连接对象 con、学生编号 sno、列名 col_name 和列值 col_value，并将这些数据插入到 sc 表中。

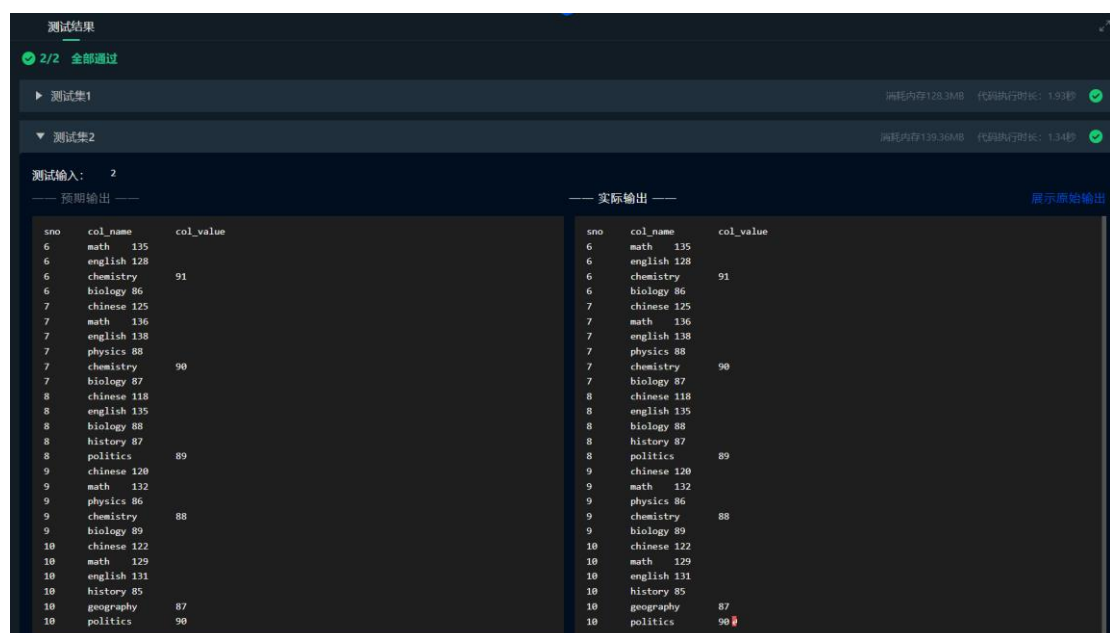
主方法 main 实现思路如下：

在 Class.forName(JDBC_DRIVER) 加载 JDBC 驱动程序。使用 DriverManager.getConnection 方法创建与数据库的连接对象 con。定义一个包含所有科目名称的数组 subject。执行查询 select * from entrance_exam，获取 entrance_exam 表中的所有数据。使用 while (res.next()) 循环遍历结果集中的每一行数据，获取当前行的学生编号 sno。遍历每个科目名称 sub，获取对应的分数 score。如果分数不为空，则调用 insertSC 方法将数据插入到 sc 表中。

最后，在 try-catch 块中捕获并处理可能的 SQLException 和 ClassNotFoundException 异常，出现异常时能够打印堆栈跟踪信息。

3) 测试结果

两个测试集均通过，如图 2-5 所示。



sno	col_name	col_value
6	math	135
6	english	128
6	chemistry	91
6	biology	86
7	chinese	125
7	math	136
7	english	138
7	physics	88
7	chemistry	90
7	biology	87
8	chinese	118
8	english	135
8	biology	88
8	history	87
8	politics	89
9	chinese	120
9	math	132
9	physics	86
9	chemistry	88
9	biology	89
10	chinese	122
10	math	129
10	english	131
10	history	85
10	geography	87
10	politics	90

图 2-5 稀疏表格转换为键值对测试结果

3 课程总结

3.1 总任务完成情况

本课程的总体任务是通过头歌平台的实训任务，让我们对数据库有基本的了解，搭建起数据库的知识框架，并且熟练掌握数据库、表的创建，数据的更新与查询相关的 SQL 语句，理解视图的概念及用法，掌握存储过程与事务、触发器、安全性控制等章节的知识，并且通过实际应用融会贯通。我在本次实践课程中最终完成了课程平台中实训 1~实训 12 中所有关卡、实训 13 的第一关和实训 14 中所有关卡，最终得分总分为 111 分。

3.2 主要工作归纳

在本次实践课程中，我深入学习了数据库的各个方面，取得了显著的进展。

在数据库、表与完整性约束定义方面，我掌握了创建数据库和表，以及实现主键、外键和检查约束的方法。在表结构与完整性约束的修改中，我学会了调整和优化表结构。通过大量练习，我熟练掌握了 SQL 语句的使用，尤其是在基于金融应用的数据查询中，学会了复杂查询、多表连接和嵌套查询等高级查询技巧。在视图的学习中，我理解了视图如何简化复杂查询并提高效率。

在数据库设计与实现部分，我参与了需求分析、概念设计和物理实现，学会了设计合理的数据模型。在数据库应用开发（JAVA 篇）中，我结合 Java 进行了数据库连接、操作和事务管理的实践，掌握了 JDBC 的使用。

在存储过程与事务、触发器、用户自定义函数的实训中，我编写了实际应用的代码，理解了这些高级功能的重要性。在安全性控制部分，我学会了权限管理和数据加密。在并发控制与事务的隔离级别实训中，我理解了并发事务的处理机制。在备份与日志、存储管理和索引管理的实践中，我系统地掌握了数据库管理的各个方面。

3.3 心得体会

此次课程实践让我对数据库有了全面而深入的理解，从简单表创建到复杂查询、存储过程与事务管理，每个环节都让我受益匪浅。在实际操作中，我遇到了许多挑战，如复杂查询的性能优化和存储过程的逻辑设计，通过查阅资料和反复实践，我逐渐克服了这些困难，极大地提高了问题解决能力和实际操作水平。头歌平台的自动测评给了我很大的激励和肯定。

数据库技术不仅是理论知识的积累，更需要通过实际操作和应用来不断完善。此次实践让我意识到自己的不足，在复杂查询优化和高并发事务处理方面还需提升。通过不断学习和实践，我相信自己能在数据库领域取得更大进步。

附录

实训 3 基于金融应用的数据查询(Select)

数据库中表，表结构以及所有字段的说明如下：

表 1 client(客户表)

字段名称	数据类型	约束	说明
c_id	INTEGER	PRIMARY KEY	客户编号
c_name	VARCHAR(100)	NOT NULL	客户名称
c_mail	CHAR(30)	UNIQUE	客户邮箱
c_id_card	CHAR(20)	UNIQUE NOT NULL	客户身份证
c_phone	CHAR(20)	UNIQUE NOT NULL	客户手机号
c_password	CHAR(20)	NOT NULL	客户登录密码

表 2 bank_card(银行卡)

字段名称	数据类型	约束	说明
b_number	CHAR(30)	PRIMARY KEY	银行卡号
b_type	CHAR(20)	无	银行卡类型(储蓄卡/信用卡)
b_c_id	INTEGER	NOT NULL FOREIGN KEY	所属客户编号,引用自 client 表的 c_id 字段。
b_balance	NUMERIC(10,2)	NOT NULL	余额,信用卡余额系指已透支的金额

说明：银行卡类型只有“储蓄卡”或“信用卡”两种取值。对于 b_balance 列，如果

b_type, b_balance 取值为(“储蓄卡”,10000), 表示这张储蓄卡内有 10000 的储蓄余额;而 b_type, b_balance 取值为(“信用卡”,10000),表示这张信用卡已经透支 10000 元。

表 3 finances_product(理财产品表)

字段名称	数据类型	约束	说明
p_name	VARCHAR(100)	NOT NULL	产品名称
p_id	INTEGER	PRIMARY KEY	产品编号
p_description	VARCHAR(4000)	无	产品描述
p_amount	INTEGER	无	购买金额
p_year	INTEGER	无	理财年限

表 4 insurance(保险表)

字段名称	数据类型	约束	说明
i_name	VARCHAR(100)	NOT NULL	保险名称
i_id	INTEGER	PRIMARY KEY	保险编号
i_amount	INTEGER	无	保险金额
i_person	CHAR(20)	无	适用人群
i_year	INTEGER	无	保险年限
i_project	VARCHAR(200)	无	保障项目

表 5 fund(基金表)

字段名称	数据类型	约束	说明
f_name	VARCHAR(100)	NOT NULL	基金名称
f_id	INTEGER	PRIMARY KEY	基金编号
f_type	CHAR(20)	无	基金类型

字段名称	数据类型	约束	说明
f_amount	INTEGER	无	基金金额
risk_level	CHAR(20)	NOT NULL	风险等级
f_manager	INTEGER	NOT NULL	基金管理者

说明：以上 3 张表中的金额都指每购入一份所要花费的金额。

表 6 property(资产表)

字段名称	数据类型	约束	说明
pro_id	INTEGER	PRIMARY KEY	资产编号
pro_c_id	INTEGER	NOT NULL	客户编号 FOREIGN KEY
pro_pif_id	INTEGER	NOT NULL	业务约束
pro_type	INTEGER	NOT NULL	商品类型:1 表示理财产品;2 表示保险;3 表示基金
pro_status	CHAR(20)	无	商品状态
pro_quantity	INTEGER	无	商品数量
pro_income	INTEGER	无	商品收益
pro_purchase_time	DATE	无	购买时间

说明：1.商品状态只有”可用”或”冻结”两种取值。2.商品收益指的是本条资产记录所记录商品的总收益,例如 f_id, f_amount 为(1031, 10000), pro_pif_id, pro_type, pro_quantity, pro_income 为(1031,3,10,27000)表示这条资产记录购入 10 份 1037 号基金, 总花费 10*10000=100000 元, 总收益为 27000 元。

实训 11 并发控制与事务的隔离级别

背景数据库有表 ticket 记录了航班余票数, 其结构如下表所示:

列	类型	说明
---	----	----

列	类型	说明
flight_no	char(6)	primary key
tickets	int	余票数

有两个涉及该表的并发事务 t1 和 t2, 分别定义在 t1.sql 和 t2.sql 代码文件中。平台会让两个事务并发执行。

实训 14 数据库应用开发(JAVA 篇)

第七关 把稀疏表格转为键值对存储

设有高考成绩登记表 entrance_exam, 其结构如下:

列名	类型	说明
sno	int	学号, 主码
chinese	int	语文
math	int	数学
English	int	英语
physics	int	物理
chemistry	int	化学
biology	int	生物
history	int	历史
geography	int	地理
politics	int	政治

转存表 sc 结构如下:

列名	类型	说明
sno	int	学号
col_name	varchar(50)	列名
col_value	varchar(50)	列值