南昌大学

《数据库课程实践报告》

题　　目：火车售票系统

学生姓名：马星

学　　号：5418122020

专业班级：计算机科学与技术(卓越)221 班

指导教师：王炜立

时　　间：2024.9.9——2024.9.13

分　　数：

二 0 二四 年 九月

目录

一、实践的要求

铁路运输作为国家基础设施的重要组成部分，承载着大量的人员与货物流动。它不仅在促进区域经济发展、连接城乡、提供就业机会等方面发挥着至关重要的作用，还在紧急情况下如自然灾害时提供救援支持。铁路系统的高效运作直接关系到社会经济的稳定与发展，是国家综合交通网络中不可或缺的一环。

基于用户需求，火车售票系统应具备以下关键功能：一是实现用户身份验证和个人信息管理；二是提供实时车次查询、购买、退票等票务服务；三是提供个人信息的修改、订单查询、充值等个人功能。

除了功能性需求外，系统的非功能性需求同样重要。这包括系统的性能要求，如快速响应时间；可用性要求，确保系统的稳定运行和高可用性；安全性要求，包括数据加密、防止非法访问和攻击；以及可维护性，便于未来的升级和维护工作。

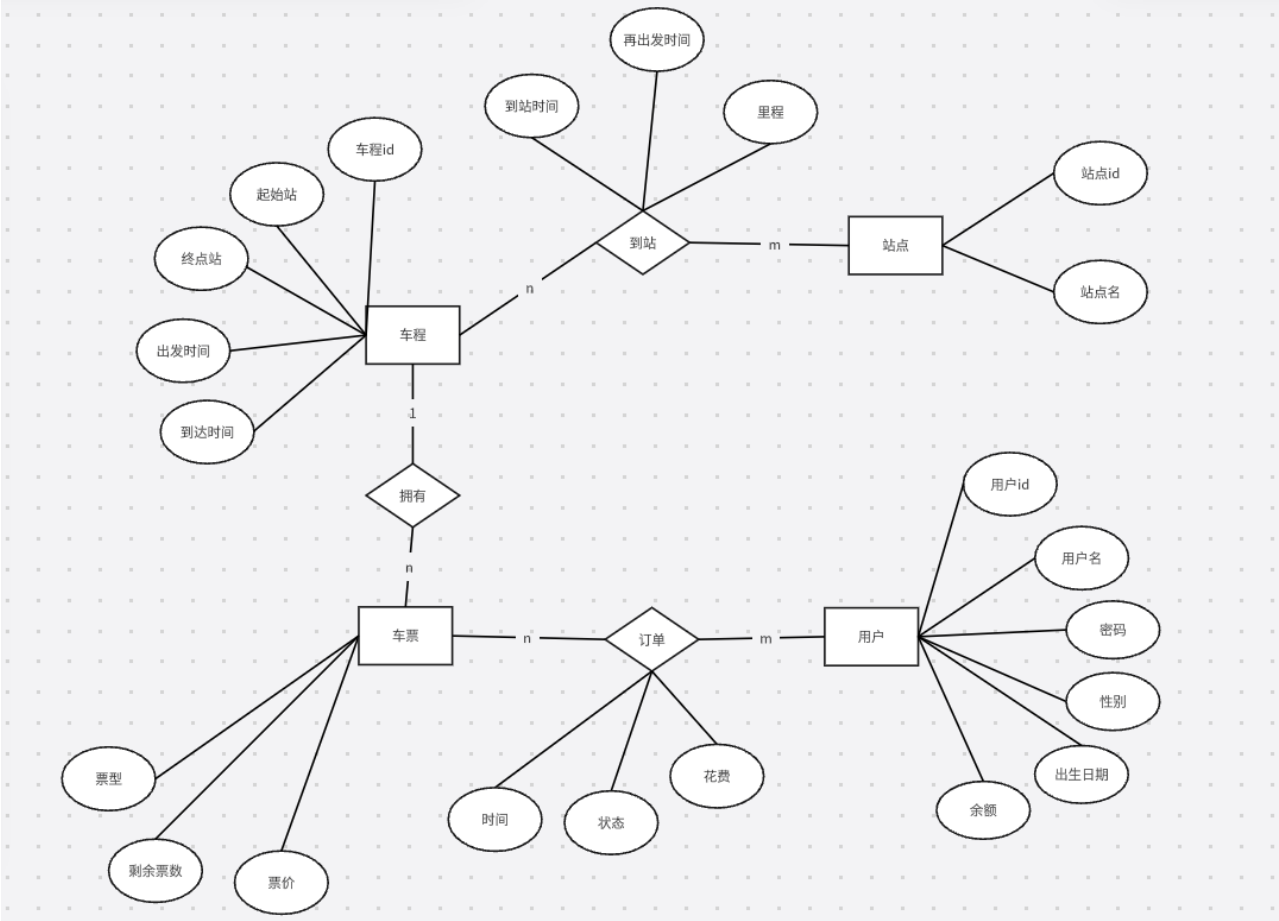
二、实践的数据库设计过程或编程部分

2.1 概要设计

(1) 需求分析

从车次入手,每趟车从起始站出发,中间经过若干站点,最后到达终点站,每趟车有票可以出售,票分为硬座票、硬卧票、软卧票,且具有不同的数量和价格。用户可以选择一趟车选择票型进行买票,用户查询车次时,输入出发站和终点站以及出行日期,系统提供经过这两个站点的车次。

(2) 概念设计 - ER 图



从需求中可分析出以下四个实体

车程: 每趟车有唯一 id 标识, 记录车次名称、起始站、终点站、出发时间、到达时间

站点: 每个站点拥有 id 和站点名

车票: 车票有不同的类型, 剩余票数和票价

用户: 用户具有 id、用户名、密码以及其他的个人信息

一趟车有多种票, 为 1:n 的关系

用户可以购买车票, 形成订单, 每种车票只有数量足够就可以被其他用户购买, 所以是 n:m 的关系

每趟车会经过不同的站点，每个站点在不同的时间有不同的车次停靠，也是 n:m 的关系

(3) 技术选型

使用 Java-FX 进行 UI 界面的开发， 数据库使用 mysql 数据， 通过 mybatis 持久层框架建立数据库的连接

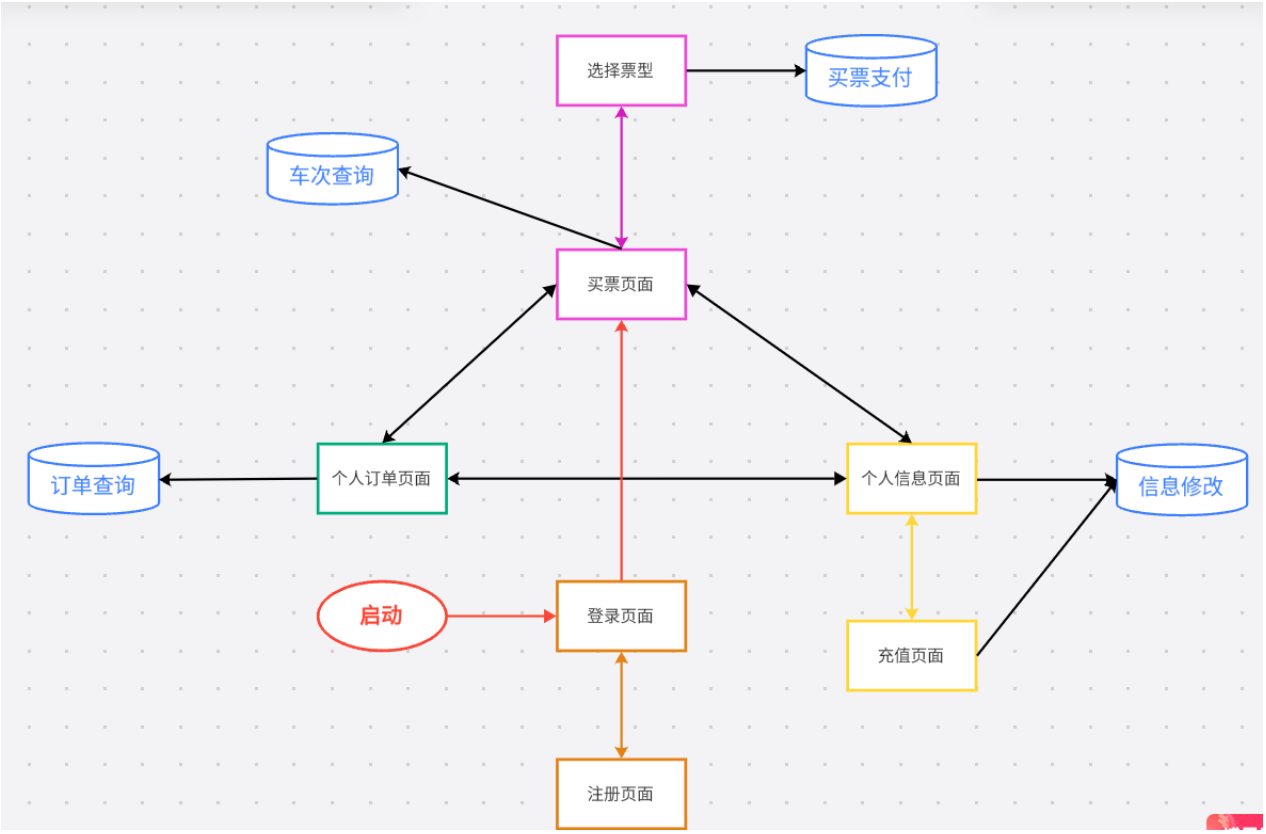
2.2 详细设计

(1) 逻辑结构 - 表设计

从 ER 图中可以得到以下六张表

表名	属性	主码
到站表	车程 id,站点 id,到达时间,再次出发时间,里程	(车程 id,站点 id)
订单表	订单 id,用户 id,车程 id,票型,下单时间,订单状态,花费,用户起始站,用户终点站	订单 id
车程表	车程 id,车次 id,起始站,终点站,开时,到时	车程 id
站点表	站点 id,站点名	站点 id
车票表	车程 id,票型,票价,剩余数量	(车程 id,票型)
用户表	用户 id,用户名,密码,手机号,性别,出生日期,余额	用户 id,用户名

(2) 流程设计



三、设计或编程实现

(1) 建表

```
-- arrive
create table arrive
(
    train_id    int      not null comment '车程 id',
    station_id  int      not null comment '站点名',
    primary key (train_id, station_id),
    arrive_time datetime null comment '到达时间',
    go_time     datetime null comment '再次出发时间',
```

```
    distance    int      not null comment '里程'
) comment '到站信息表';

-- orders
create table orders
(
    id                      int  not  null  comment '订单 id'  primary  key
auto_increment,
    user_id          int not null comment '用户 id',
    train_id         int not null comment '车程 id',
    type             varchar(100) comment '票型',
    time             datetime comment '下单时间',
    status           varchar(100) comment '订单状态',
    constraint order_status check ( status in ('未出行', '已完成', '已退
票')),
    cost             int not null comment '花费',
    constraint order_cost check ( cost >= 0 ),
    start_station_id int not null comment '用户起始站',
    end_station_id   int not null comment '用户终点站'
) comment '订单信息表';

-- ride
create table ride
(
    train_id         int          not null comment '车程 id' primary key
auto_increment,
    train_name       varchar(100) not null comment '车次名称',
    start_station_id int          not null comment '起始站 id',
```

```
    end_station_id   int          not null comment '终点站 id',
    start_time       DATETIME     not null comment '开时',
    end_time         DATETIME     not null comment '到时',
    constraint ride_time check ( end_time > start_time )
)
    comment '车程信息表';


-- station
create table station
(
    station_id  int not null comment '站点 id' primary key auto_increment,
    station_name varchar(100) not null comment '站点名' unique
) comment '站点信息表';

-- ticket
create table ticket
(
    train_id int not null comment '车程 id' primary key,
    type     int not null comment '票型',
    price    int not null comment '票价',
    remain   int not null comment '剩余数',
    constraint ticket_remain check ( remain >= 0 )
) comment '车票信息表';


-- user
create table user
```

```
(
    id int not null comment '用户 id' primary key auto_increment,
    username varchar(100) not null comment '用户名' unique,
    password varchar(100) not null comment '密码',
    phone    varchar(100) comment '手机号',
    birthday date         null comment '出生日期',
    sex      char(1)      null comment '性别',
    money    int          not null comment '余额',
    constraint user_余额 check (money >= 0),
    constraint user_手机号 check (phone = '未绑定' or (LENGTH(phone) = 11
AND phone regexp '^[0-9]+$'))
) comment '用户信息表';
```

(2) 存储过程

存储过程是一组预先编译的 SQL 语句，存储在数据库中，可以通过调用其名称来执行。它可以接收参数并返回结果，类似于编程语言中的函数。存储过程第一次创建时进行编译，之后每次调用都不需要重新编译，因此可以提高执行效率。

对于查询站点、查询车次,以及买票退票时更新票数和用户余额等非常频繁的操作,可以使用存储过程预编译,加快程序的运行效率

```
    create procedure UpdateTicketRemain(in trainId int, in tp varchar(100),
in up int)
    begin
        update ticket
        set remain = remain + up
        where train_id = trainId
          and ticket.type = tp;
```

```
end;
create procedure UpdateUserMoney(in userId int, in up int)
begin
    update user
    set money = money + up
    where id = userId;
end;
create procedure QueryStationName(in stationId int)
begin
    select station_name
    from station
    where station_id = stationId;
end;
create procedure QueryStationId(in stationName varchar(100))
begin
    select station_id
    from station
    where station_name = stationName;
end;
create procedure QueryTrainId(in startStation varchar(100),
                              in    endStation    varchar(100),    in
localDate DATE)
begin
    select train_id
    from arrive
             join station on arrive.station_id = station.station_id
    where station_name in (startStation, endStation)
    group by train_id
```

```
    having Count(DISTINCT station_name) = 2
       and Date(Min(arrive_time)) = localDate
       and Date(Min(go_time)) >= localDate;

end;
create procedure QueryTrainById(in id int)
begin
    select ride.train_id,
           train_name,
           s1.station_id as start_station_id,
           s2.station_id as end_station_id,
           start_time,
           end_time
    from ride
             join station s1 on ride.start_station_id = s1.station_id
             join station s2 on ride.end_station_id = s2.station_id
    where ride.train_id = id;
end;
create procedure QueryUserByName(in username varchar(100))
begin
    select *
    from user
    where user.username = username;
end;
```

(3) 触发器

在买票后,票的剩余数量应该减去 1,买票用户的账户余额应该减去花销

在退票后,票的剩余数量应该加 1,买票费用应该退回到用户账户中

```
 create trigger cancel_order # 退票后, 票数+1, 余额+cost
    after update
    on orders
    for each row
begin
    IF NEW.status = '已退票' THEN
        CALL UpdateTicketRemain(NEW.train_id, NEW.type, 1);
        CALL UpdateUserMoney(NEW.user_id, NEW.cost);
    end IF;
end;

create trigger buy_ticket # 买票后, 票数-1, 余额+cost
    after insert
    on orders
    for each row
begin
    CALL UpdateTicketRemain(NEW.train_id, NEW.type, -1);
    CALL UpdateUserMoney(NEW.user_id, -NEW.cost);
end;
```

(4) 外键

外键是数据库中用来维护数据一致性和完整性的一种机制。它是在两个表之间建立联系的一种方法，其中一个表的字段（外键）与另一个表的主键相对应。通过使用外键，数据库能确保数据的参照完整性，防止出现孤立的数据记录。

但是考虑到以下因素:

1. 数据修复与维护:

•在有外键约束的情况下，数据的修复和修改会变得复杂，可能需要先调整或临时解除外键关系，这增加了维护的难度

2. 业务逻辑与数据管理:

•将数据逻辑放在应用层处理而不是数据库层，能够为开发团队提供更多的控制力和灵活性，以应对快速变化的业务需求。

•实际开发中，经常需要在业务逻辑中处理复杂的数据关系，外键的严格约束可能限制了这些必要的灵活性。

3. 性能考虑:

•每次数据插入或更新时，数据库都需要检查外键的有效性，这会增加额外的查询负担，影响性能

所以并没有定义外键, 而是将外键的一致性约束, 放在了逻辑层面

四、测试结果


火车售票系统 - 我要买票
江西
上海
2024/9/13
查询
02:24
3时56分
06:20
¥144
江西
O8476
上海
购买
硬座 100张
144元/张
硬卧 100张
216元/张
软卧 100张
360元/张
10:09
2时4分
12:13
¥87
江西 始
A3663
终 上海
购买
硬座 100张
87元/张
硬卧 100张
131元/张
软卧 100张
219元/张
11:54
4时50分
16:44
¥158
江西 始
Y1264
终 上海
购买
硬座 100张
158元/张
硬卧 100张
237元/张
软卧 100张
396元/张
19:48
3时55分
23:43
¥144
江西
A4204
上海
购买
硬座 100张
144元/张
硬卧 100张
216元/张
软卧 100张
361元/张
我要买票
我的订单
个人信息


火车售票系统 - 选座购买
02:24
3时56分
06:20
江西
O8476
上海
硬座
¥144
100张
购买
硬卧
¥216
100张
购买
软卧
¥360
100张
购买


火车售票系统 - 我的订单
江西 - 上海 O8476
¥144
9月13日 02:24 出发
root 硬座
退 票
北京 - 四川 O7365
¥246
9月13日 01:37 出发
root 硬卧
退 票
江西 - 河北 S9512
¥142
9月12日 04:45 出发
root 硬座
已完成
江西 - 上海 O6458
¥165
9月12日 00:49 出发
root 硬卧
已完成
江西 - 上海 O6458
¥110
9月12日 00:49 出发
已完成
按下单时间(最近)
按出发时间(最近)
我要买票
我的订单
个人信息


火车售票系统 - 个人信息
用户名
root
密码
123456
性别
男
女
手机号
18397887506
出生日期
2004/4/23
余额
3137 元
去充值
保存
我要买票
我的订单
个人信息

五、问题和讨论

(1) 如何查询中转方案

假设车 A 由 A1 地经过 X 地到达 A2 地, 车 B 有 B1 地经过 X 地到达 B2 地, 而用户需要从 A1 地到 B2 地

A: A1 → X → A2

B: B1 → X → B2

用户: A1 → X → B2

用户可以在 X 地进行中转, 如何通过 车次-站点 的到站时刻表进行查询?

此为一次中转, 可能还有两次中转等情况

(2) 座位细化

如果将每张票对应一个座位, 且用户购买的是中间的一段路程, 其他时间这个座位依然可以购买, 又或者是同车换座等情况, 这种情况该怎么设计数据库?

六、总结体会

在这次的数据库实践课程实验中，我选择了“火车售票系统”作为我的实验题目。通过这个实验，我深刻体会到了数据库技术在实际应用中的重要性，尤其是在处理大量数据和保证数据一致性方面。以下是我在实验过程中的一些具体体会：

1. 需求分析的重要性：在开始设计数据库之前，我首先进行了详细的需求分析。这帮助我明确了系统需要实现的功能，例如车次查询、票务管理、用户登录等。这一过程让我了解到，一个好的数据库设计始于清晰完整的需求分析。

2. 数据库设计的复杂性：火车售票系统涉及到的信息包括车次信息、用户信息、票价信息、座位信息等。我需要考虑如何合理地组织这些数据，使之既能高效存储又能快速访问。在这个过程中，我学习到了如何设计合理的表结构及设置合适的索引来优化查询性能。

3. SQL 语言的应用：实验过程中，我使用 SQL 语言进行数据的增删改查操作。通过实际操作，我更加熟悉了 SQL 语句的编写，理解了事务处理和复杂的查询语句，如联查、子查询等，对 SQL 有了更深入的了解。

4. 数据一致性与完整性的维护：为了保证火车票数据的准确性，我运用了数据库的约束机制，如主键、外键、唯一约束等。这确保了数据库中的数据不会出现不一致或冗余的情况，这对于一个售票系统来说至关重要。

5. 测试与优化：开发完成后，我对系统进行了多轮测试，包括功能测试和性能测试。通过测试，我发现了一些问题并进行了相应的优化。这个过程让我意识到，持续的测试和优化对于保证系统的稳定运行是非常重要的。

通过本次实验，我不仅加深了对数据库理论知识的理解，而且通过实践提升了解决实际问题的能力。我相信，这次的实验经验将对我未来的学习和工作产生积极的影响。