

**《数据库设计与开发》**

**课程设计**

**题 目 机票购票管理系统**

姓 名

学 号

院 系

专 业

**二Ｏ一九 年 月 日**

目 录

1. **概述···············································1**
   1. **目的与要求······································1**
   2. **设计环境········································1**
2. **需求分析········································1**
   1. 系统功能要求设计····················································1
   2. 系统模块设计·····················································2
3. **总体设计···············································3**
   1. **数据表设计·······································3**
4. **数据库结构示意图·······································4**
5. **系统实现···············································5**
   1. 获取数据库连接···················································5
   2. 数据库接口嗯设计··················································6
      1. 管理员相关操作················································6
      2. 顾客相关操作·················································7
      3. 用户相关操作·················································8
6. **功能测试··············································9**
   1. 登陆模块····························································9
   2. 注册模块····························································10
   3. 用户管理模块······················································11
   4. 航班管理模块···················································12
   5. 班机管理模块····················································14
   6. 订单管理模块····················································15
7. **安全性测试············································15**
8. **总结····························16**

## 一 概述

### 1.1 目的与要求

随着科技的发展，飞机已经成为一重要的交通方式，而管理飞机订票信息、飞机航班信息、客户信息等也成为了一个必不可少的环节。该课程设计将设计一个机票预定系统，数据库中包含班机信息、用户信息、航班信息以及订单信息。目的旨在通过该课程设计，将数据库理论知识更加的巩固加深、加强动手能力与实践能力，学以致用，与现实生活中的应用充分结合起来。

### 1.2 设计环境

① mysql5.5

② jdk1.8

③ tomcat8.5

编程工具：

1. Eclipse

## 二 需求分析

### 2.1 系统功能要求设计

此系统实现如下系统功能：

**（**1）使得航班信息、班机信息、用户信息等信息管理工作更加清晰、条理化、自动化。

（2）通过用户名和密码登录系统，查询航班信息，班机信息，修改用户信息等功能。 容易地完成飞机信息的查询操作。

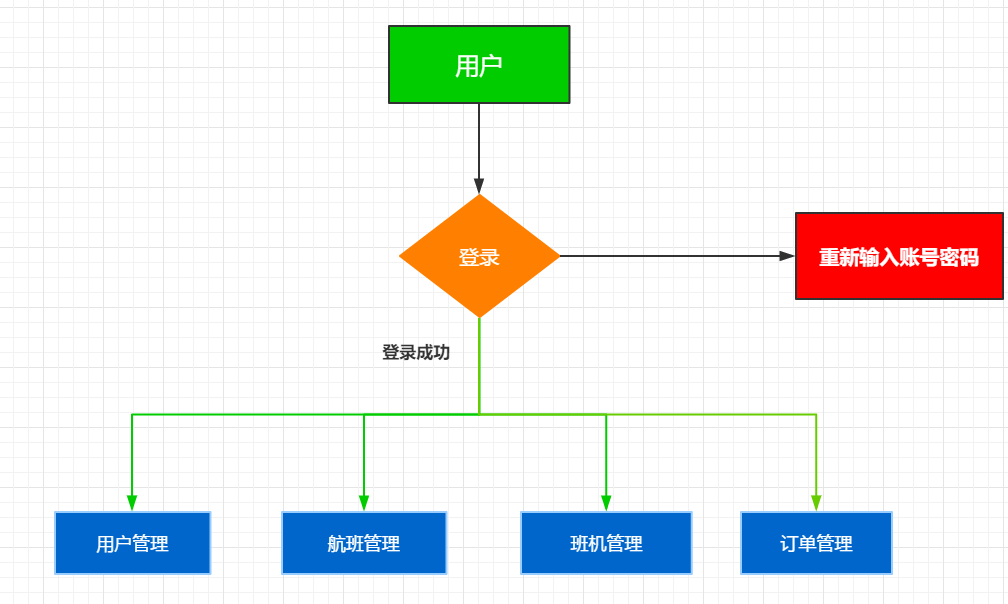
(3) 设计人机友好界面，功能安排合理，操作使用方便，并且进一步考虑系统在安全性，完整性，并发控制，备份和恢复等方面的功能要求。

### 2.2 系统模块设计

机票预定系统分为四个模块：

* 用户管理模块
* 用户信息的展示
* 用户信息的修改
* 航班管理模块
* 航班信息的展示
* 航班信息的修改
* 航班信息的删除
* 增加航班信息
* 班机管理模块
* 班机信息的展示
* 班机信息的删除
* 新增班机信息
* 订单管理模块
* 订单信息的展示

系统流程图如下：



### 三 总体设计

3.1 数据表设计

t\_user表基本情况数据表，结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| U\_NO | Int | 主键 | 用户编号 |
| U\_USERNAME | varchar(20) | 非空唯一 | 用户名 |
| U\_PASSWORD | varchar(20) | 非空 | 密码 |
| U\_NUCKNAME | varchar(20) | 非空 | 用户昵称 |
| U\_ID | varchar(20) | 非空唯一 | 用户ID |
| U\_TYPE | varchar(20) | 非空 | 用户类型 |
| U\_BALANCE | Int | 非空 | 用户余额 |

t\_flight表基本情况数据表，结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| F\_NO | int | 主键 | 航班编号 |
| F\_A\_NO | int | 非空 | 班机编号 |
| F\_START | varchar(25) | 非空 | 航班出发地 |
| F\_DIST | varchar(25) | 非空 | 航班目的地 |
| F\_PRICE | float | 非空 | 价格 |
| F\_TIME | timestamp | 非空 | 起飞时间 |
| F\_PSG\_NUM | int | 非空 | 当前乘客数 |

t\_airplane表基本情况数据表，结构如下

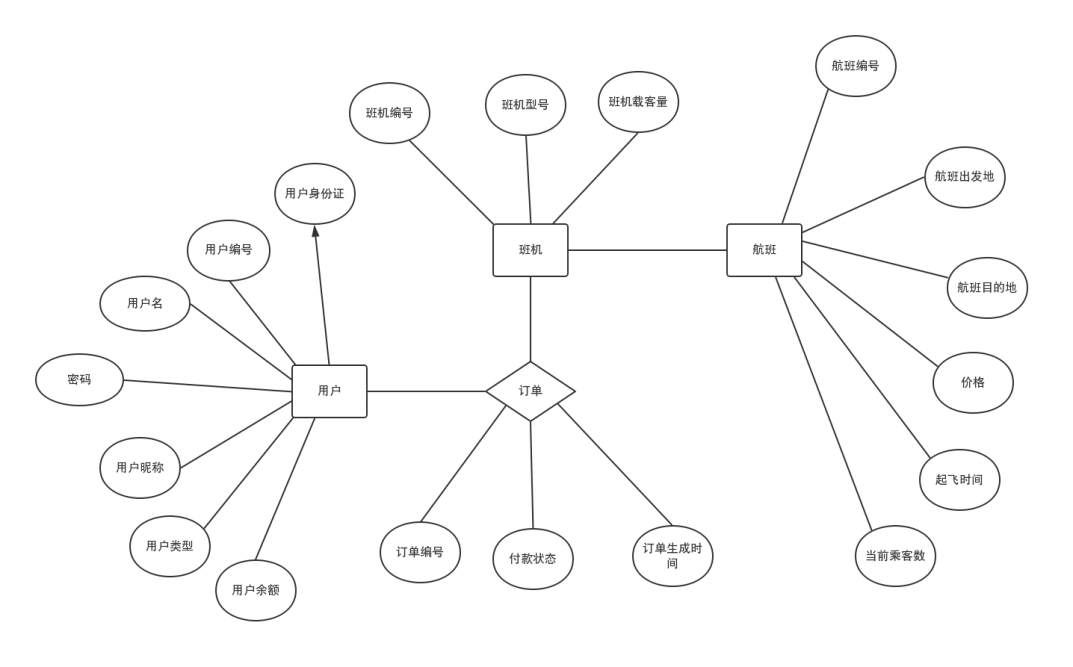
|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| A\_NO | int | 主键 | 班机编号 |
| A\_MODEL | Varchar(20) | 非空 | 班机型号 |
| A\_CAPACITY | int | 非空 | 班级载客量 |

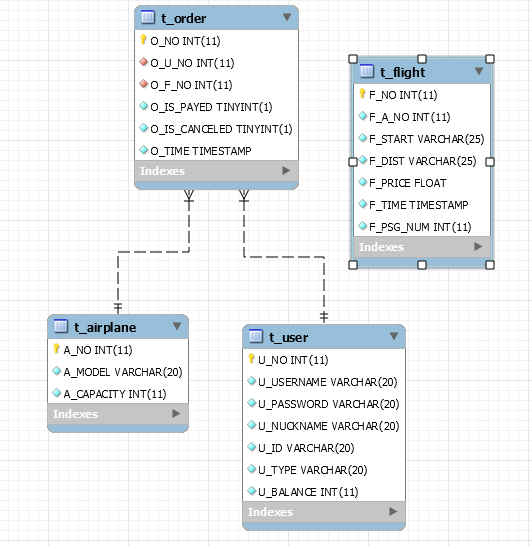
t\_order表基本情况数据表，结构如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| O\_NO | int | 主键 | 订单编号 |
| O\_U\_NO | int | 非空 | 用户编号 |
| O\_F\_NO | int | 非空 | 班机编号 |
| O\_IS\_PAYED | int | 非空 | 是否付款 |
| O\_IS\_CANCELED | int | 非空 | 是否取消 |
| O\_TIME | TIMESTAMP | 非空 | 订单生成时间 |

## 四 数据库结构示意图

实体关系ER图：





## 五 系统实现

展示部分核心代码

### 5.1 获取数据库连接

该系统使用了C3p0连接池得到方式获取数据库连接。

**C3p0配置文件：**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<c3p0-config>

<default-config>

<property name=*"driverClass"*>com.mysql.jdbc.Driver</property>

<property name=*"jdbcUrl"*>jdbc:mysql:///flight\_management</property>

<property name=*"user"*>root</property>

<property name=*"password"*>root</property>

</default-config>

</c3p0-config>

**封装与数据库操作相关的工具类：**

**public** **class** DaoUtils {

**private** **static** DataSource *source* = **new** ComboPooledDataSource();

**private** DaoUtils() {

}

// 获取数据源

**public** **static** DataSource getSource(){

**return** *source*;

}

// 获取连接

**public** **static** Connection getConn(){

**try** {

**return** *source*.getConnection();

} **catch** (SQLException e) {

e.printStackTrace();

**throw** **new** RuntimeException(e);

}

}

}

### 5.2 数据层接口设计

#### 5.2.1 管理员相关操作

**public** **interface** AdminDao {

/\*\*

\* 查询所有班机

\* **@return**

\*/

**public** List<AirPlane> queryAllAirPlane();

/\*\*

\* 添加新的班机

\* **@param** airPlane

\*/

**public** **void** insertAirPlane(AirPlane airPlane) **throws** FMException;

/\*\*

\* 删除一个班机

\* **@param** airPlane

\*/

**public** **void** deleteAirPlane(AirPlane airPlane) **throws** FMException;

/\*\*

\* 添加新的航班

\* **@param** flight

\*/

**public** **void** insertFlight(Flight flight) **throws** FMException;

/\*\*

\* 修改航班信息

\* **@param** flight

\*/

**public** **void** updateFlight(Flight flight) **throws** FMException;

/\*\*

\* 查询所有旅客信息

\* **@return**

\*/

**public** List<User> queryUsers();

/\*\*

\* 删除一条航班

\* **@param** flight

\*/

**public** **void** deleteFlight(Flight flight) **throws** FMException;

/\*\*

\* 查询所有航班信息

\* **@return**

\*/

**public** List<Flight> queryAllFlight();

// 删除用户

**public** **void** deleteUser(User user) **throws** FMException;

// 修改用户信息

**public** **void** updateUser(User user) **throws** FMException;

// 查询所有订单

**public** List<Order> queryAllOrder();

}

#### 5.2.2 乘客相关操作

**public** **interface** PassengerDao {

/\*\*

\* 根据出发地和目的地查询航班。

\* **@param** start

\* **@param** dist

\* **@return** 返回Flight表，若发生错误返回null

\*/

**public** List<Flight> findFlightByStartAndDist(String start,String dist);

/\*\*

\* 订机票

\* **@param** user

\* **@param** flight

\* **@throws** Exception

\*/

**public** **void** bookFlight(User user,Flight flight) **throws** FMException;

/\*\*

\* 取消订单

\* **@param** order

\*/

**public** **void** bounce(Order order) **throws** FMException;

/\*\*

\* 用户付款

\* **@param** user

\* **@param** flight

\*/

**public** **void** payFlight(Order order) **throws** FMException;

/\*\*

\* 获取用户的订单列表

\* **@param** user

\* **@return**

\*/

**public** List<Order> getOrderListOfUser(User user);

/\*\*

\* 获取用户已付款的订单列表

\* **@param** user

\* **@return**

\*/

**public** List<Order> getPayedOrderListOfUser(User user);

/\*\*

\* 获取用户未付款的订单列表

\* **@param** user

\* **@return**

\*/

**public** List<Order> getUnPayedOrderListOfUser(User user);

**public** List<Flight> getUnBookedFlight(User user);

**public** Order getOrder(Order order);

}

#### 5.2.3 用户相关操作

**public** **interface** UserDao {

/\*\*

\* 根据用户名密码检查登录，若查询到用户则返回该用户信息，若无结果返回null。

\* **@param** username

\* **@param** password

\* **@return**

\*/

**public** User login(String username,String password);

/\*\*

\* 添加用户信息

\* **@param** user

\* **@param** number

\*/

**public** **void** insertUser(User user) **throws** FMException;

/\*\*

\* 修改用户余额

\* **@param** user

\* **@param** number

\*/

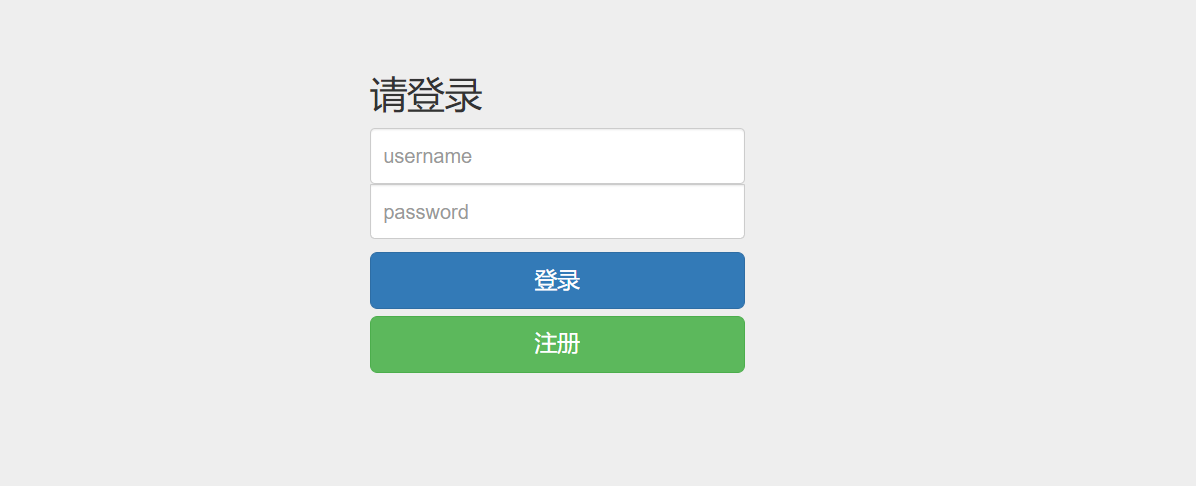
**public** **void** reCharge(User user, **float** number);

}

## 六 功能测试

### 6.1 登录模块

登录界面：



简单的进行登录验证功能：

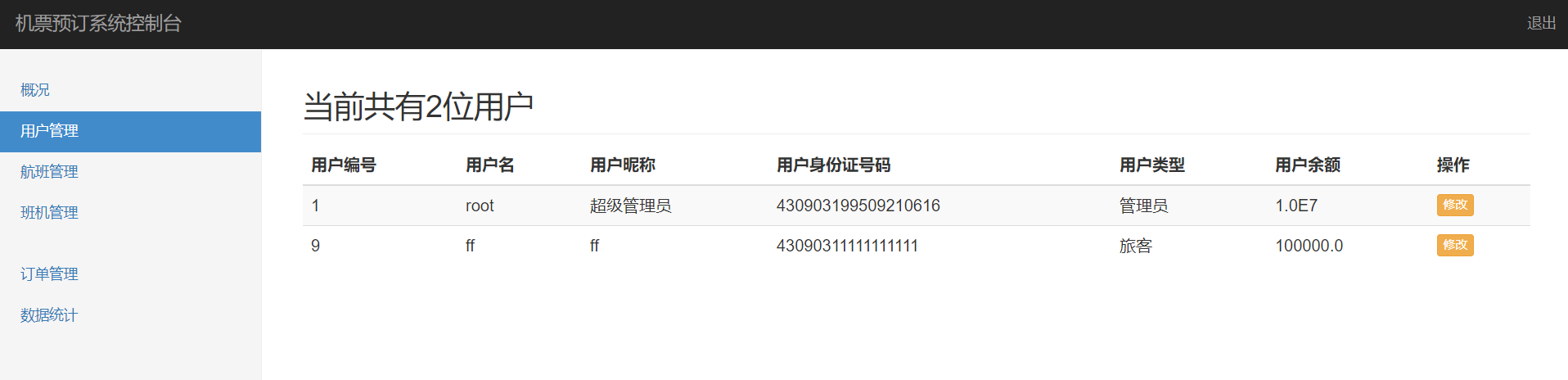


### 6.2 注册模块



### 6.3 用户管理模块

#### 6.3.1 用户信息显示



#### 6.3.2 用户信息修改



验证修改是否成功：



### 6.4 航班管理模块

#### 6.4.1 航班信息展示



#### 6.4.2 修改航班信息



查看是否修改成功：



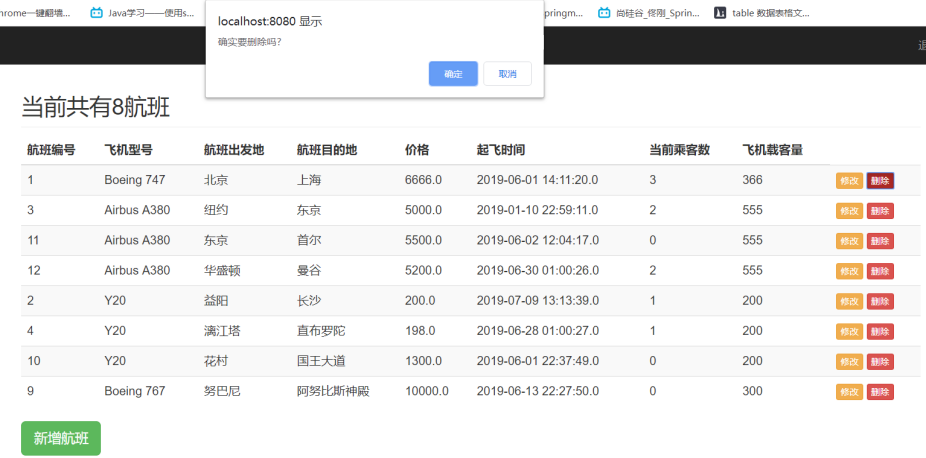
#### 6.4.3 航班信息的添加



查看是否添加成功：



#### 6.4.4 航班信息的删除

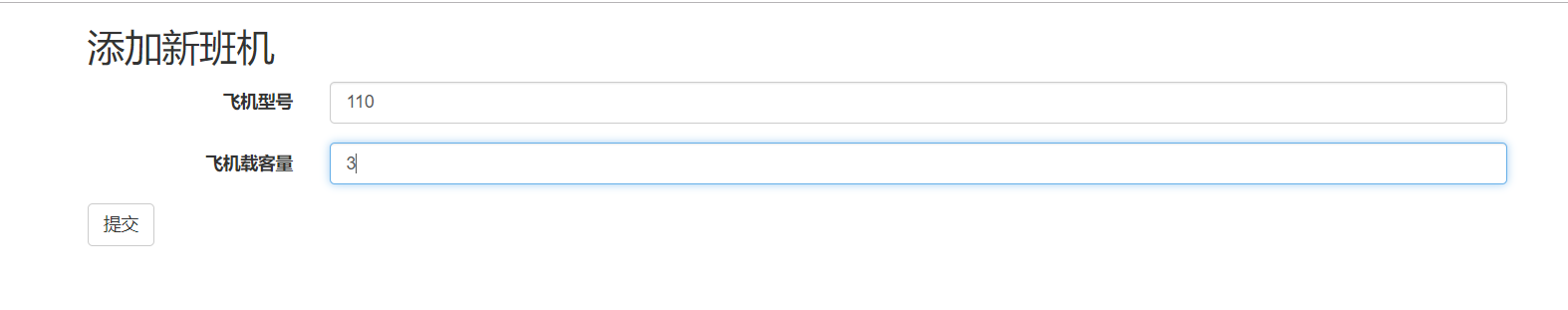


## 6.5 班机管理模块

### 6.5.1 班机信息的显示



### 6.5.2 航班信息的添加



验证是否添加成功：



### 6.5.3 航班信息的删除



## 6.6 订单管理模块

#### 6.6.1 订单信息的显示



## 七 安全性测试

经过简单的安全测试，由于正确的使用了MySQL JDBC中的语法，没有使用拼接字符串的方法。以我们的水平，想要在该平台上完成数据库的sql注入似乎很难。或许有几点需要注意的是数据库中的所有数据均是明文存储的，包括数据库中多数用户的密码等等。这中行为是极为不安全的。是潜在的安全已更换之一

## 八 总结

收获：

在这次数据库课程设计中，使我在数据库上的操作更为熟练。在编写程序的过程中，遇到了许多的困难，在解决问题的过程中，需要对整个业务的执行流程了然于心，使得我需要不断的去思考，找到问题的所在，从而使我的综合能力得到提升。同时还会运到一些不可预估的程序错误，使我不得不去网上查阅相关资料，也就锻炼了我检索的能力与自学的能力。

不足：

由于是第一次设计数据库，缺乏经验，使得在刚开始设计时，出现了一些漏洞以及不足，导致在编写代码时出现困难。当我试图去修改一个表时，又会出现动一发而牵动全身的尴尬情况，这也是数据库表结构设计存在不合理的导致的问题，受到自身能力和时间的限制，到最后也没有实现理想中的应用。