数据库课程说明书

2017-2018 学年第 二 学期

学　　院：	大数据学院
专　　业：	物联网工程
姓　　名：	王天锐　　学 号：1607094155
课程设计题目：	火车票预售系统
起 迄 日 期：	2018 年 6 月 12 日 ~ 2018 年 6 月 29 日
课程设计地点：	主楼六层物联实验室
指 导 教 师：	于一 柴锐 甄成方 樊彩霞
学科部副主任：	王东

下达任务书日期: 2018 年 6 月 12 日

1 系统设计

1.1 设计目的

乘坐火车是我们生活中几乎不可缺少的一件事儿，每天都会有各种各样的火车班次发布与被预定。针对这个火车票预售的环节我设计了一个火车票预售系统，为购票用户与卖票管理人员之间搭建平台。让我们的用户能够通过该软件对管理人员发布的航班进行预购与查询。另一方面也可以加强我们的管理人员对班次信息与乘客的管理与查询。

本系统的根本目的是让管理人员能够发布与查询班次信息、查询乘客信息等；用户可以通过该系统对班次进行预购、自己购票记录的查询等。

1.2 需求分析

1.2.1 信息要求

该系统主要记录用户、班次、火车、银行卡之间的关系

(1)用户分为管理员与购票用户：

1)售票管理员信息：管理员编号、管理员名字、管理员电话。

2)购票用户信息：身份证号、电话号码、银行卡号。

(2)班次信息：

班次号、火车号、出发地点、目的地、出发时间、到达时间。

(3)火车信息：

火车号、火车节数、座位数、各种座位票价、火车车速。

(4)银行卡信息：

银行卡号、余额、持有人身份证号。

(5)身份证信息：

身份证号、姓名、性别、所有者

(6)车票信息：

车票号、班次号、座位号、乘客身份证号、车票价钱、车厢数

1.2.2 处理要求

能够正确、高效、迅速地完成所有操作。

一个管理员可以管理多个班次、一个用户可以多次订购不同时间段的车票

1.2.3 安全性与完整性需求

(1) 安全性

1) 该系统需要用户进行账号的注册与登陆。

2) 通过对不同的用户种类的检测来给予不同的权限与界面。

3) 用户登陆自己账号后只能查询自己用户名下身份证的购票信息与个人信息。

4) 用户不可对班次、火车等信息进行修改。

5) 售票员能对班次信息进行修改与查询，对于用户信息只能查询不能修改。

(2) 完整性

1) 实体完整性

手机号、班次号、火车号、银行卡号、身份证号、车票号分别为用户、班次、火车、银行卡、身份证、车票的主码。

2) 参照完整性

班次号中的火车号为火车表的主码、银行卡号中所有者号码为用户的主码、车票中的火车号与班次号分别为火车表与班次表的主码。以上外码要么为空要么是参照表中已有数据。

3) 用户定义完整性

性别只能是男女、车票钱不能为空、车速不能为空、班次目的地与出发地不能为空、用户类型只能是 0 与 1：0 表示普通用户、1 表示管理员用户。用户名字不能为空。

1.3 开发和运行环境选择

开发工具：

前台开发语言为 Java，后台数据库为 SQL SERVER2017。

运行环境：

Java 1.8 版本

Windows2000 以上

2 数据库设计

2.1 数据库概念设计

本系统中包括六个实体：身份证实体、银行卡实体、用户实体、车票实体、火车实体、班次实体，根据需求需求分析的结果以上六个实体嗦对应的 ER 属性图如图 2.1、2.2、2.3、2.4、2.5、2.6。

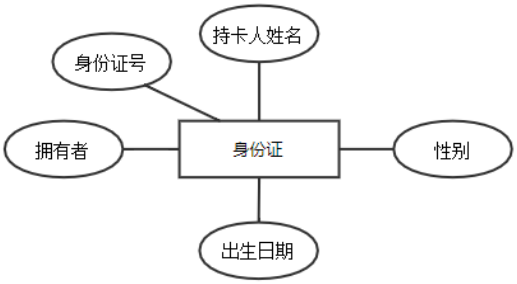

图 2.1 身份证实体属性图

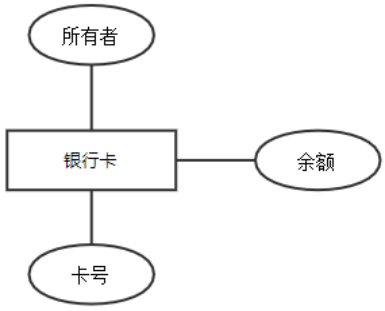

图 2.2 银行卡实体属性图

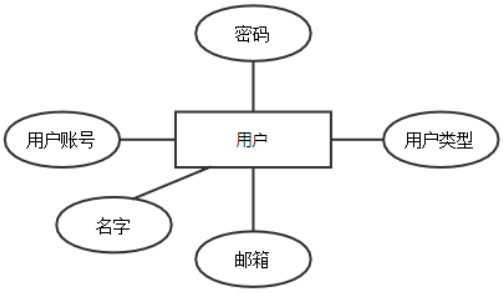

图 2.3 用户实体属性图

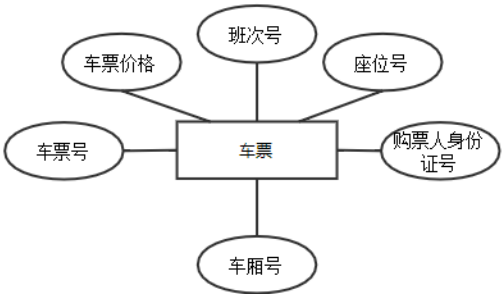

图 2.4 车票实体属性图

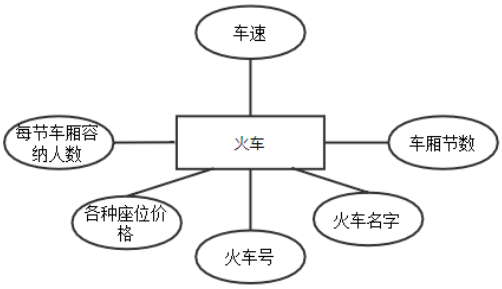

图 2.5 火车实体属性图

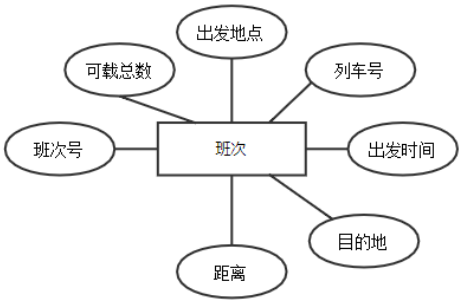

图 2.6 班次实体属性图

系统整体 E-R 图为图 2.7

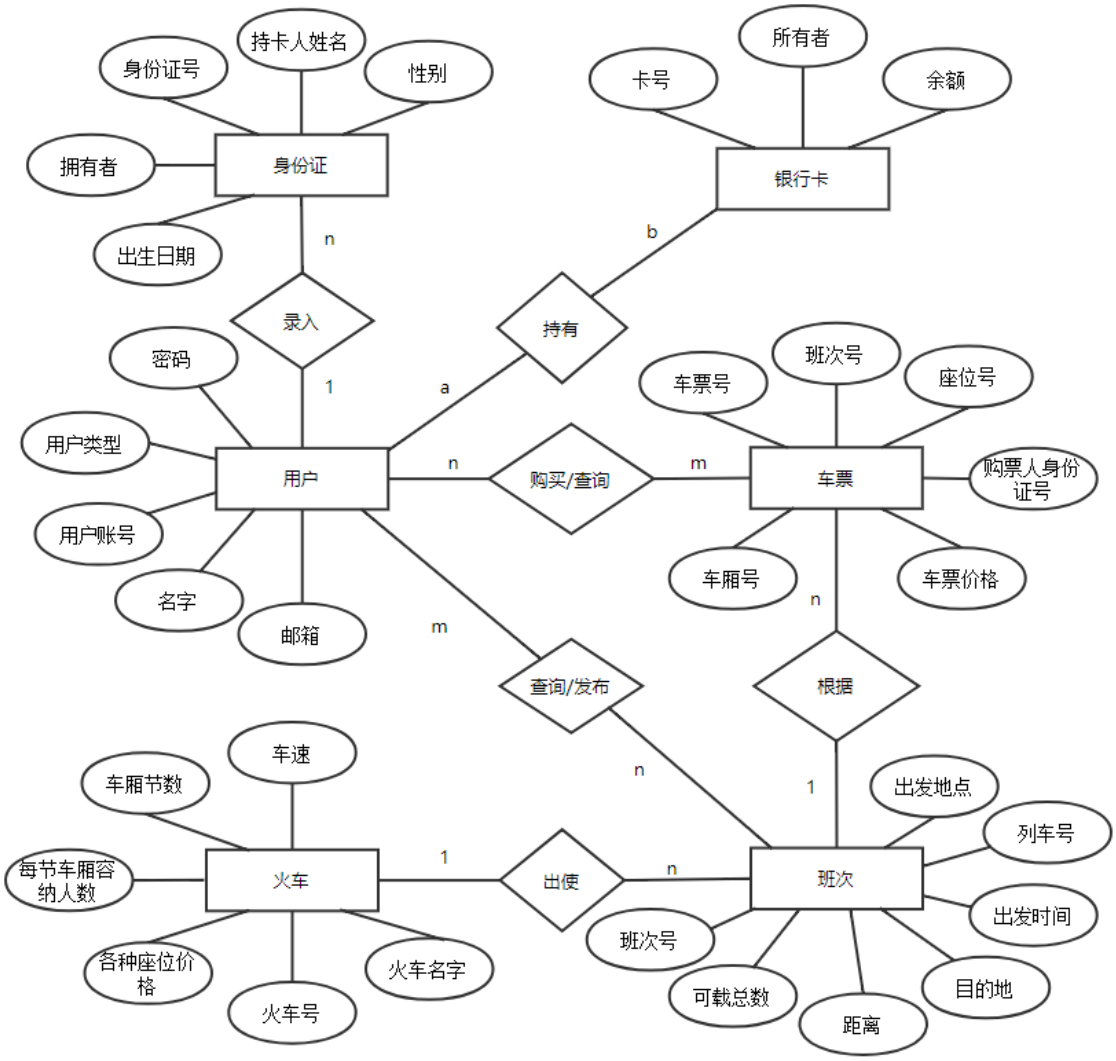

图 2.7 火车票预售系统整体 E-R 图

2.2 数据库逻辑设计

根据 E-R 图向关系模型的转换原则，一个实体型转换为一个关系模式，实体的属性就是关系的属性。因此按照图 2.7 中的整体 E-R 图，本数据库系统中应包括六张表：身份证、银行卡、用户、车票、火车、和班次。

表 2.1 车票表(Ticket)

中文含义	字段名称	数据类型	长度	能否为空	备注
车票号	TicketNumber	Char	20	No	主码
班次号	TClassNumber	Char	20	No	主码+外码，指向班次表中的主码
座位号	TSeatNumber	Char	20	No	
购票人身份证号	TIdCardNumber	Char	20	No	主码+外码，指向身份证表中主码
价钱	TicketPrice	Float		No	

表 2.2 用户表(UserTrain)

中文含义	字段名称	数据类型	长度	能否为空	备注
帐号	UserPhoneNumber	Char	20	No	主码（帐号为手机号）
密码	UserPassWord	Char	20	No	
邮箱	UserEmail	Char	20	No	用于忘记密码时登录
用户名	UserName	Char	20	No	
用户类别	UserType	Int	2	No	0 为普通用户，1 为管理员用户

表 2.3 火车表(Train)

中文含义	字段名称	数据类型	长度	能否为空	备注
火车号	TrainNumber	Char	20	No	主码
火车名字	TrainName	Char	20	No	
车厢数	TrainComparNumber	Int		No	
车厢容纳量	TrainCapacity	Int		No	
火车速度	TrainSpeed	Float		No	
一等座价钱	TrainFirstPrice	Float		No	
二等座价钱	TrainSecondPrice	Float		No	
卧铺价钱	TrainBedPrice	Float		No	

表 2.4 银行卡表（Credit Card）

中文含义	字段名称	数据类型	长度	能否为空	备注
银行卡号	CreditCardNumber	Char	20	No	主码
所有者	CreditOwner	Char	20	No	外码，指向用户表中的帐号
余额	CreditBalace	float	20	No	

表 2.5 班次表(Classes)

中文含义	字段名称	数据类型	长度	能否为空	备注
班次号	ClassNumber	Char	20	No	主码
火车号	ClassTrainNumber	Char	20	No	外码，指向火车表的主码
出发地点	ClassDepaturePlace	Char	20	No	
目的地	ClassGoalPlace	Char	20	No	
出发时间	ClassDepatureTime	Time		No	
可载总数	ClassPeopleNumber	Int		No	
距离	ClassDistance	Float		No	

表 2.6 身份证表(IdCard)

中文含义	字段名称	数据类型	长度	能否为空	备注
身份证号	IdCardNumber	Char	20	No	主码
持卡人姓名	IdCardName	Char	20	No	
性别	IdCardSex	Char	20	No	只能写男或女
出生日期	IdCardBirthday	Date		No	
归属用户	IdCardOwnerUser	Char	20	No	外码，指向用户表中的帐号

3 火车票预售系统详细设计

3.1 功能概述

售票管理员系统主要包含如下几个功能：

(1) 发布与删除班次及其相关信息

(2) 查看班次详细信息

(3) 通过班次号对班次进行详细查询

(4) 通过身份证号码查询乘车人信息

用户系统主要包含如下几个功能：

(1) 通过地点、时间查询班次信息

(2) 通过班次号查询班次详细信息

(3) 添加和删除信用卡

(4) 添加、修改和删除乘车人信息

(5) 查询得到班次后可以为乘车人购买车票

(6) 查看自己用户的购票记录

整体功能描述为图 3.1 所示

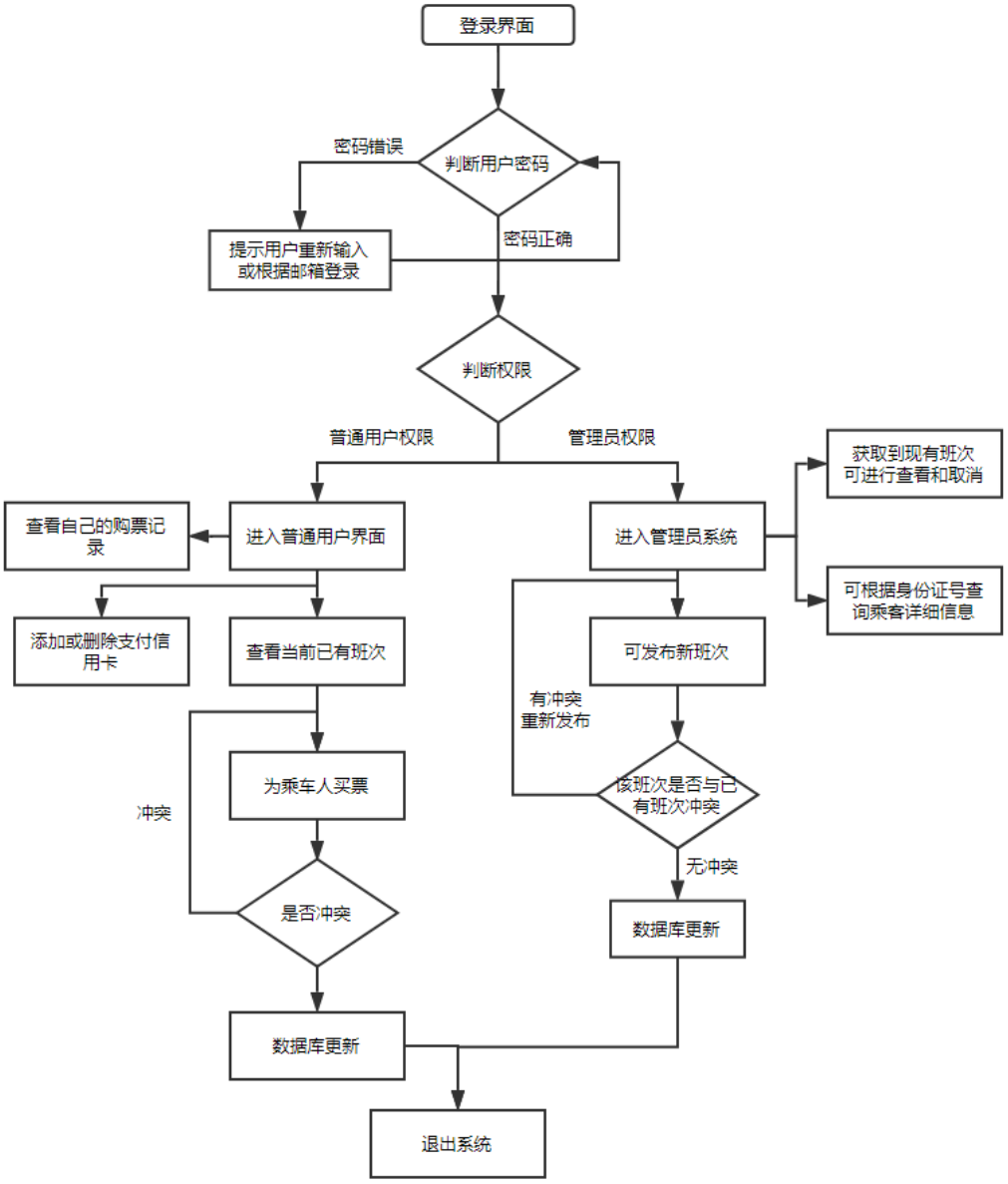

图 3.1 火车票预售系统整体功能流程图

3.2 火车票预售系统详细设计

3.2.1 界面设计

(1) 管理员界面设计

在使用管理员账号登录过后，会出现管理员主界面。在界面最上方有初始地点选择和时间选择控件，可以通过时间和地点的详细选择来针对性地发布班次信息，在点击发布后会弹出火车号填写对话框，在此处填写火车号以完成班次发布的目的，如图 3.2.在发布下方有一个查看按钮,通过在左边列表中选择想查看的班次再点击该按钮即可查看到详细的班次信息如图 3.3。

在查看下方还有一个删除按钮，管理员可以通过选取班次点击该按钮来完成取消班次的操作。

图 3.2 班次发布示意图

图 3.3 查看班次信息示意图

(2) 用户系统界面设计

在使用用户权限帐号登录过后，会进入相应的用户界面，用户界面分为四个 fragment，第一个 fragment 通过站点和时间来查询班次列表，在查询到班次后可以通过购买按钮对车票进行购买，购买时需要选择乘车人、勾选座位信息和选择支付的信用卡，如图 3.4 和图 3.5 所示。

图 3.4 购票选择乘车人

图 3.5 购票选择信用卡支付

在第二个 fragment 里用户可以通过班次号来查询班次信息与购票，如图 3.6 与 3.7，在点击查询后按钮会自动变成购票，后续购票操作和第一个 fragment 一样。

图 3.6 输入班次号查询

图 3.7 查询后可以直接购买

在第三个 fragment 里用户可以查询到自己所有的购票记录，可以通过查看详细按钮进行查看，还可以通过退票按钮进行退票操作，如图 3.8 和图 3.9 所示。

图 3.8 查看车票详细信息

图 3.9 退票操作

最后是用户信息界面，用户可以在此处看到自己用户名以及邮箱号，如图 3.10。用户可以在此界面增、删、改常用乘车人信息，如图 3.11 所示。也可以添加和删除信用卡，如图 3.12 和图 3.13 所示。

图 3.10 个人信息界面

图 3.11 乘车人信息增改界面

图 3.12 信用卡增改界面

图 3.13 删除操作界面

3.2.2 功能实现

整体功能是通过两个 Activity 与四个 Fragment 来实现，具体实现功能顺序为图 3.17 所示。本系统大量使用 Dialog 来对详细信息填写进行管理。

功能实现过程中使用到 UI 的类有 17 个，其中三个抽象父类用于限制所有界面的业务处理与逻辑执行顺序，其余 14 个子类用于详细界面以及功能的实现，整体界面代码框架如图 3.18 所示。注：Dialog 是在 Fragment 中使用，Fragment 是放置到底层 Activity 使用。

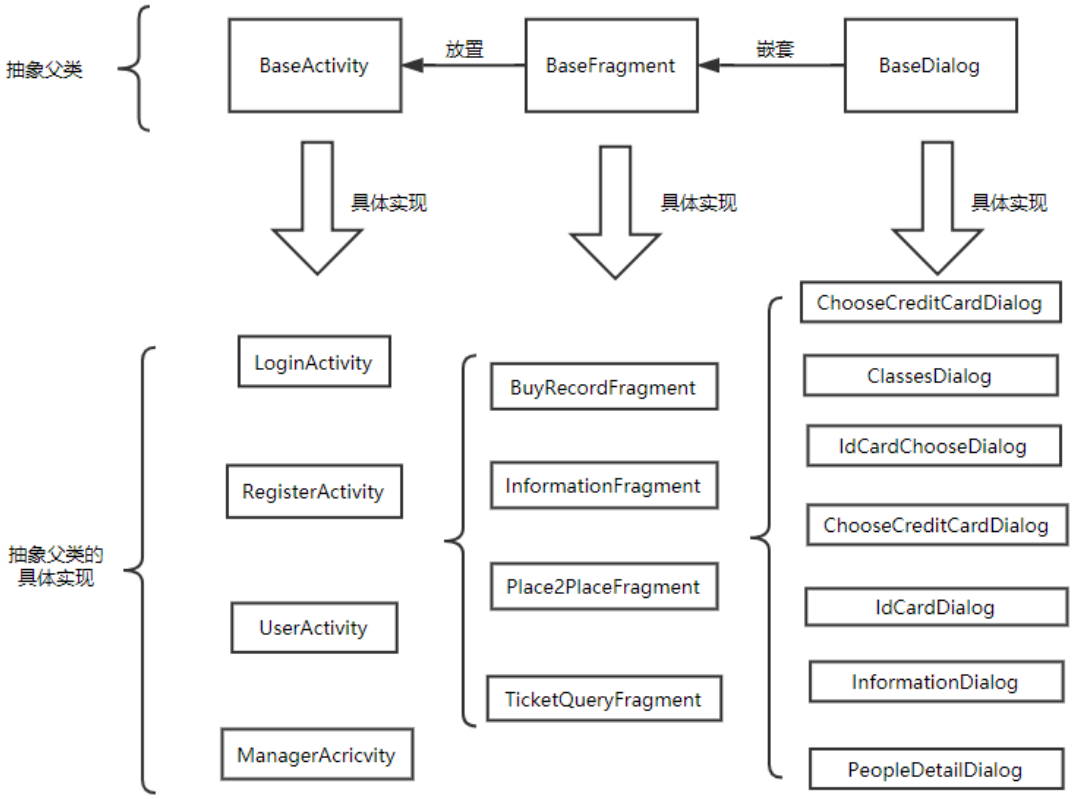

图 3.18 整体界面代码框架

3.2.3 主要代码

(1) 所有 SQL 逻辑操

```
public class SqlHelper {

    private Connection connection = null;

    String INSERT_SQL = "INSERT INTO table VALUES(x);";

    String QUERY_SQL = "SELECT sth FROM table ";

    /**

     * 构造函数，实例化 Connection

     * @param sqlUser

     */

    public SqlHelper(SqlUser sqlUser) {

        this.connection = sqlUser.getConnection();

    }

    /**

     * 查询列车

     * @param trainNumber

     */

    public Train queryTrain(String trainNumber) {

        String sql = "SELECT
    TrainNumber,TrainName,TrainCompartmentNumber,TrainCapacityOfCompartment,Train
    Speed,firstPrice,secondPrice,bedPrice FROM Train WHERE TrainNumber = ?";

        Train train = new Train();
```

```
        train.setTrainNumber(" ");
        try {
            sql = sql.replace("?", formatString(trainNumber));
            ResultSet resultSet = executeQuery(sql);
            System.out.println(sql);
            while (resultSet.next()) {
                train = new Train(
                    resultSet.getString("TrainNumber").trim(),
                    resultSet.getString("TrainName").trim(),
                    resultSet.getInt("TrainCompartmentNumber"),
                    resultSet.getInt("TrainCapacityOfCompartment"),
                    resultSet.getFloat("TrainSpeed"),
                    resultSet.getFloat("firstPrice"),
                    resultSet.getFloat("secondPrice"),
                    resultSet.getFloat("bedPrice") ); }
    } catch (SQLException e) {
            e.printStackTrace();
    }
    return train;
}

    /**
```

```
 * 添加班次
 * @param trainClass
 */
public boolean addClass(TrainClass trainClass) {
    String sql = "INSERT INTO Classes VALUES(cn,ctn,cdp,cgp,cdis,cde,cpn,time)";
    sql = sql.replace("cn", formatString(trainClass.getClassNumber()));
    sql = sql.replace("ctn", formatString(trainClass.getTrainNumber()));
    sql = sql.replace("cdp", formatString(trainClass.getDepaturePlace()));
    sql = sql.replace("cgp", formatString(trainClass.getGoalPlace()));
    sql = sql.replace("cdis", trainClass.getDinstance() + "");
    sql = sql.replace("cde",
    formatString(trainClass.getDepatureDay().toString()));
    sql = sql.replace("cpn", trainClass.getPassengerNumber() + "");
    sql = sql.replace("time", formatString(trainClass.getTime() + ""));
    return executeUpdate(sql);
}
/**
 * 管理员删除班次
 * @param trainClass
 * @return
 */
public Boolean deleteClasses(TrainClass trainClass) {
```

```
        String sql1 = "DELETE FROM Ticket WHERE TicketClassesNumber = " +
        formatString(trainClass.getClassNumber());
        String sql2 = "DELETE FROM Classes WHERE ClassesNumber = '" +
        trainClass.getClassNumber() + "'";
        executeUpdate(sql1);
        return executeUpdate(sql2);
    }
    /**
     * 查询班次
     * @return
     */
    public List<TrainClass> queryClasses() throws SQLException {
        ResultSet resultSet;
        List<TrainClass> mList = new ArrayList<>();
        String sql = "SELECT
        ClassesNumber,ClassesTrainNumber,ClassesDepaturePlace,ClassesGoalPlace,Cl
        assesDistance," +"ClassesDepatureTime,ClassesPassengerNumber,DepatureTime
        FROM Classes ORDER BY ClassesDepatureTime DESC";
        resultSet = executeQuery(sql);
        while (resultSet.next()) {
            mList.add(new TrainClass(
                    resultSet.getString("ClassesNumber").trim(),
                    resultSet.getString("ClassesTrainNumber").trim(),
```

```
                resultSet.getString("ClassesDepaturePlace").trim(),
                resultSet.getString("ClassesGoalPlace").trim(),
                resultSet.getFloat("ClassesDistance"),
                resultSet.getDate("ClassesDepatureTime"),
                resultSet.getInt("ClassesPassengerNumber"),
                resultSet.getString("DepatureTime").trim()));
        }
        return mList;
    }
    /**
     * 按班次号查询
     * @param classesNumber
     * @return
     */
    public TrainClass queryClasses(String classesNumber) throws SQLException {
        String sql = "SELECT
        ClassesNumber,ClassesTrainNumber,ClassesDepaturePlace,ClassesGoalPlace,Cl
        assesDistance,ClassesDepatureTime,ClassesPassengerNumber,DepatureTime FROM
        Classes WHERE ClassesNumber = '" + classesNumber + "'";
        ResultSet resultSet = executeQuery(sql);
        TrainClass re = new TrainClass();
        re.setClassNumber(" ");
        while (resultSet.next()) {
```

```
            re = new TrainClass(
                    resultSet.getString("ClassesNumber").trim(),
                    resultSet.getString("ClassesTrainNumber").trim(),
                    resultSet.getString("ClassesDepaturePlace").trim(),
                    resultSet.getString("ClassesGoalPlace").trim(),
                    resultSet.getFloat("ClassesDistance"),
                    resultSet.getDate("ClassesDepatureTime"),
                    resultSet.getInt("ClassesPassengerNumber"),
                    resultSet.getString("DepatureTime").trim()
            );
        }
        return re;
    }
    /**
     * 按起终点查询
     * @param startplace
     * @param goalplace
     * @return
     * @throws SQLException
     */
    public List<TrainClass> queryClasses(String startplace, String goalplace, Date
    depatureDay) throws SQLException {
```

```
        String sql = "SELECT
        ClassesNumber,ClassesTrainNumber,ClassesDepaturePlace,ClassesGoalPlace,Cl
        assesDistance," +"ClassesDepatureTime,ClassesPassengerNumber,DepatureTime
        FROM Classes WHERE ClassesDepaturePlace = '"+ startplace + "' AND
        ClassesGoalPlace = '" + goalplace + "' AND ClassesDepatureTime = '" +
        depatureDay + "'";
        ResultSet resultSet;
        resultSet = executeQuery(sql);
        List<TrainClass> mList = new ArrayList<>();
        while (resultSet.next()) {
            mList.add(new TrainClass(
                    resultSet.getString("ClassesNumber").trim(),
                    resultSet.getString("ClassesTrainNumber").trim(),
                    resultSet.getString("ClassesDepaturePlace").trim(),
                    resultSet.getString("ClassesGoalPlace").trim(),
                    resultSet.getFloat("ClassesDistance"),
                    resultSet.getDate("ClassesDepatureTime"),
                    resultSet.getInt("ClassesPassengerNumber"),
                    resultSet.getString("DepatureTime").trim()
            ));
        }
        return mList;
    }
```

```
/**
 * 更新班次信息
 * @param trainClass
 * @return
 */
public boolean updateClassesPassengerNumber(TrainClass trainClass) {
    String sql = "UPDATE Classes SET ClassesPassengerNumber = " +
    trainClass.getPassengerNumber()+ " WHERE ClassesNumber = " +
    trainClass.getClassNumber();
    System.out.println(sql);
    return executeUpdate(sql);
}

/**
 * 买票
 * @param trainClass
 * @param idCard
 * @return
 */
public boolean buyTicket(TrainClass trainClass, IdCard idCard, int nearWindow,
int seatType, CreditCard creditCard) throws SQLException {
    Ticket ticket = new Ticket();
    ticket.setTicketNumber(" ");
```

```
Train train = queryTrain(trainClass.getTrainNumber());
TrainClass trainClassQuery = queryClasses(trainClass.getClassNumber());
int passengerNumber = trainClassQuery.getPassengerNumber();
if (passengerNumber + 1 >= train.getCapacityOfCompartment() *
train.getCompartmentNumber()) {
    return false;
}
int seatNumber = passengerNumber;
if (nearWindow == 1) {
    ConstantsUtils.temp++;
    seatNumber = ConstantsUtils.temp * 4;
    seatNumber += 1;
    while (seatNumber % 4 != 0 && seatNumber % 4 != 1) {
        System.out.println(seatNumber);
        seatNumber++;
    }
} else {
    seatNumber += 1;
    while (seatNumber % 4 == 0 || seatNumber % 4 == 1) {
        seatNumber++;
    }
}
```

```
//更新班次
trainClass.setPassengerNumber(passengerNumber + 1);
boolean update = updateClassesPassengerNumber(trainClass);
Seat seat = null;
switch (seatType) {
    case 0:
        seat = new Seat(
                train.getTrainNumber(),
                seatNumber + "",
                train.getFirstPrice()
        ); break;
    case 1:
        seatNumber += 120;
        seat = new Seat(
                train.getTrainNumber()+seatNumber + "",
                train.getSecondPrice()
        ); break;
    case 2:
        seatNumber += 240;
        seat = new Seat(
                train.getTrainNumber(),
                seatNumber + "",
```

```
                train.getBedPrice()

        ); Break; }

ticket.setClassNumber(trainClass.getClassNumber());

ticket.setTicketNumber(trainClass.getClassNumber() +

idCard.getIdCardNumber().substring(1, 5));

ticket.setSeatNumber(seatNumber + "");

ticket.setIdCardNumber(idCard.getIdCardNumber());

ticket.setTicketTrainNumber(train.getTrainNumber());

ticket.setTicketPrice(seat.getPrice());

ticket.setCompartment(seatNumber / 40 + 1);

String sql = "INSERT INTO Ticket VALUES(" +

formatString(ticket.getTicketNumber()) + ","

        + formatString(ticket.getClassNumber()) + "," +

formatString(ticket.getSeatNumber()) + ","+

formatString(ticket.getIdCardNumber()) + "," + ticket.getTicketPrice() + ","

+ ticket.getCompartment()+ ")";

boolean insert = false;

if (useCreditCard(creditCard, ticket.getTicketPrice())) {

    insert = executeUpdate(sql);

} else {

    insert = false;

}

return insert && update;
```

```
}

/**
 * 查票
 * @param user
 * @return
 */
public List<Ticket> queryTickets(User user) {
    String sql = "SELECT * FROM Ticket WHERE TicketIdCardNumber = ";
    List<IdCard> idCards = null;
    List<Ticket> tickets = new ArrayList<>();
    try {
        idCards = queryIdCard(user);
        for (IdCard idCard : idCards) {
            sql = sql + formatString(idCard.getIdCardNumber());
            ResultSet resultSet = executeQuery(sql);
            System.out.println(sql);
            while (resultSet.next()) {
                tickets.add(new Ticket(
                        resultSet.getString(1).trim(),
                        resultSet.getString(2).trim(),
                        resultSet.getString(3).trim(),
                        resultSet.getString(4).trim(),
```

```
                        resultSet.getFloat(5),
                        resultSet.getInt(6)
                ));
            }
            sql = "SELECT * FROM Ticket WHERE TicketIdCardNumber = ";
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return tickets;
}
/**
 * 退票
 * @param ticket
 * @return
 */
public boolean returnTheTicket(Ticket ticket) {
    String sql = "DELETE FROM Ticket WHERE TicketTrainNumber = " +
    formatString(ticket.getTicketNumber());
    try {
        TrainClass trainClass = queryClasses(ticket.getClassNumber());
        trainClass.setPassengerNumber(trainClass.getPassengerNumber() - 1);
```

```
            updateClassesPassengerNumber(trainClass);
            useCreditCard(queryCreditCard(Main.user).get(0),
            -(ticket.getTicketPrice()));
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return executeUpdate(sql);
    }
    /**
     * 查询车票记录
     * @param idCard
     * @return
     */
    public List<Ticket> queryTicketById(IdCard idCard) throws SQLException {
        String sql = "SELECT * FROM Ticket WHERE TicketIdCardNumber = " +
        formatString(idCard.getIdCardNumber())+ " ORDER BY TicketClassesNumber
        DESC";
        List<Ticket> mList = new ArrayList<>();
        ResultSet resultSet = executeQuery(sql);
        while (resultSet.next()) {
            mList.add(new Ticket(
                    resultSet.getString(1).trim(),
                    resultSet.getString(2).trim(),
```

```
                resultSet.getString(3).trim(),
                resultSet.getString(4).trim(),
                resultSet.getFloat(5),
                resultSet.getInt(6)));
        }
        return mList;
    }
    /**
     * 查询身份证
     * @param id
     */
    public IdCard queryIdCard(String id) throws SQLException {
        String sql = "SELECT IdCardNumber,IdCardName,IdCardSex,IdCardBirthday FROM
        IdCard WHERE IdCardNumber = "+ formatString(id);
        IdCard idCard = new IdCard();
        idCard.setIdCardNumber(" ");
        ResultSet resultSet = executeQuery(sql);
        while (resultSet.next()) {
            idCard.setIdCardNumber(resultSet.getString("IdCardNumber").trim());
            idCard.setName(resultSet.getString("IdCardName").trim());
            idCard.setBirthday(resultSet.getDate("IdCardBirthday"));
            idCard.setSex(resultSet.getString("IdCardSex").trim());
```

```
        }
        return idCard;
    }
    /**
     * 更新类 sql 语言执行者
     * @param sql
     * @return
     */
    private boolean executeUpdate(String sql) {
        try {
            Statement statement = connection.createStatement();
            int result = statement.executeUpdate(sql);
            if (result > 0) {
                  return true;
            } else {
                statement.close();
                return false;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
```

```
    }

    /**
     * 查询类 sql 语句执行者
     * @param sql
     */
    private ResultSet executeQuery(String sql) {
        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            return resultSet;
        } catch (SQLException e) {
            System.out.println("SqlHelper :SQL query 执行失败 " + e.toString());
            e.printStackTrace();
            return null;
        }
    }
    //为字符串添加单引号
    private String formatString(String input) {
        System.out.println("'" + input + "'");
        return "'" + input + "'";
    }
}
```

4 系统测试

测试发布班次如图 4.1 到 4.2 所示。

图 4.1 发布班次示意图

图 4.2 发布班次示意图

到数据库查询如图 4.3 所示：

结果 消息

	ClassesNumber	ClassesTrainNumber	ClassesDepaturePlace	ClassesGoalPlace	ClassesDistance	ClassesDepatureTime	ClassesPassengerNumber	DepatureTime
7	100171	WANG-1001	海南	成都	1230	2018-07-02	0	20:25
8	100172	WANG-1001	海南	上海	4512	2018-07-03	0	11:49
9	100183	WANG-1001	深圳	太原	3456	2018-07-03	1	11:49
10	2002102	WANG-2002	厦门	上海	8461	2018-07-03	2	11:49
11	200213	WANG-2002	成都	太原	1321	2018-07-03	0	11:49
12	200225	WANG-2002	上海	西藏	7894	2018-07-03	0	09:20
13	200241	WANG-2002	北京	成都	4212	2018-07-03	0	11:42
14	200271	WANG-2002	海南	成都	1230	2018-07-02	0	20:25
15	300310	WANG-3003	上海	成都	7123	2018-06-21	1	11:32
16	3003102	WANG-3003	厦门	上海	8461	2018-07-03	0	11:49
17	300313	WANG-3003	成都	太原	1321	2018-07-03	1	09:20
18	300325	WANG-3003	上海	西藏	7894	2018-07-03	1	15:20
19	300341	WANG-3003	北京	成都	4212	2018-07-02	0	20:25
20	400413	WANG-4004	成都	太原	1321	2018-07-03	0	09:20
21	400441	WANG-4004	北京	成都	4212	2018-07-02	0	20:25
22	4004610	WANG-4004	天津	厦门	9974	2018-06-30	0	15:26
23	400462	WANG-4004	海南	太原	3564	2018-06-29	1	07:35

图 4.3 查询班次结果

用户买票操作测试如图 4.6 与图 4.7 所示

图 4.6 购票操作示意图

图 4.7 购票操作示意图

数据库查询结果如图 4. 8 所示

	TicketTrainNumber	TicketClassesNumber	TicketSeatNumber	TicketIdCardNumber	TicketPrice	compartment
1	1001131012	100113	5	510122199711015522	30	1
2	10011322199	100113	4	510122199711015511	30	1
3	10011322199	100113	4	510122199711015522	30	1
4	10012509415	100125	1	1607094154	30	1
5	10012509415	100125	244	1607094156	110	7
6	10018309415	100183	121	1607094155	40	4
7	200210209415	2002102	241	1607094155	120	7
8	20021021012	2002102	5	510122199711015511	40	1
9	30032556898	300325	1	123456898	20	1
10	40046222199121	400462	121	510122199711015511	55	4

图 4.8 买票信息入库示意图

退票操作测试如图 4.9 所示

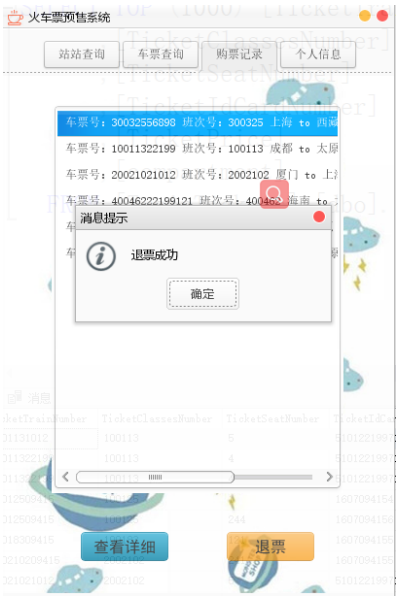

图 4.9 退票操作示意图

数据库查询结果如图 4.10 所示

	TicketTrainNumber	TicketClassesNumber	TicketSeatNumber	TicketIdCardNumber	TicketPrice	compartment
1	10011322199	100113	4	510122199711015511	30	1
2	10011322199	100113	4	510122199711015522	30	1
3	10012509415	100125	1	1607094154	30	1
4	10012509415	100125	244	1607094156	110	7
5	10018309415	100183	121	1607094155	40	4
6	200210209415	2002102	241	1607094155	120	7
7	20021021012	2002102	5	510122199711015511	40	1

图 4.10 退票后数据库查询结果

5 个人体会

三个星期的时间非常快就过去了，这三个星期不敢说自己有多大的进步，获得了多少知识，但起码是了解了项目开发的部分过程。虽说上过数据库等相关的课程，但是没有亲身经历过相关的设计工作细节。这次课设证实提供了一个很好的机会。通过这次的系统设计，我在很多方面都有所提高。综合运用所学知识的理论知识实际训练从而培养和提高学生独立工作的能力，巩固所学的知识，掌握系统程序的编排和运行，使自己的独立思考能力有了显著提高。从各种文档的阅读到开始的需求分析、概念结构设计、逻辑结构设计、物理结构设计。亲身体验了一回系统的设计开发过程。很多东西书上写的很清楚，貌似看着也很简单，思路非常清晰。但真正需要自己想办法去设计一个系统的时候才发现其中的难度。经常做到后面突然就发现自己一开始的设计有问题，然后又回去翻工，在各种反复中不断完善自己的想法。我们学习并应用了SQL 语言，对数据库的创建、修改、删除方法有了一定的了解，通过导入表和删除表、更改表学会了对于表的一些操作，为了建立一个关系数据库信息管理系统，必须得经过系统调研、需求分析、概念设计、逻辑设计、物理设计、系统调试、维护以及系统评价的一般过程，为毕业设计打下基础。很多事情不是想象中的那么简单的，它涉及到的各种实体、属性、数据流程、数据处理等等。很多时候感觉后面的设计根本无法继续，感觉像是被前面做的各种图限制了。在做关系模型转换的时候碰到有些实体即可以认为是实体又可以作为属性，为了避免冗余，尽量按照属性处理了。不管做什么，我们都要相信自己，不能畏惧，不能怕遇到困难，什么都需要去尝试，有些你开始认为很难的事在你尝试之后你可能会发现原来并没有你以前觉得的那样，自己也是可以的。如果没有自信，没有目标，没有信心就不可能把事情做好，当其他人都在迷茫的时候，自己一定要坚信目标。有一个这样的感觉就是课程设计学到的东西比一个学期都多。

参考文献

[1] CSDN 论坛、简书论坛、知乎论坛

[2] 毕广吉.Java 程序设计实例教程[M]. 北京：冶金工业出版社，2007 年

[3] 肖成金，吕冬梅。 Java 程序开发数据库与框架应用[J]. 科技展望，2017,05:19.

[4] 吕锋,梅细燕,周晓东;基于 JDBC 的数据库管理及其应用[J];武汉理工大学学报;2002 年 10 期

[5] 姚永一，SQL Server 数据库实用教程，北京：电子工业出版社,2010.

[6] 高云，崔艳春，SQL Server 2008 数据库技术实用教程，北京：清华大学出版社，2011

[7] 何玉洁，梁琦，数据库原理与应用（第二版），北京：机械工业出版社，2011

[8] 壮志剑，数据库原理与 SQL Server，北京：高等教育出版社，2008

[9] 杜丁超.计算机软件 Java 编程特点及其技术分析

[10] 萧仁惠,陈锦辉. JDBC 数据库程序设计[J].北京:中国铁道出版社,2004.