

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 机票预订系统 |
| 姓 名： | 杨雨鑫 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1806 |
| 学 号： | U201814655 |
| 指导教师： | 胡侃 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2021年 7月 7日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

目 录

[1 课程任务概述 1](#_Toc31176)

[2 SQL练习部分 2](#_Toc20800)

[2.1 任务要求 2](#_Toc18899)

[2.2 完成过程 4](#_Toc9240)

[2.3任务总结 13](#_Toc18049)

[3软件功能学习部分 13](#_Toc5136)

[3.1 任务要求 13](#_Toc29621)

[3.2 完成过程 13](#_Toc1983)

[3.3任务总结 15](#_Toc20196)

[4 综合实践任务 16](#_Toc23321)

[4.1系统设计目标 16](#_Toc3903)

[4.2 需求分析 16](#_Toc12143)

[4.3 总体设计 18](#_Toc1068)

[4.4 数据库设计 19](#_Toc25725)

[4.5 详细设计与实现 22](#_Toc27805)

[4.6 系统测试 30](#_Toc19187)

[4.7 系统设计与实现总结 37](#_Toc13827)

[5 课程总结 39](#_Toc25613)

[附录 40](#_Toc18413)

# 1 课程任务概述

**实验一 Sql练习部分**

基于educoder平台的covid19数据库操作

1.建表

2.数据更新

3.数据查询

4.了解系统的查询性能分析功能（选做）

5. DBMS函数及存储过程和事务（选做）

**实验二 软件功能学习部分**

1. 数据备份操作
2. 设置权限操作

**实验三 数据库应用系统设计与实现**

选题：机票预订系统

1. 系统功能的基本要求
2. 每个航班信息的输入。
3. 每个航班的坐位信息的输入；
4. 当旅客进行机票预定时，输入旅客基本信息，系统为旅客安排航班，打印取票通知和帐单；
5. 旅客在飞机起飞前一天凭取票通知交款取票；
6. 旅客能够退订机票；
7. 能够查询每个航班的预定情况、计算航班的满座率。
8. 包含事务（包含commit，rollback），存储过程/触发器，视图，函数。
9. 在程序中需要体现SQL和编程语言的结合。
10. 数据库要求：在数据库中至少应该包含下列数据表：
11. 航班信息表；
12. 航班坐位情况表；
13. 旅客订票信息表；
14. 取票通知表；
15. 帐单。

设计一个B/S或C/S模式的系统实现上述功能。

# SQL练习部分

## 2.1 任务要求

假设在某个区域内的所有地点都存储在地点表中。该地区中的所有人员信息存储在人员表中。根据收集到的人员行程建立了疫情期间某个地区的人员行程表。

该地区的部分人员进行了核酸检测，检测结果保存在诊断表中。根据诊断表中的检测结果，对新冠确诊和无症状感染者的密切接触者进行隔离，密切接触者的信息存储到密切接触表中，隔离信息存储到隔离表中。隔离地点共设置了4个，并且每个隔离地点都有容量限制，隔离点的信息存储在隔离地点表中。

* + 1. **建表**

**人员表【人员编号，姓名，电话】**

person (id int, fullname char(20), telephone char(11))

主码为人员编号

**地点表【地点编号，地点名称】**

location (id int, location\_name char(20))

主码为地点编号

**行程表【行程编号，人员编号，所在地点编号，开始时间，结束时间】**

itinerary (id int, p\_id int, loc\_id int, s\_time datetime, e\_time datetime)

主码为行程编号，

**诊断表【诊断编号，人员编号，诊断日期，诊断结果】**

diagnose\_record (id int, p\_id int, diagnose\_date datetime, result int)

主码为诊断编号，诊断结果包括：1：新冠确诊，2：无症状感染者，3：正常

**密切接触者表【密切接触编号，被接触者编号，接触日期，接触地点，病例人员编号】**

close\_contact (id int, p\_id int, contact\_date datetime, loc\_id int, case\_p\_id int)

主码为密切接触编号

**隔离表【隔离编号，被隔离人员编号，开始隔离日期，结束隔离日期，隔离地点编号，隔离状态】**

isolation\_record (id int, p\_id int, s\_date datetime, e\_date datetime, isol\_loc\_id int, state int)

主码为隔离编号，隔离状态：1：正在隔离 2：隔离结束 3：转入医院

**隔离地点表【隔离地点编号，隔离地点名，房间容量】**

isolation\_location (id int, location\_name char(20), capacity int)

主码为隔离地点编号

* + 1. **数据更新**

1. **分别用一条sql语句完成对人员表基本的增、删、改的操作；**
2. **批处理操作**

将行程表中所有到达地点2的记录插入到新表location\_record\_2中。

1. **数据导入导出**

通过查阅DBMS资料学习数据导入导出功能，并将任务2.1所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

1. **观察性实验**

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

1. **触发器实验**

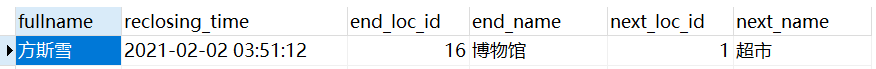
编写一个触发器，用于实现以下完整性控制规则：

当隔离表中的某位隔离人员在诊断表中的诊断结果为“1：新冠确诊”，将隔离状态从“1”改成“3”。

* + 1. **查询**

请分别用一条SQL语句完成下列各个小题的需求：

1. 查询截至目前的人流量大于30的地点，并按照从高到低排序
2. 查询出每个隔离地中正在进行隔离的人数，并按数量由多到少排序
3. 查询行程表中连续行程中的一对行程地点（例如，方斯雪，在2021-02-02 03:02:12至2021-02-02 03:51:12 在博物馆中，2021-02-02 03:51:12 至 2021-02-02 04:23:12出现超市中，输出【姓名，重合时间，起始地点id，起始地点，结束地点id，结束地点】，查询结果如下图所示）



1. 查询贾涵山的行程情况
2. 查询地名中带有‘店’字的地点名称
3. 新发现一位确诊者，已知他在2021.2.2日20:05:40到21:25:40之间在“活动中心”，查询他接触到的人员的名字
4. 查询正在使用的隔离区名（使用DISTINCT关键字）
5. 查询有出行记录的人员（使用EXISTS关键字）
6. 查询没有达到地点“Today便利店”的人数（使用NOT EXISTS 关键字）
7. 查询去过所有地点的人员
8. 新建一个视图，里面有每个隔离点的id,名称，已用房间，容量
9. 从视图中查询还有空房间的隔离点的剩余房间数目
10. 查询靳宛儿传染的人有哪些
11. 查询每个地点的密切接触者的数量
12. 查询感染人数最多的用户的名字和感染人数
13. 查询2021-02-02 10:00:00到14:00:00，行程记录最频繁的五个人及行程记录数目
14. 隔离点中，查询房间数(capacity)居第二多的隔离点名称及其房间数。
    * 1. **了解系统查询性能分析功能**

选择上述任务中某些较为复杂的SQL语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

* + 1. **DBMS函数及存储过程**

1. 编写一个依据人员编号计算其达到所有地点的次数的自定义函数，并利用其查询至少到达过3个地点的人员。
2. 尝试编写DBMS的存储过程，建立每个隔离点的人数统计表，并通过存储过程更新该表。

## 2.2 完成过程

实验环境如下：

操作系统：windows10

数据库：MySQL

交互图形界面：Workbench

Educoder通关截图：



图2.1 educoder测试通过截图

**2.2.1 建表**

首先直接根据老师在群文件中提供的sql文件并结合educoder平台的要求完成数据库和表的创建，并且完成数据库数据的导入。导入完成后在workbench中查看数据库的表格如下：

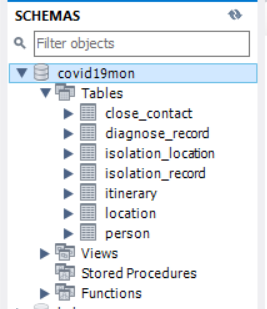


图2.2 数据库表成功创建截图

上图说明数据库被成功创建，主要难点是设置主键的语句以及添加外键的语句比较难写。主键声明直接在constraint语句中声明键的名称以及属性，在设置外键时使用的sql语句为： constraint （需要被设置为外键的名称） foreign key(属性重命名) references person(外键依赖的项)；

**观察性实验：**如果在设置外码时不参考被参照关系的主码，系统会直接报错：



图2.3 设置外码报错截图

### 2.2.2数据更新

1. **增删改操作：**通过使用INSERT，DELETE，UPDATE指令完成对数据的插入删除以及更新。
2. **批处理操作：**把行程表中所有到达地点2的记录插入到新表location\_record\_2中，这里直接使用create和select语句的结合完成：



图2.4 筛选旧表并把结果保存到新表中

1. **数据的导入导出：**通过使用workbench中自带的import和export功能完成数据的导入和导出。其中导出的结果为一个sql文件，并且也可以通过一个SQL文件完成对一个数据库的加载。

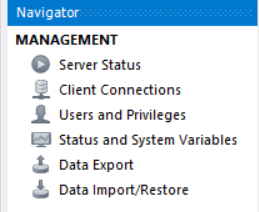


图2.5 数据库的导入和导出

如图2.4所示，直接在workbench中点击data export/import完成数据的导入导出。

1. **观察性实验：**建立一个关系，不设置主码，在关系中插入重复的元组，观察发生的现象：

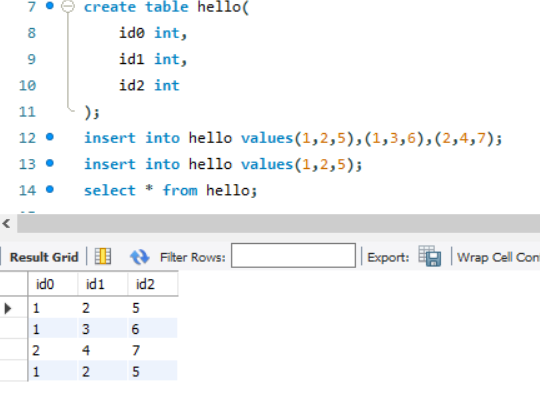


图2.5 在不设置主码的关系中重复插入元素

如上图所示，发现元组(1,2,5)被成功插入两次，说明非主码的属性可以为重复值。

1. 通过编写自己的触发器，完成当隔离表中的某一个隔离人员在诊断表中的在诊断结果为“1：新冠确诊”的时候，将隔离的状态从“1”改成“3”。

通过对触发器的语法进行学习之后，编写了我自己的触发器，这里我的设计思路时分成两个部分来写，第一个部分是用户在插入了新冠确诊病例，针对INSERT操作进行一个触发器操作，第二个部分是在用户更新了某个新冠确诊病例，针对UPDATE操作进行一个触发器操作，最终都是把隔离的状态从1修改为3。具体实现代码如下：

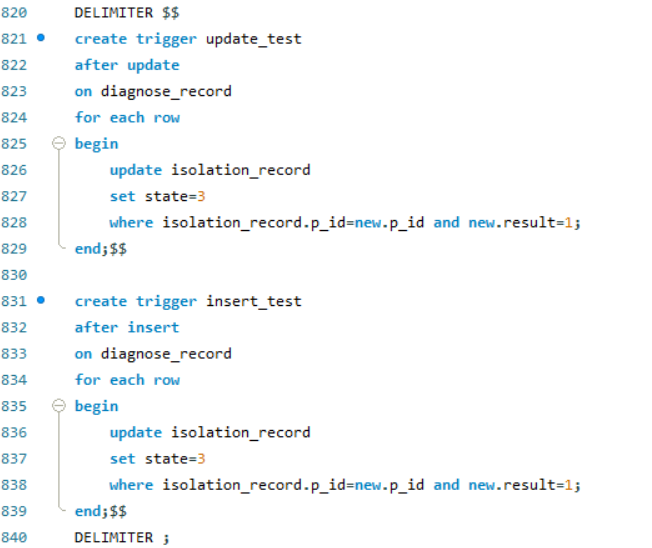


图2.6 触发器更新操作源码截图

创建方式如上图所示，最终成功在educoder上通过测试。

### 2.2.3查询

1）select location\_name,count(p\_id) as visitors

from itinerary,location

where itinerary.loc\_id=location.id

group by loc\_id having visitors>30

order by visitors desc,location\_name asc;

首先通过group by聚集函数针对地点id完成分组，再通过count函数对信息条数进行技术作为访客人数，通过having进行筛选出人数大于30的地点并进行降序排序。

2）select location\_name,count(isol\_loc\_id) as number

from isolation\_record,isolation\_location

where state=1 and isolation\_record.isol\_loc\_id=isolation\_location.id

group by location\_name

order by number desc,location\_name;

通过state进行筛选最后通过order by进行排序。

3）select distinct(a.p\_id) as id,person.fullname,person.telephone,a.e\_time as reclosing\_time,a.loc\_id as loc1,c.location\_name as address1,b.loc\_id as loc2,d.location\_name as address2

from itinerary as a,itinerary as b,location as c,location as d,person

where a.e\_time=b.s\_time and person.id=a.p\_id and c.id=a.loc\_id and d.id=b.loc\_id and a.p\_id>30 and a.p\_id=b.p\_id

order by a.p\_id,reclosing\_time asc;

创建出两个表通过对e\_time和s\_time进行匹配连接最后再通过对p\_id进行distinct操作筛选输出答案。

4）select fullname,telephone,location\_name,s\_time,e\_time

from (person) left outer join (itinerary,location) on(person.id=itinerary.p\_id and location.id=itinerary.loc\_id)

where (fullname='贾涵山' or fullname='充珉瑶')

order by fullname,s\_time;

通过对人物表，行程表，地点表进行连接，再对名字进行筛选得到该同学的行程表。

5）select id,location\_name

from location

where location\_name like '%店%';

通过使用like函数完成对所有带有“店”字的地点名称的筛选。

6）select fullname,telephone

from person,location,itinerary

where person.id=itinerary.p\_id and itinerary.loc\_id=location.id and ((s\_time between '2021-2-2 20:05:40' and '2021-2-2 21:25:40') or(e\_time between '2021-2-2 20:05:40' and '2021-2-2 21:25:40') or (s\_time<'2021-2-2 20:05:40' and e\_time>'2021-2-2 21:25:40'))and location\_name='活动中心'

order by fullname;

通过对事件进行筛选（一共有三种情况表示该人在该时间段内出现在该地点），筛选出在该时间段出现在活动中心的所有人，并通过order by对名字进行排序。

7）SELECT location\_name

from isolation\_location l

WHERE l.id in (

SELECT isol\_loc\_id

FROM isolation\_record r

WHERE r.state=1);

通过嵌套查询，找到state为1的隔离区名。

8）select fullname,telephone

from person inner join itinerary

where fullname!= '充珉瑶'

order by s\_time,person.id limit 0,30;

查询所有出行记录的人员，通过limit只输出排序后的前三十个。

9）select count(person.id) as number

from person

where person.id not in (select p1.id from person p1,itinerary i1,location l1 where p1.id=i1.p\_id and l1.id=i1.loc\_id and l1.location\_name='Today便利店');

通过嵌套查询，筛选出没有到达“Today便利店”的表，在使用not in进行筛选。

10）select fullname

from person,itinerary

where itinerary.p\_id=person.id

group by fullname having count(distinct loc\_id)=16;

直接通过对loc\_id进行计数，结果为16的筛选出来。

11）create view isolation\_location\_status as

select isolation\_location.id as id,isolation\_location.location\_name as location\_name,isolation\_location.capacity as capacity,sum(case when isolation\_record.state=1 then 1 else 0 end) as occupied

from isolation\_location,isolation\_record

where isolation\_location.id=isolation\_record.isol\_loc\_id

group by isolation\_location.id;

通过create view操作完成新的视图的创建。

12）select location\_name,(capacity-occupied) as available\_rooms

from isolation\_location\_status

order by id;

通过11创建出来的视图中的occpuied筛选出灭有被占用的房间数目。

13）select fullname,telephone

from person as x2,itinerary as y2,location as z2

where x2.fullname!='靳宛儿' and z2.location\_name='棋牌室' and y2.p\_id=x2.id and z2.id=y2.loc\_id and ((y2.s\_time between '2021-02-02 16:07:32' and '2021-02-02 17:14:32') or(y2.e\_time between '2021-02-02 16:07:32' and '2021-02-02 17:14:32') or (y2.s\_time<'2021-02-02 16:07:32' and y2.e\_time>'2021-02-02 17:14:32'))

order by fullname;

通过姓名进行筛选，筛选出她出现在某一地点时间段中出现在该地点的所有人的信息。

14）select location.location\_name,count(location.location\_name) as close\_contact\_number

from close\_contact,location

where close\_contact.loc\_id=location.id

group by location\_name

order by close\_contact\_number desc,location\_name;

通过对密切接触者出现的地点进行group by操作筛选出来并使用count函数进行计数。

15）select case\_p\_id,fullname,count(p\_id) as infected\_number

from close\_contact,person

where close\_contact.case\_p\_id=person.id

group by person.id

order by infected\_number desc limit 0,1;

通过count函数进行计数，并进行排序，再使用limit函数筛选出感染人数最多的那个人。

16）select p.fullname,count(\*) as record\_number

from itinerary i,person p

where i.p\_id=p.id

and !(i.e\_time< '2021-02-02 10:00:00' or i.s\_time> '2021-02-02 14:00:00')

GROUP BY i.p\_id

ORDER BY record\_number DESC ,fullname LIMIT 3;

通过对时间段的信息进行分组计数，最后排序，通过limit输出前五名。

17）select location\_name,capacity

from isolation\_location

where capacity in (select a.capacity from (select distinct capacity from isolation\_location group by capacity order by capacity desc limit 1,1) as a);

直接使用嵌套查询配合in进行筛选。

### 2.2.4了解系统的查询性能分析功能（选做）

选择上述2.3任务中某些较为复杂的SQL语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

选择上述第16个查询语句，其分析计划和实际的执行计划如下所示。

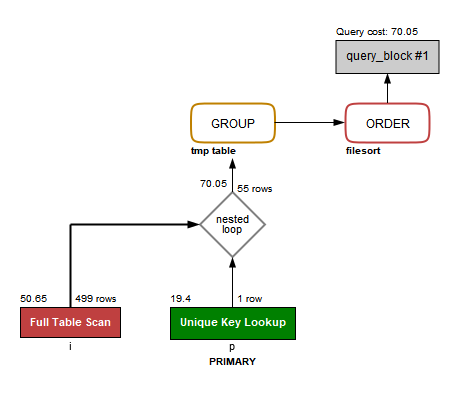


图 2.7 执行计划

从上图中我们可以发现，最大的开销在于索引扫描排序。

### 2.2.5 DBMS函数及存储过程（选做）

1. 编写一个依据人员编号计算其达到所有地点的次数的自定义函数，并利用其查询至少到达过3个地点的人员。

函数：

create function Count\_Records(@personId INT)

returns int

as

begin

declare @sum INT;

select @sum = count(\*)

from itinerary

where @personId = p\_id

return @sum;

end

查询：

select id, fullname, telephone

from person

where dbo.Count\_Records(id) >= 3

order by id;

1. 尝试编写DBMS的存储过程，建立每个隔离点的人数统计表，并通过存储过程更新该表。

## 2.3任务总结

本次任务完成了针对一个数据库进行建表操作，设置主码和外码，并且观察了主码与外码的含义，学会了通过workbench直接向数据库导入数据。

此次实验还完成了对表的增删查改操作，批处理操作。完成了educoder平台上17个查询训练，创建了触发器并且在平台上编写了函数成功通过测试。

这次实验在educoder平台上进行测试，但是我还是先在本地的mysql软件上调试成功后再进行测试，在educoder上提供的命令行调试还是很不习惯。

# 3软件功能学习部分

## 3.1 任务要求

1）练习SQL Server或其他某个主流关系数据库管理系统软件的备份方式：数据和日志文件的脱机备份、系统的备份功能。

2）练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

## 3.2 完成过程

### 3.1.1练习SQL Server备份功能。

针对创建好的数据库手动进行备份，如3.1所示：

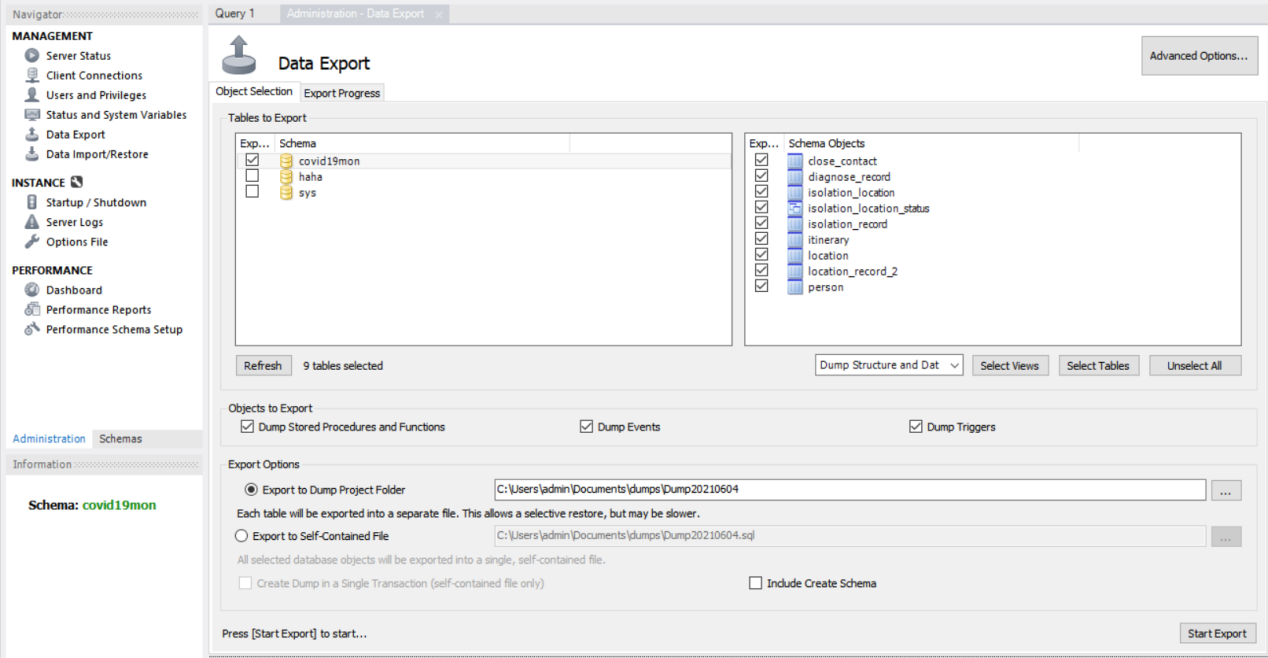


图 3.1 系统手动备份

注意在上图中，我选择备份的数据库是covid19mon这个数据库，默认选择所有的表，设置好保存路径，最后就成功生成一个sql文件保存在目标目录下。数据和日志文件的脱机备份如图3.2，进行数据的备份如下图所示：

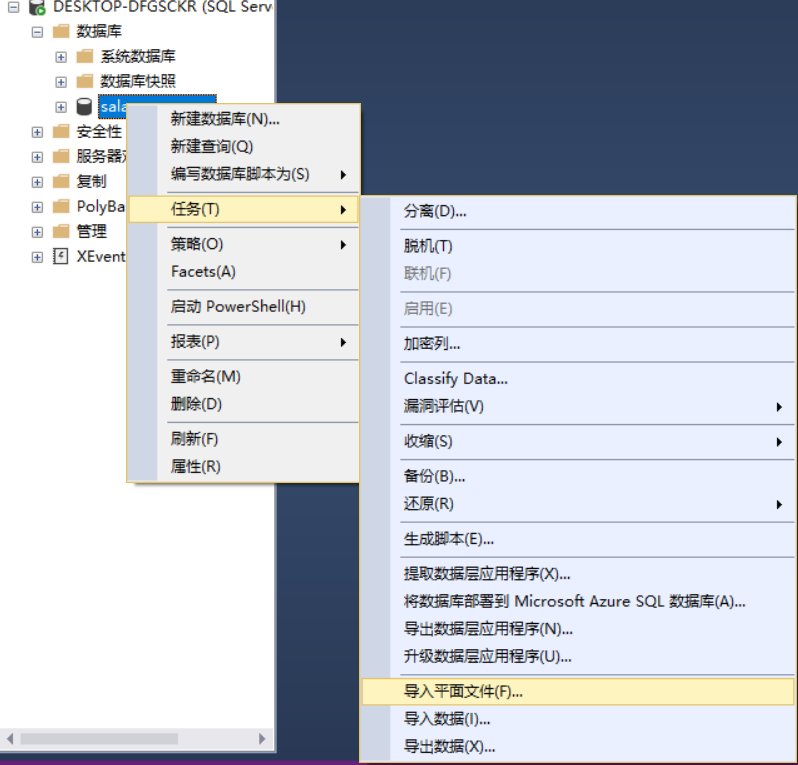


图 3.2 脱机备份

### 3.1.2增加用户并配置权限

右键安全性->用户名可以创建用户test。

在SqlServer新建的数据库中右键打开数据库属性，在权限一栏中可以增加用户并更改权限，如图 3.4。

拒绝更新权限后尝试更新person表，结果如图 3.5。

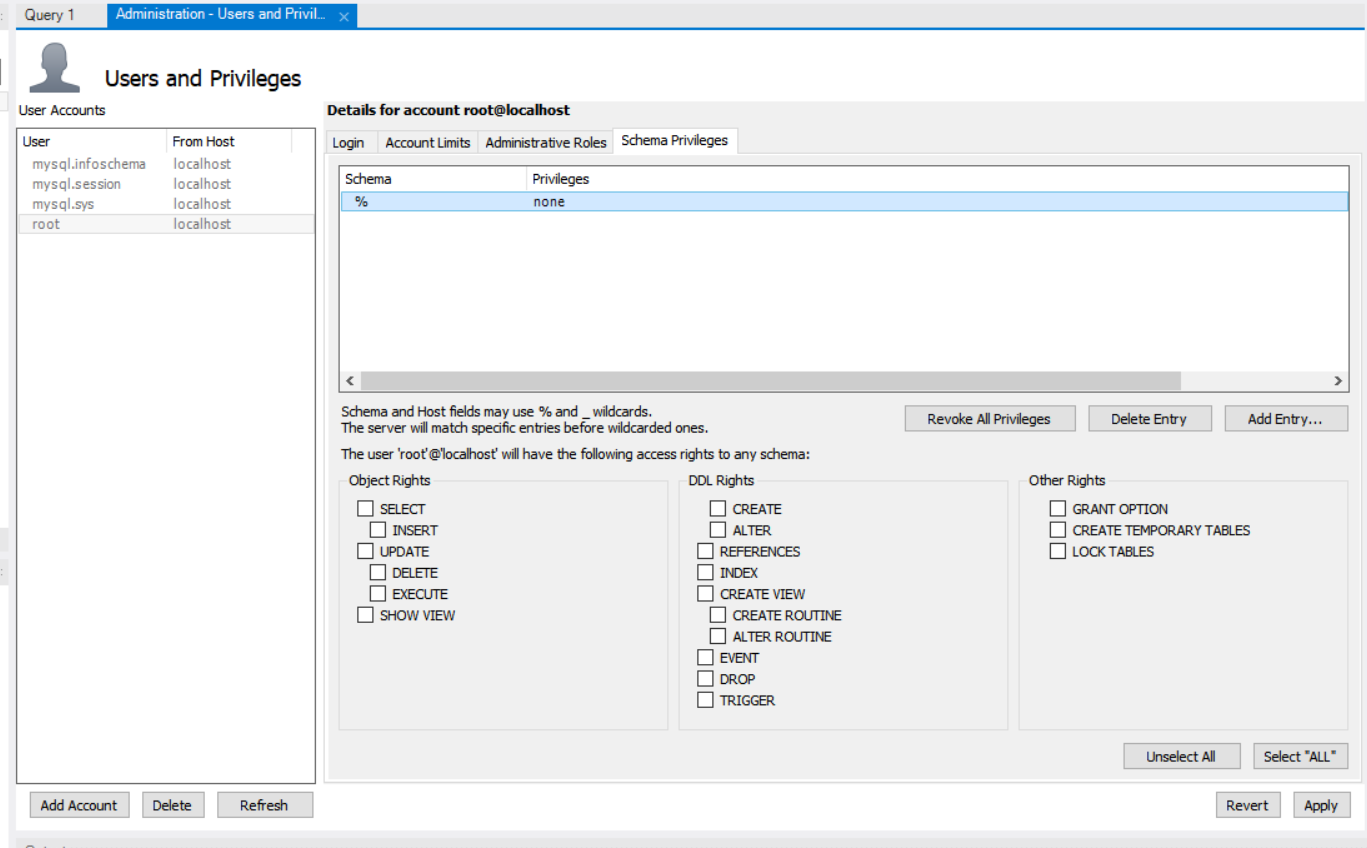


图 3.3 用户权限更新

全选后点击apply可以看到权限已经成功的更新了，更新后的权限如下图  


图3.4 用户权限更新成功

## 3.3任务总结

本次实验主要是通过查询资料和询问同学进行，让我对SqlServer的备份和配置权限有了了解。

# 4 综合实践任务

## 4.1系统设计目标

数据库是人们用来针对数据进行管理的新兴技术，是计算机学科中一个重要的分支。为了帮助人们在生活中解决各种类型的问题，数据库在数据管理方面的优势便很好的体现了出来。于是，作为信息处理系统核心和基础的数据库技术得到越来越广泛的应用。从小型单项事务处理系统到大型信息系统，从联机事务处理到联机分析处理，从一般企业管理到计算机辅助设计与制造，计算机集成制造系统、电子政务、电子商务、地理信息系统等，越来越多的应用领域采用数据库技术来存储和处理信息资源。众所周知，机场作为一个公共交通枢纽，每天需要产生大量的数据，如何针对这些数据进行管理便成为了一项很大的工程。

本次课程设计选题为机票预订系统。采用C/S架构来搭建整个软件框架。设计出来的系统应该可以完成以下的功能：

1. 每个航班信息的输入。
2. 每个航班的坐位信息的输入；
3. 当旅客进行机票预定时，输入旅客基本信息，系统为旅客安排航班，打印取票通知和帐单；
4. 旅客在飞机起飞前一天凭取票通知交款取票；
5. 旅客能够退订机票；
6. 能够查询每个航班的预定情况、计算航班的满座率。
7. 包含事务（包含commit，rollback），存储过程/触发器，视图，函数。

## 4.2 需求分析

展开需求分析，给出需求分析的主要内容的阐述，包括具体的功能需求、性能需求、数据完整性需求、数据流图、数据字典等。

**4.2.1 系统功能需求**

整个机票预订系统的功能可以具体划分为以下的几个模块：



图4.1 系统功能需求图

其中主要的模块主要分为两个，一个是用户功能模块，一个是管理员模块。用户模块分为两个模块，查看模块和修改信息模块。查看模块中只能对数据进行查看，具体包括了搜索航班信息，查看已经预定的机票订单，查看已经退订的机票订单和信息接收功能。用户信息修改模块包括了三个功能，分别是用户的下订单记录，取票操作以及用户的密码修改。

管理员模块同样也分为两个模块，一个是管理员查看信息模块，一个是管理员修改信息模块。管理员查看信息模块包括了查看所有用户的账号以及密码，查看所有用户的订单记录，查看所有用户的退单记录，查看给所有用户的取票记录以及航班记录。另一个是管理员修改信息模块，包括了航班信息，用户账号以及密码的修改。

**4.2.2 系统数据流图**

数据库中应该包含以下几张表：

1. 管理员账号密码表
2. 用户账号密码表
3. 航班信息表
4. 用户订单表
5. 用户退订表
6. 用户取票信息表

数据流图如下：

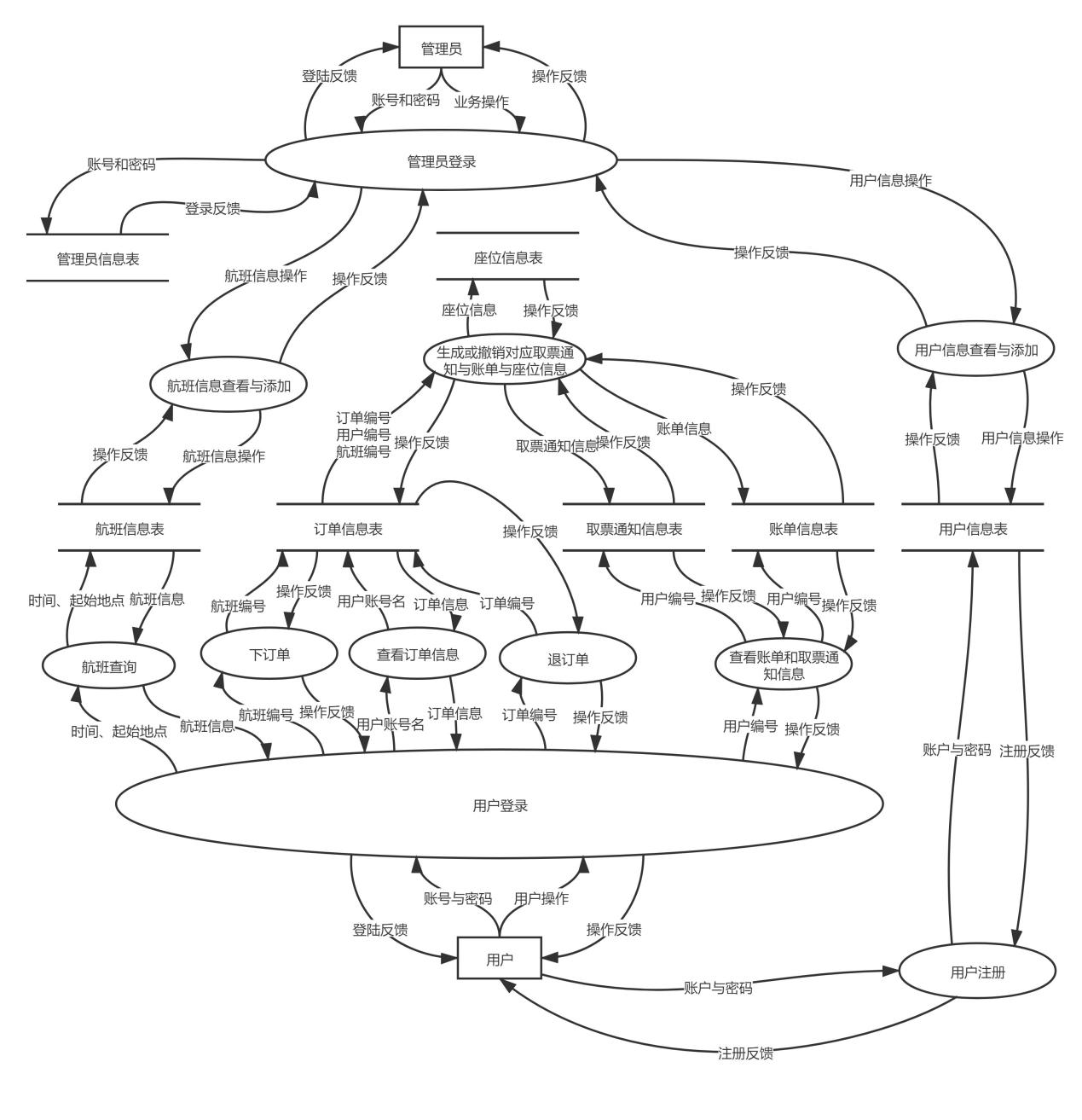


图4.2 系统数据流图

根据上面制定的系统功能需求图绘制出系统数据流图，完整的介绍了管理员和用户在对每一个数据表的数据流操作。

## 4.3 总体设计

**4.3.1 系统架构**

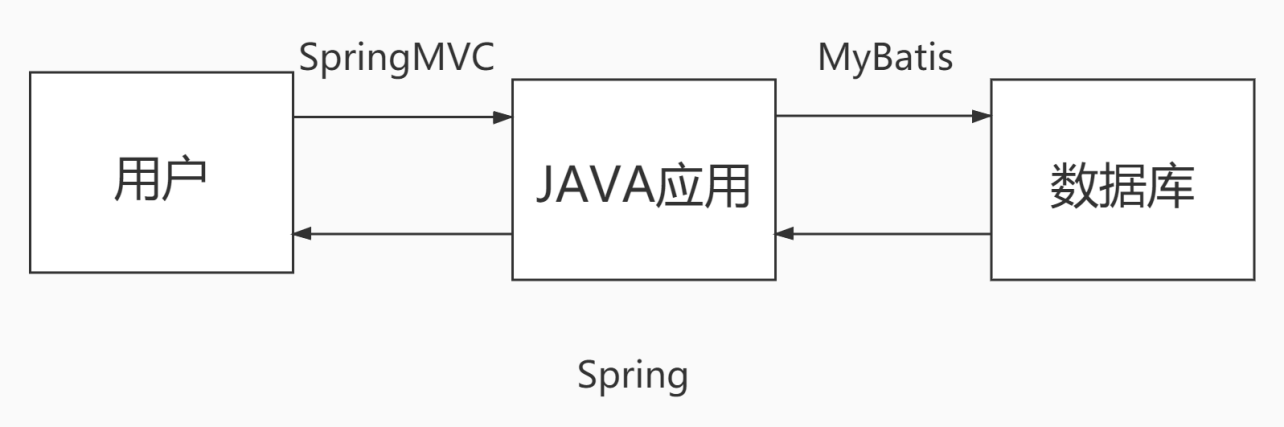
本次实验，我决定采用B/S架构针对整个程序进行实现。B/S程序架构图如下：

图4.3 系统总体架构

这里，我决定使用SSM框架进行开发。SSM框架是Spring，SpingMVC，MyBatis三项技术的统称。

Spring是一个开源框架，是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。简单来说，Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

MyBatis是一个基于Java的持久层框架。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs映射成数据库中的记录。

## 4.4 数据库设计

**4.4.1 ER图设计**

根据题目要求设计出的ER图如下：

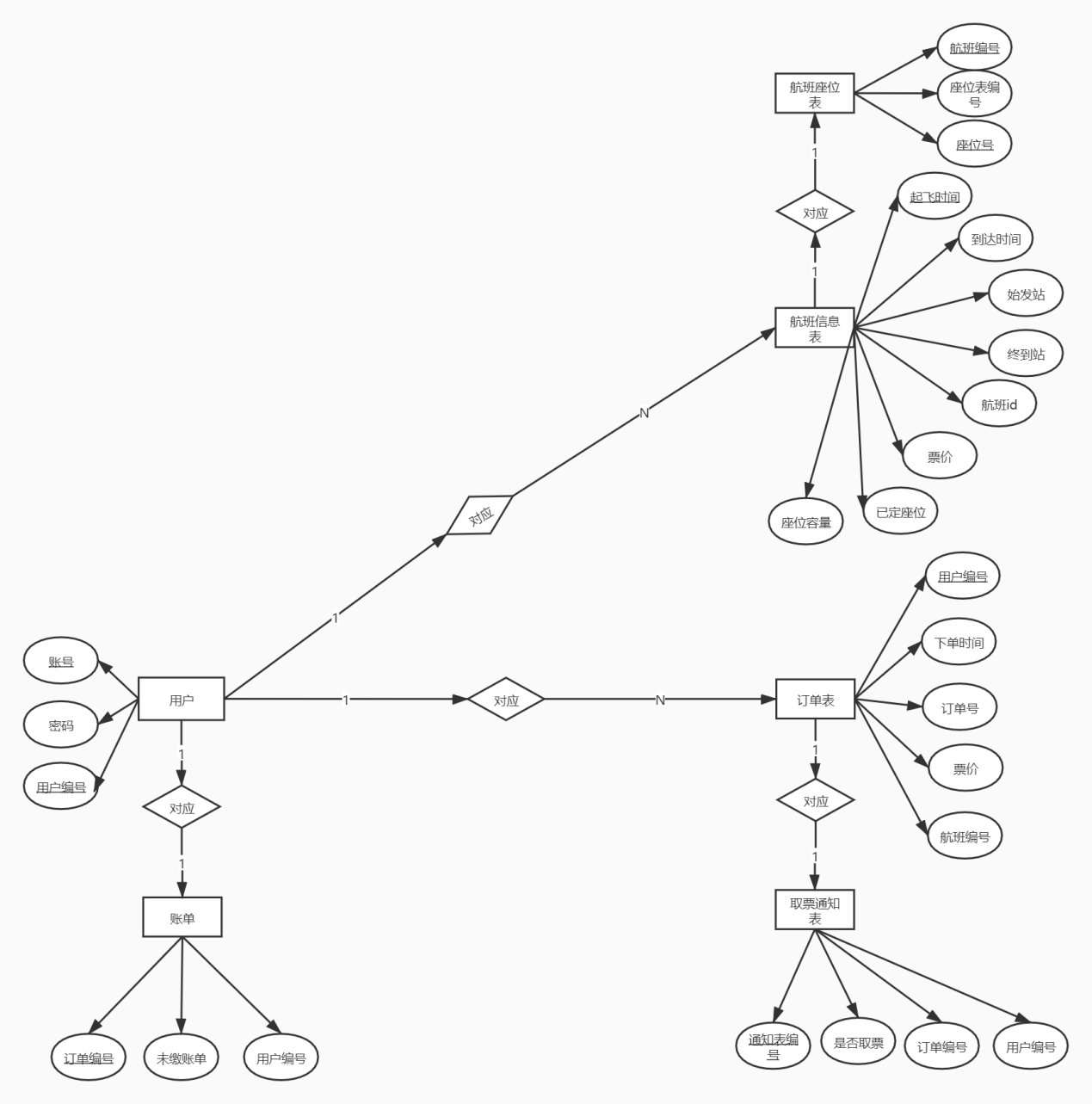
****

图4.4 系统总体ER图

**4.4.2 数据字典设计**

1. 管理员信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 管理员ID | INT |  |
| Username | 管理员账号名 | CHAR(20) | 主码 |
| Password | 密码 | CHAR(20) |  |

1. 用户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 用户编号 | INT | 自增 |
| Username | 用户账号名 | CHAR(20) | 主码 |
| Password | 密码 | VARCHAR（20） |  |

1. 账单表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 账单编号 | INT | 主码，自增 |
| P\_id | 人员编号 | INT | 外键 |
| Pay | 账单金额 | INT |  |

1. 订单表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 订单编号 | INT | 主码，自增 |
| P\_id | 人员编号 | INT | 外键 |
| Air\_info | 航班编号 | CHAR(20) | 外键 |
| Booktime | 下单时间 | DATETIME |  |
| Price | 票价 | INT |  |

1. 取票通知表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 取票通知表编号 | INT | 主码，自增 |
| P\_id | 人员编号 | INT | 外键 |
| Book\_id | 订单编号 | INT | 外键 |
| Withdraw | 是否成功取票 | BOOL |  |

1. 航班信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 航班编号 | CHAR(20) | 主码 |
| Takeoff\_time | 起飞时间 | DATETIME |  |
| Arrive\_time | 到达时间 | DATE |  |
| origin\_stat | 始发站 | CHAR（20） |  |
| Term\_stat | 到达站 | CHAR（20） |  |
| Price | 票价 | INT |  |

1. 航班座位表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 |
| ID | 航班座位表编号 | INT | 主码，自增 |
| Flight\_id | 航班编号 | CHAR（20） | 外键 |
| Seatnum | 座位号 | INT |  |

**4.4.3 触发器以及视图设计**

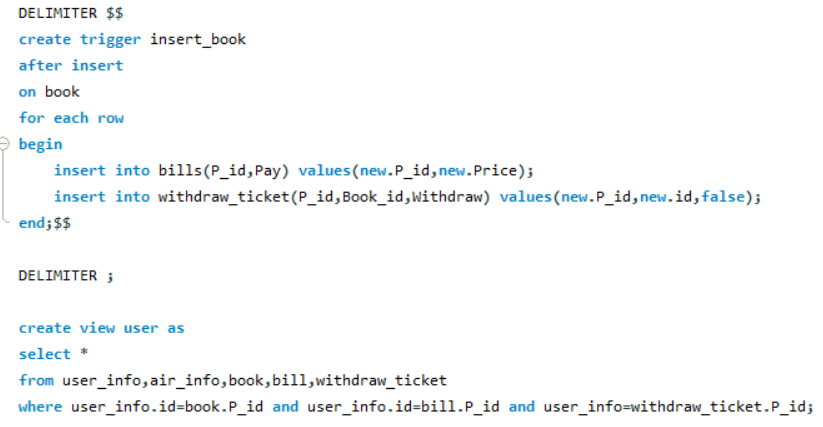


图4.5 触发器以及视图设计

通过以上的触发器能够在插入新的订单信息时直接插入对应的账单信息以及取票信息，创建的视图能够只让用户访问数据，配合对用户的授权管理实现权限管理功能。

## 4.5 详细设计与实现

本系统把数据库中表的定义以及触发器的定义在MySQLWorkbench进行运行，通过后端的SSM框架实现需要的业务逻辑，最后把数据传回到前端jsp页面上，前端的页面采用了bootstrap提供的静态页面样例进行改造。

**4.5.1 业务流程图**

根据整个系统框架绘制出一幅页面逻辑图，如图4.5所示：

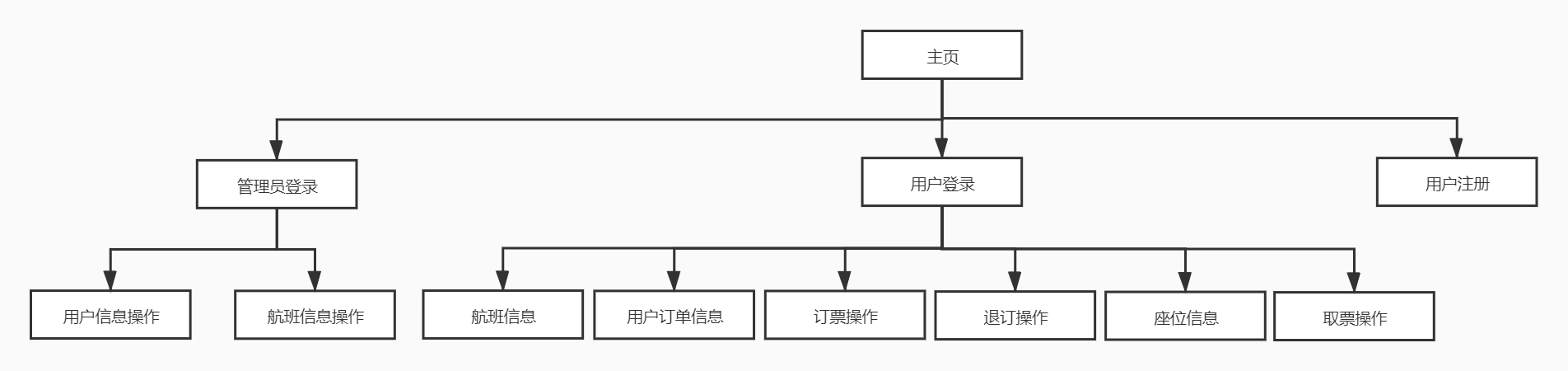


图4.6 页面逻辑图

**4.5.2 文件逻辑**

文件一共分为三个主要模块。

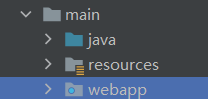


图4.7 文件主要模块

第一个是java模块，包含了除了数据库建表操作以及触发器功能的主要后端代码。java文件夹包含了所有的后端实现代码，其中controller文件中包含了接收从前端传输过来的各种信号，并按照设计要求通过编写java程序完成相应动作。dao层和entity层是通过mybatis generator逆向工程根据在数据库中已经设计好的表自动生成的两个文件夹。dao层中包括了所有表名称的mapper，每一个mapper中包含了基本的增删查改的接口，entity中包含了每个表的实体类型，并且实现了每个实体的example类型帮助我们更加便捷的编写自己需要的数据库操作。这里给出mybatis生成的entity类的代码样例：

Entity中的AirSeat样例：

package yyx.entity;  
public class AirSeat {  
 private Integer id;  
 private String flightId;  
 private Integer seatnum;  
 public AirSeat(Integer id, String flightId, Integer seatnum) {  
 this.id = id;  
 this.flightId = flightId;  
 this.seatnum = seatnum;  
 }  
 public AirSeat() {  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getFlightId() {  
 return flightId;  
 }  
 public void setFlightId(String flightId) {  
 this.flightId = flightId == null ? null : flightId.trim();  
 }  
 public Integer getSeatnum() {  
 return seatnum;  
 }  
 public void setSeatnum(Integer seatnum) {  
 this.seatnum = seatnum;  
 }  
}

对应的dao层的AirSeatmapper样例：

package yyx.dao;  
import java.util.List;  
import org.apache.ibatis.annotations.Param;  
import yyx.entity.AirSeat;  
import yyx.entity.AirSeatExample;  
public interface AirSeatMapper {  
 long countByExample(AirSeatExample example);  
  
 int deleteByExample(AirSeatExample example);  
  
 int deleteByPrimaryKey(Integer id);  
  
 int insert(AirSeat record);  
  
 int insertSelective(AirSeat record);  
  
 List<AirSeat> selectByExample(AirSeatExample example);  
  
 AirSeat selectByPrimaryKey(Integer id);  
  
 int updateByExampleSelective(@Param("record") AirSeat record, @Param("example") AirSeatExample example);  
  
 int updateByExample(@Param("record") AirSeat record, @Param("example") AirSeatExample example);  
  
 int updateByPrimaryKeySelective(AirSeat record);  
  
 int updateByPrimaryKey(AirSeat record);  
}

第二个是resources模块，里面包括了基本的mybatis配置文件，springmvc配置文件，以及自动生成的mapper的xml文件，对应了dao层中的每一个接口。

第三个是webapp模块，包含了我们使用的bootstrap依赖包和所有的前端页面的jsp文件，实现了前端页面的显示。

**4.5.3 具体工作举例**

接下来我会详细介绍如何完成一个具体页面的具体开发。

如何完成用户订单功能介绍。

首先能够进入订单页面的前提是用户已经成功完成登录，进入了用户主页。我们之前已经把用户id作为userid传到了当前页面中：



图4.8 userid传参功能

接下来需要我们实现点击订票页面这个按钮实现页面的跳转，在前端页面中为这个按钮添加信号连接：

<li><a href="/suitairinfo/${userid}">订票页面</a></li>

代表在按下这个按钮后会发出一个suitairinfo的信号，这个信号会被后端的controller层的subscribecontroller文件中接收后进行以下的操作处理：

@RequestMapping("/suitairinfo/{userid}")  
public String suitairinfo(@PathVariable(value="userid")Integer userid ,Model model){  
 model.addAttribute("userid",userid);  
 return "WEB-INF/views/searchpage";  
}

这个方法代表从前端收到一个传来的userid参数，userid的传参类型是路径传参，需要在方法变量声明中使用PathVariable标记，此外还会传入一个会话变量model，这里不对该变量进行任何操作，把userid再添加回model中再传回前端并返回接下来跳转的页面serachpage：



图4.9 searchpage页面截图

可以看到这个页面我设计了五个输入框需要用户手动输入，为了完成这个页面逻辑，前端需要我们添加一个表单，每个input代表了一个输入栏，一共有5个input包括在form表单中，最后的button按钮为提交表单操作，提交后的动作我设置为跳转到findsubscribe页面。

<thread>  
 <form class="form-signin" action="/findsubscribe/${userid}/1" ,method="get">  
 <th><input name="from" class="form-control" placeholder="出发地" required autofocus></th>  
 <th><input name="to" class="form-control" placeholder="目的地" required autofocus></th>  
  
 <th><input name="year" type="number" class="form-control" placeholder="year" required autofocus></th>  
 <th><input name="month" type="number" class="form-control" placeholder="month" required autofocus></th>  
 <th><input name="day" type="number" class="form-control" placeholder="day" required autofocus></th>  
 <th><button class="btn btn-lg btn-primary btn-block" type="submit">查询</button></th>  
 </form>

我在前端发出的跳转信号会被controller层接收，以及提交的表单会使用httpget方法传送到后端。在controller层中的subscribecontroller文件中接收信号并进行搜索操作：

@RequestMapping(value="/findsubscribe/{userid}/{pn}",method = RequestMethod.*GET*)  
public String findsubscribe(@PathVariable(value="pn")Integer pn,@PathVariable(value="userid")Integer userid,  
 @RequestParam(value="from",defaultValue = "wuhan") String from,@RequestParam(value="to",defaultValue = "guangzhou")String to ,  
 @RequestParam(value="year",defaultValue = "2021") Integer year,@RequestParam(value="month",defaultValue = "6")Integer month,  
 @RequestParam(value="day",defaultValue = "18")Integer day, Model model) throws ParseException {  
 PageHelper.*startPage*(pn, 5);  
 List<AirInfo> getfromdb=airInfoService.getset(from,to,year,month,day);  
 PageInfo page=new PageInfo(getfromdb,5);  
 model.addAttribute("pageInfo",page);  
 model.addAttribute("userid",userid);  
 model.addAttribute("from",from);  
 model.addAttribute("to",to);  
 model.addAttribute("year",year);  
 model.addAttribute("month",month);  
 model.addAttribute("day",day);  
 System.*out*.println(from);  
 return "WEB-INF/views/subscribe";  
}

可以看到这里我通过httpget方法获取了前端传来的form表单，获取了对应的变量名。为了实现我想要完成的查找在该时间段之后起飞的满足要求的查找方式，需要在service层中加入一个airinfoservice类，在其中编写出我们需要的方法，最后完成封装，这时再在controller层创建出一个airinfoservice对象出来调用我们在service层中编写的getset方法。在service层中getset方法具体实现代码如下：

public List<AirInfo> getset(String from,String to,Integer year,Integer month,Integer day) throws ParseException {  
 SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");  
 Date date1 = dateformat.parse(year+"-"+month+"-"+day);  
 Date date2 = dateformat.parse(year+"-"+month+"-"+(day+1));  
 AirInfoExample airInfoExample=new AirInfoExample();  
 AirInfoExample.Criteria c1=airInfoExample.createCriteria();  
 c1.andTakeoffTimeBetween(date1,date2).andOriginStatEqualTo(from).andTermStatEqualTo(to);  
 return airInfoMapper.selectByExample(airInfoExample);  
}

注意到这里我使用了selectbyexample这个方法，由于mybatis generator工具只生成了根据主码进行增删查改的功能，此外该工具专门留了一个接口用来实现各种其他非主属性的增删查改以及大小比较等等功能，这些都需要我创建出一个对应的example对象出来并同时创建出对应的critia对象，并且把对应的约束条件放在critia对象中，最后调用selectbyexample方法完成个性化增删查改，配合后端的java代码逻辑，甚至可以完成sql语句并不能够实现的功能，极大的提高了程序开发者的扩展性。这里getset方法最后返回了指定时间区间内的航班信息，传到controller层，把对应的数据添加到model中传回前端，进入subscribe页面：

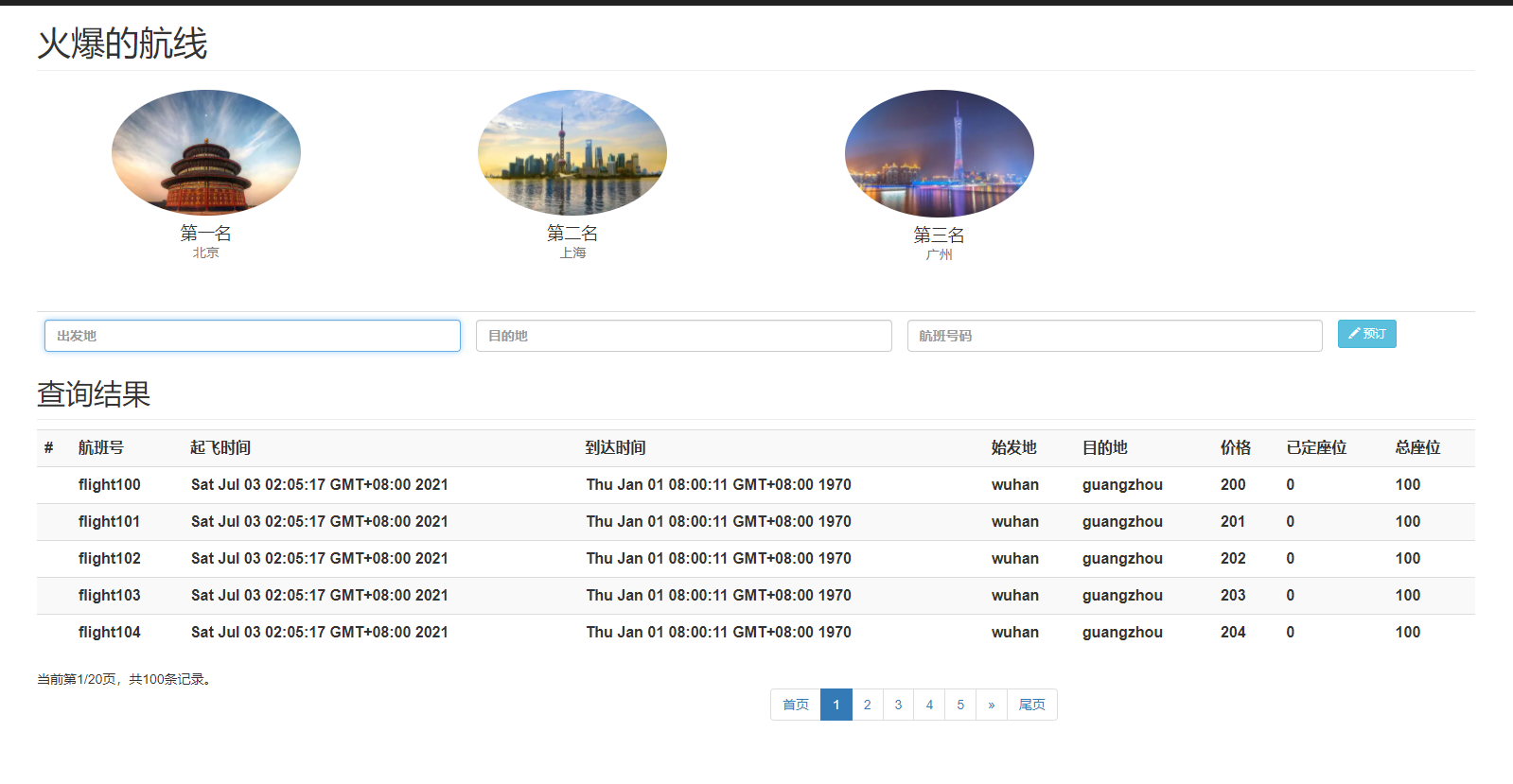


图4.10 subscribe界面显示截图

同样的，这里用户可以直接输入自己需要预定航班的出发地，目的地和航班号，点击预定按钮完成机票预订。实现的前端代码如下：

<table class="table table-stripped">  
 <thread>  
 <form class="form-signin" action="/subscribe/${userid}" ,method="get">  
 <th><input name="from" class="form-control" placeholder="出发地" required autofocus></th>  
 <th><input name="to" class="form-control" placeholder="目的地" required autofocus></th>  
 <th><input name="airinfo" class="form-control" placeholder="航班号码" required autofocus>  
 </th>  
 <th><button type="submit" class="btn btn-info btn-sm">  
 <span class="glyphicon glyphicon-pencil" aria-hidden="true"></span>  
 预订  
 </button></th>  
 </form>  
 </thread>  
</table>

前端发出的subscribe信号被后端的controller层接收，并进行一系列操作：

@RequestMapping(value="/subscribe/{userid}",method = RequestMethod.*GET*)  
public String subscribe(@RequestParam(value="from")String from,@RequestParam(value="to")String to,  
 @PathVariable(value="userid")Integer userid,  
 @RequestParam(value="airinfo") String airinfo,  
 Model model ){  
 AirInfo airInfo=airInfoMapper.selectByPrimaryKey(airinfo);  
 if(airInfo.getBookedseats().equals(airInfo.getTotalseats())){  
 model.addAttribute("userid",userid);  
 return "WEB-INF/views/userpage";  
 }  
 Integer price=airInfo.getPrice();  
 bookMapper.insertSelective(new Book(null,userid,airinfo,new Date(),price));  
 airInfoMapper.updateByPrimaryKey(new AirInfo(airInfo.getId(),airInfo.getTakeoffTime(),  
 airInfo.getArriveTime(),airInfo.getOriginStat(),  
 airInfo.getTermStat(),airInfo.getPrice(),  
 airInfo.getBookedseats()+1,airInfo.getTotalseats()));  
 airSeatMapper.insertSelective(new AirSeat(null,airInfo.getId(),airInfo.getBookedseats()));  
 model.addAttribute("userid",userid);  
 return "WEB-INF/views/userpage";  
}

这里看到在插入新的订票信息后，会在该航班的已订座位数上加一，由于在最初的数据库设计的时候，已经添加了触发器，在添加新的订单时会同时添加账单以及取票通知单，最后返回用户主界面。至此用户的订票功能整个开发流程完成。

其余的各种页面逻辑功能基本上都可以按照这种类似的开发流程完成，最后完成所有功能的联调，消除所有的bug即可。

## 4.6 系统测试

首先在mysqlworkbench中运行一遍自己的sql脚本，完成数据库表触发器视图等等的创建，接着在java工程里面的test文件夹目录下运行mappertest文件，完成数据的插入，插入完成后切换到tomcat服务器进行运行测试。默认自动打开index.jsp页面如下：

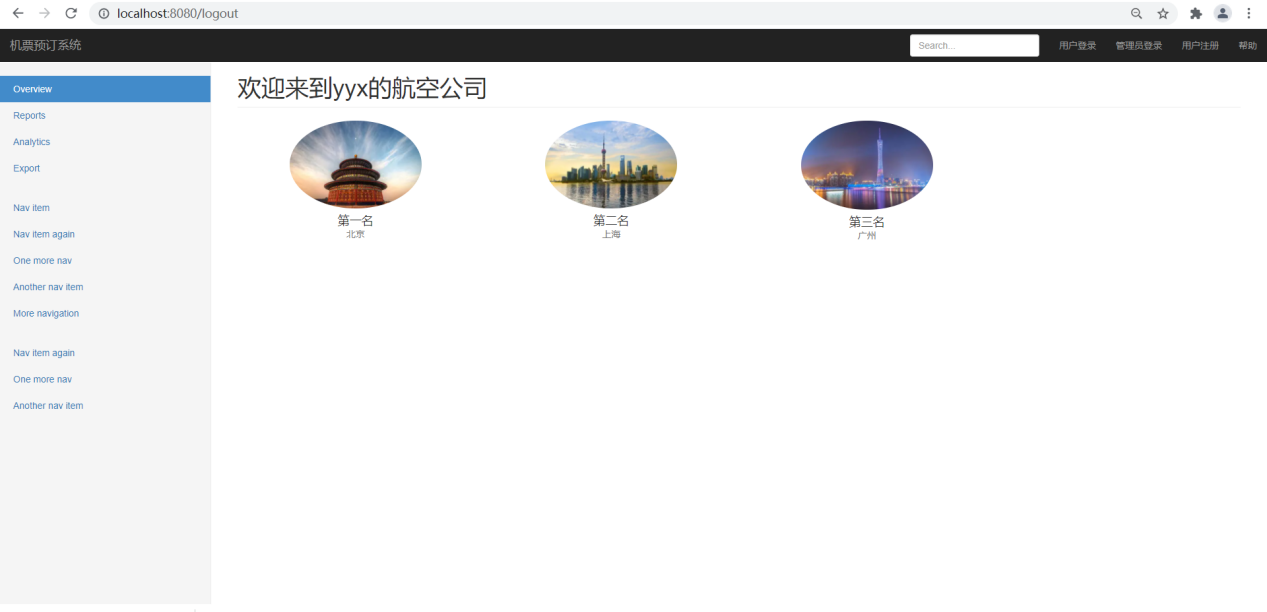


图4.11 主页截图

首先测试用户注册功能，点击右上角的用户注册按钮，输入注册的用户名为“杨雨鑫”，密码为12345678，点击注册按钮：

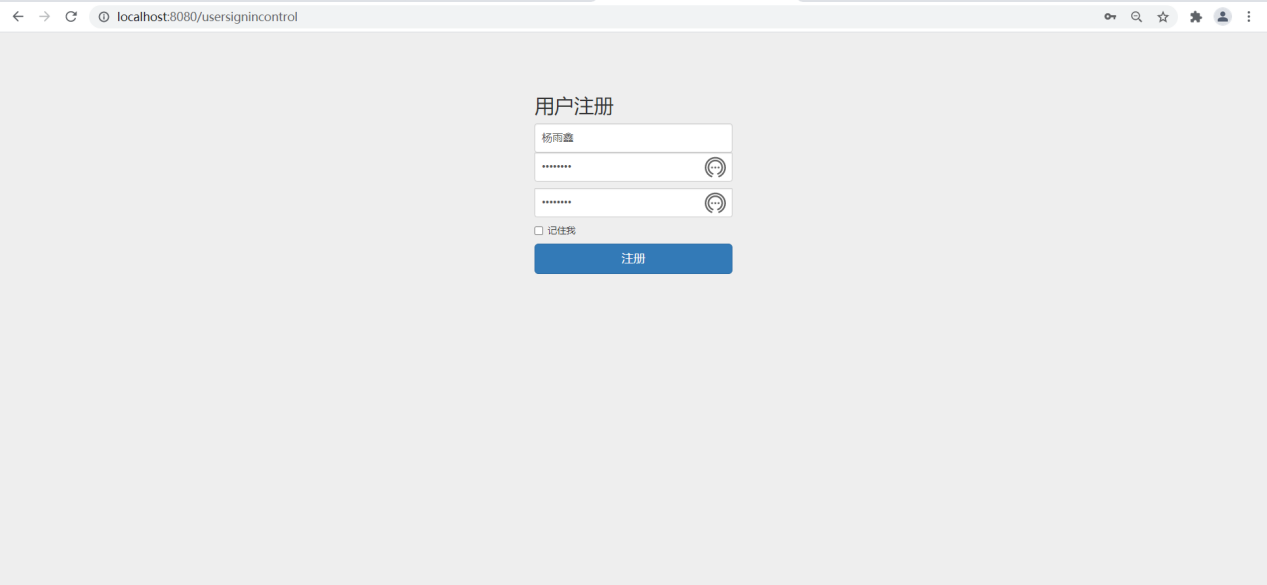
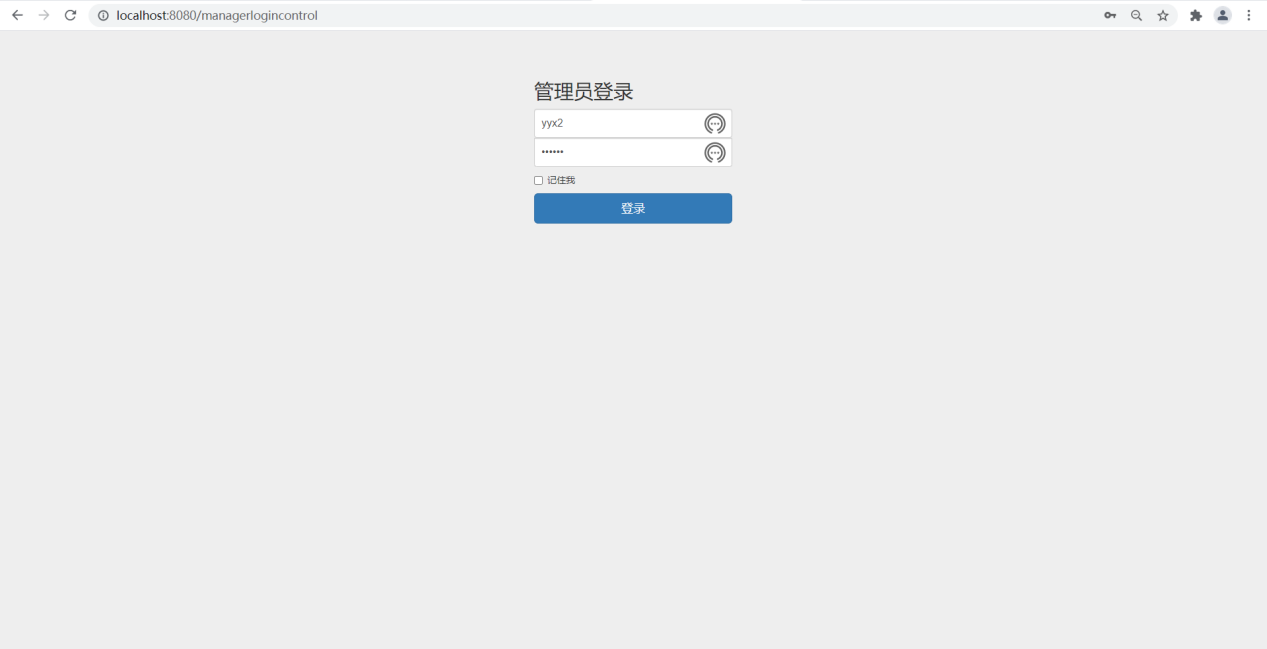


图4.12 用户注册截图

注册成功之后我们会返回到之前的主页，接下来我们使用管理员账号登陆，之前插入数据的时候管理员一共有100个，都以yyx开头，这里我们选择用户名为yyx2，密码为123456的管理员进行登录：



4.13 管理员用户登录

登陆后成功进入管理员主页，可以看到右上角有提示信息“你好管理员”，代表登陆成功：

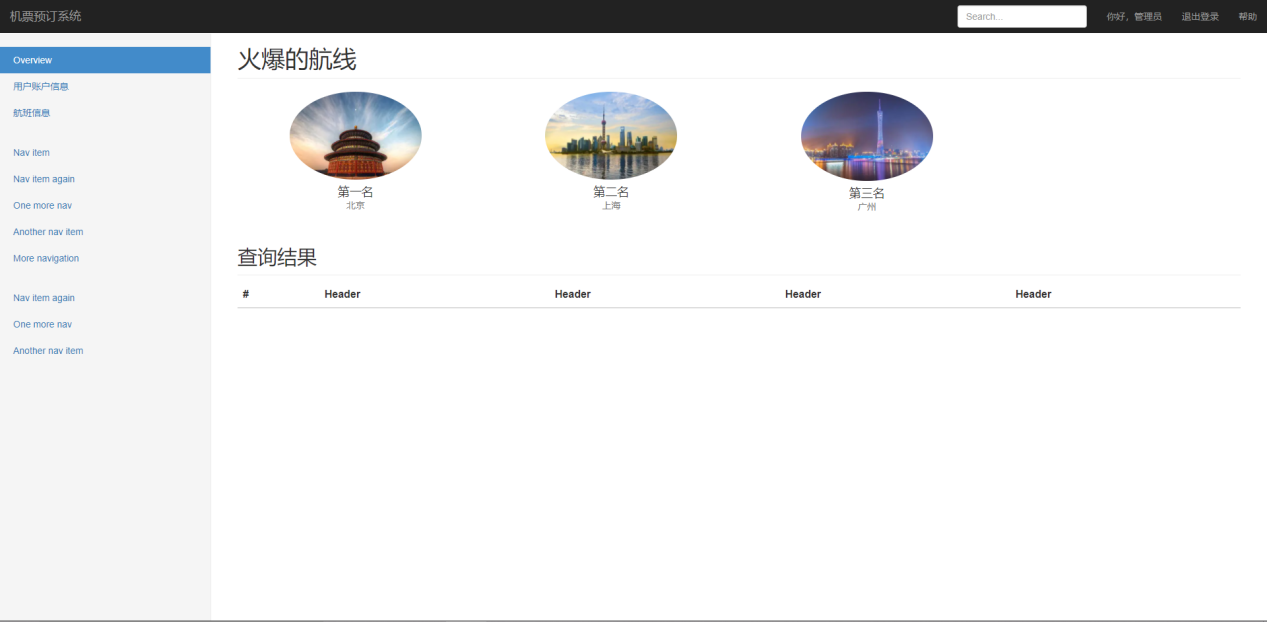


图4.14 管理员主页

接下来为了验证我们刚才是否成功进行了用户注册，我们进入管理员的查看用户信息按钮，查看所有的用户信息：

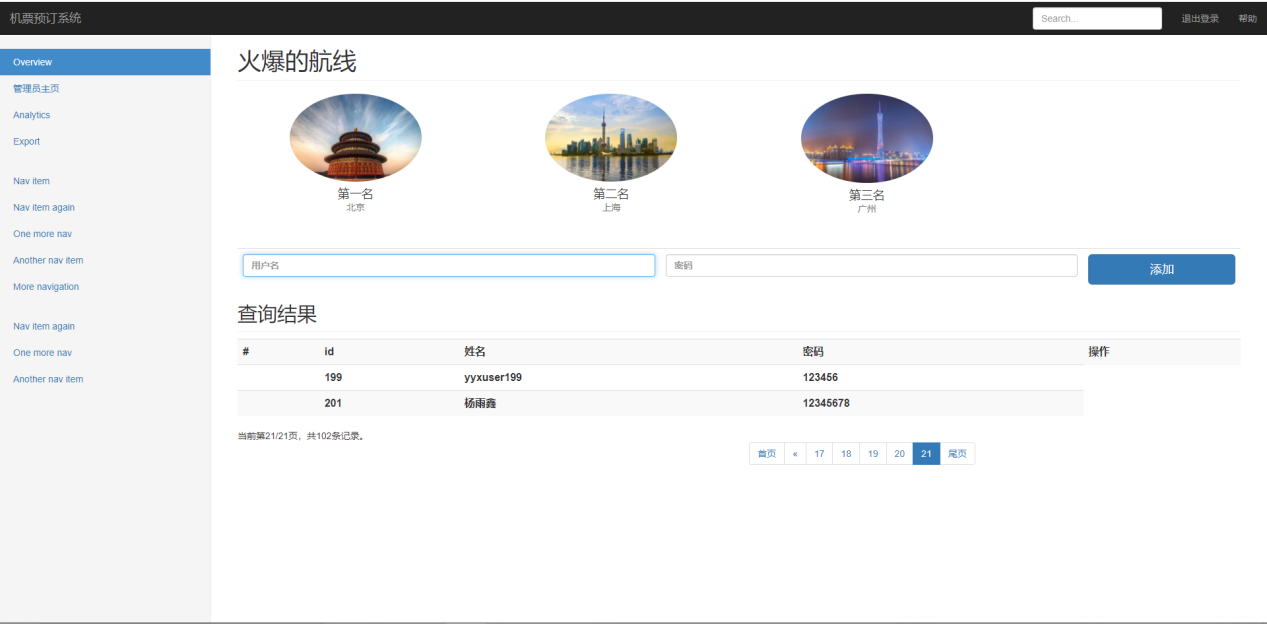


图4.15 查看用户信息功能截图

在尾页找到了刚刚添加的用户杨雨鑫，以及密码12345678，说明刚才的用户注册功能正确。接下来在这个页面直接添加新的用户，输入用户名“张三”，密码“hahaha”点击添加：

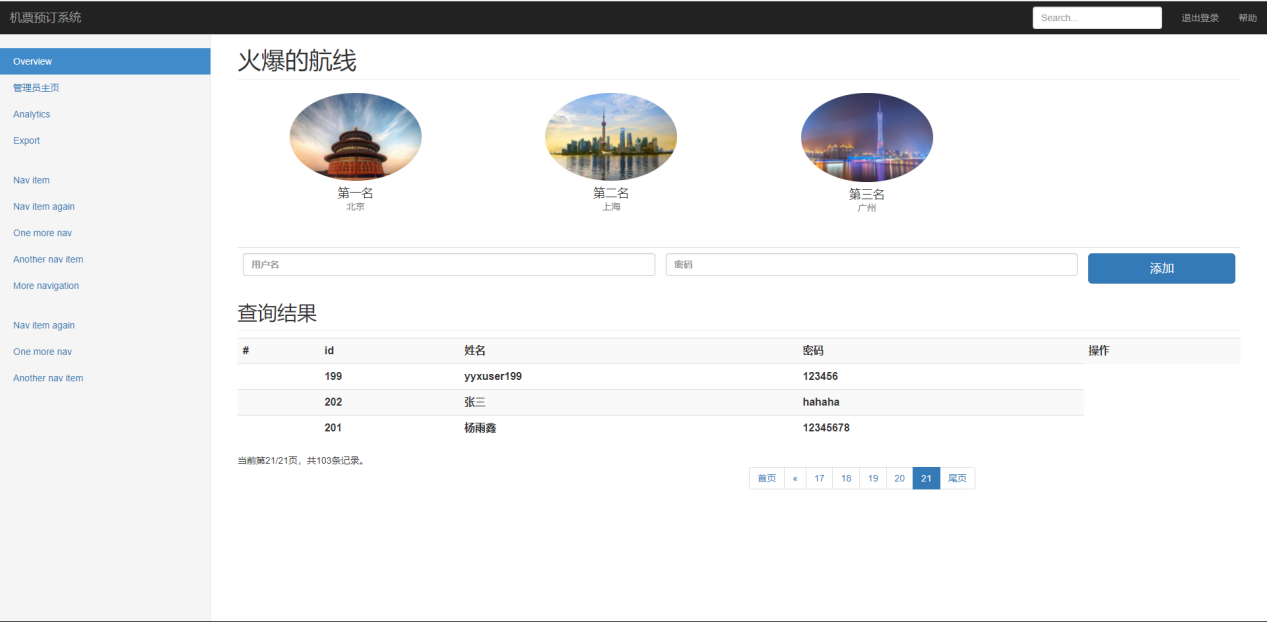


图4.16 管理员直接添加用户信息

同样的在尾页看到了新插入的用户张三的信息，说明管理员的插入用户信息功能成功。接着我们点击左边的按钮回到管理员主页，点击左侧的航班信息按钮进入航班信息页面：

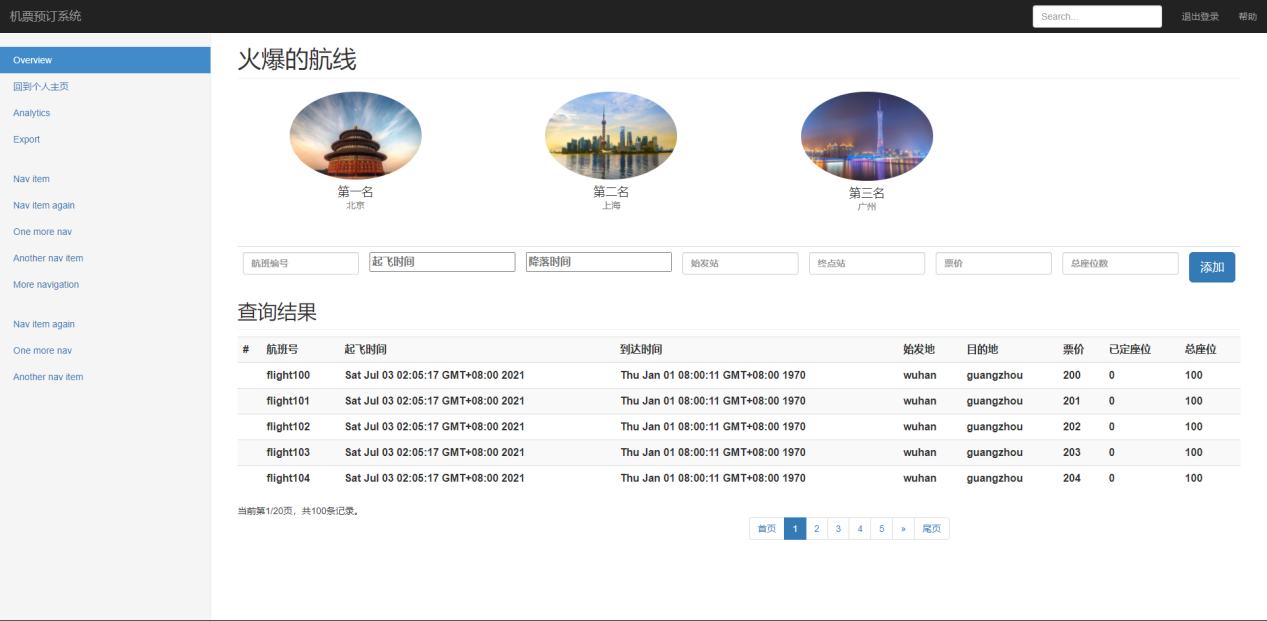


图4.17 管理员编辑航班信息页面

这里我们可以看到所有的航班记录信息，我们可以添加新的航班信息进入数据库中，这里我们添加NH3575号航班，2021年3月4日22：00起飞，2021年3月5日0：20降落，始发站为武汉，终点站为北京，票价为500，总座位数为120。点击添加按钮完成航班信息的添加：

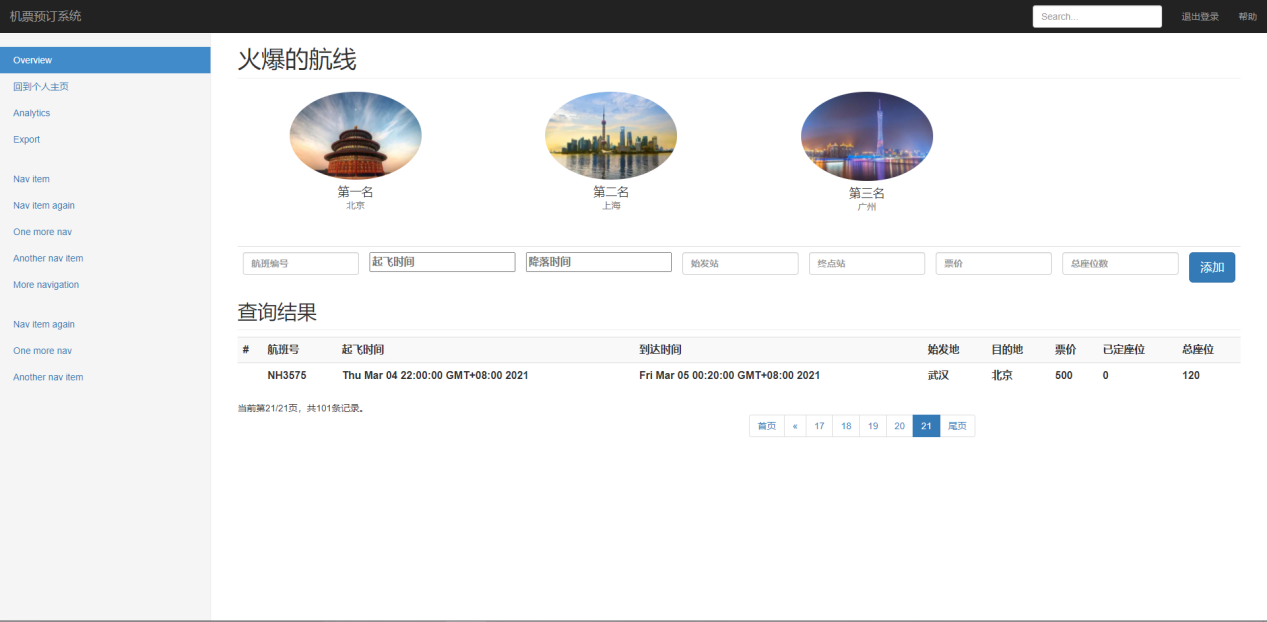


图4.18 管理员添加新的航班信息

添加完成之后，我们在尾页看到了刚才添加的新的航班信息，信息与我们之前添加的一致，已订座位数初始化为0，说明航班添加功能正确。至此，管理员功能全部测试完毕，点击右上角的退出登录回到首页。重新选择用户登录，这里我们使用的是之前创建的账户杨雨鑫，密码12345678：

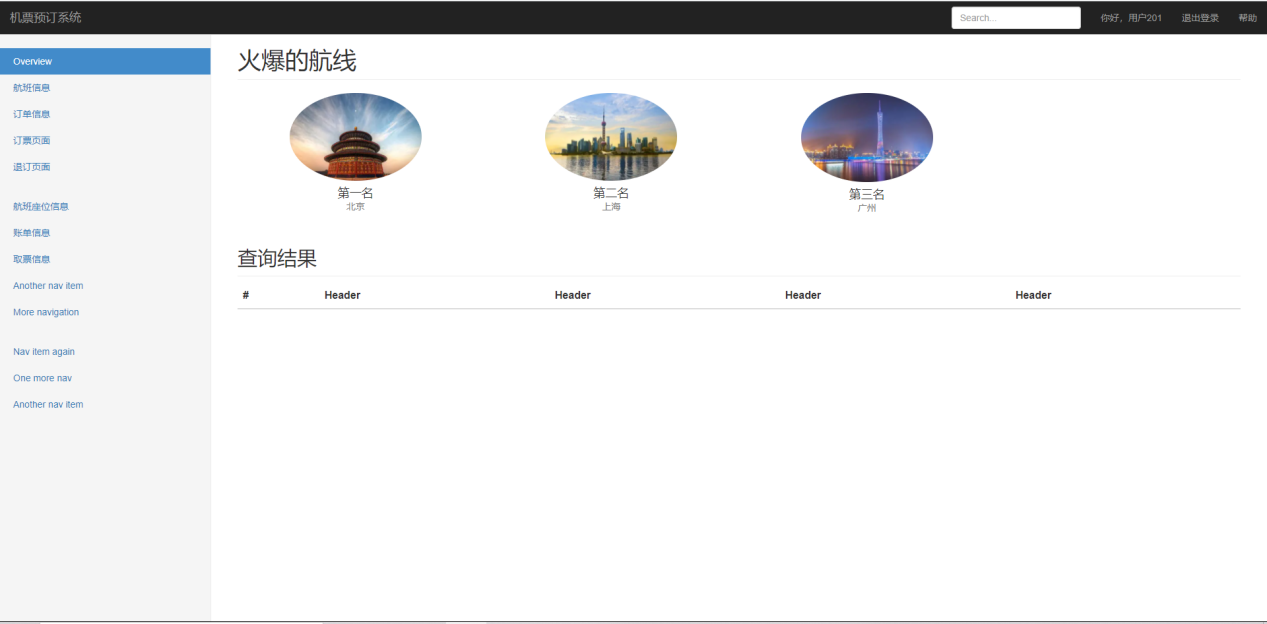


图4.19 用户登录主页

这里我们成功进入了用户主页，右上角会显示出当前的用户id201，说明用户登录功能正确。点击左侧的航班信息查看航班信息，可以通过航班的已订座位数和座位总数反映出满座率：

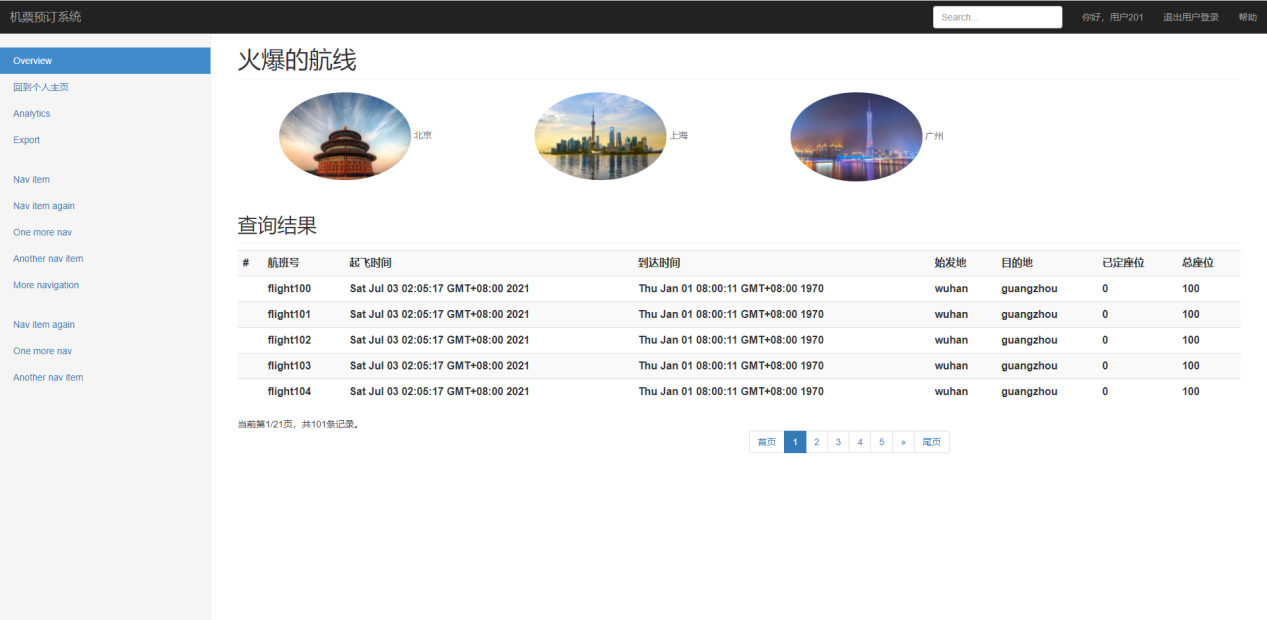


图4.20 用户航班信息查看

这里可以看到之前添加的所有的航班信息，接着点击左侧的按钮回到个人主页，接下来进入订票页面，输入出发地和目的地，系统自动搜索可以选择的行程：

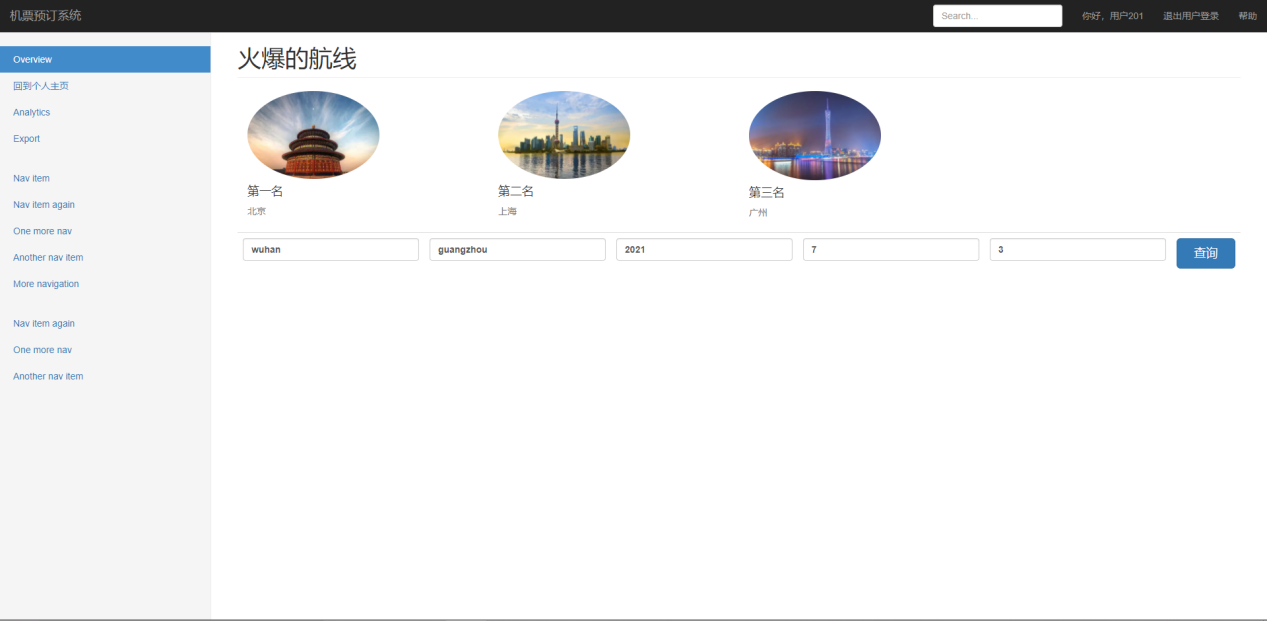


图4.21 行程查询航班页面

接下来页面会跳转进入订票界面，用户只用输入出发地和目的地以及航班编号就可以订票了：

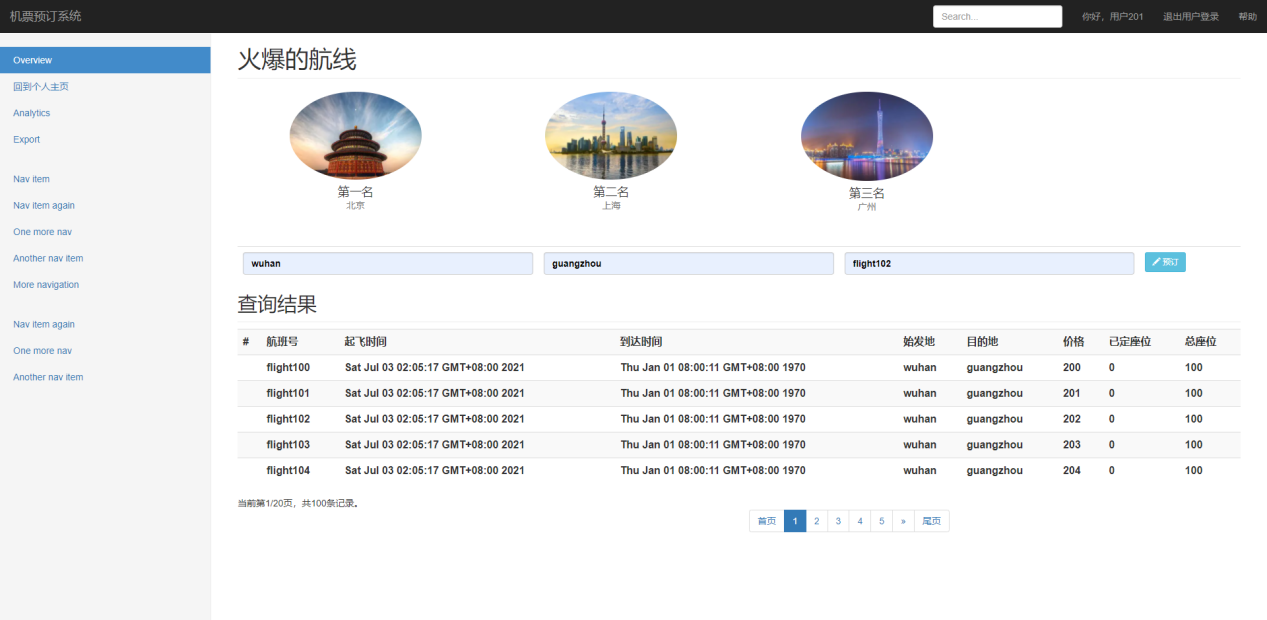


图4.22 订票页面

接下来查看该用户的订单页面和取票页面，可以看到两个表中和该订单有关的账单信息和取票信息成功添加到了对应的表中，说明之前设置的触发器正常工作了，为了验证，我们分别进入账单查看界面和取票信息查看界面进行确认：

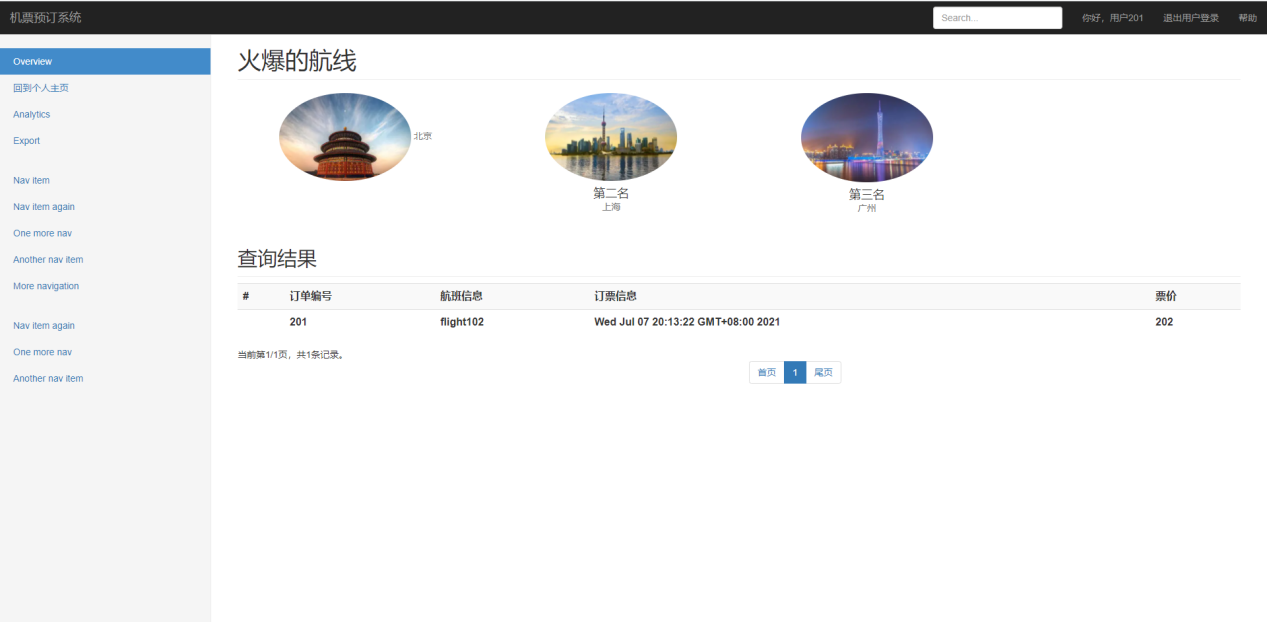


图4.23 订单查看页面

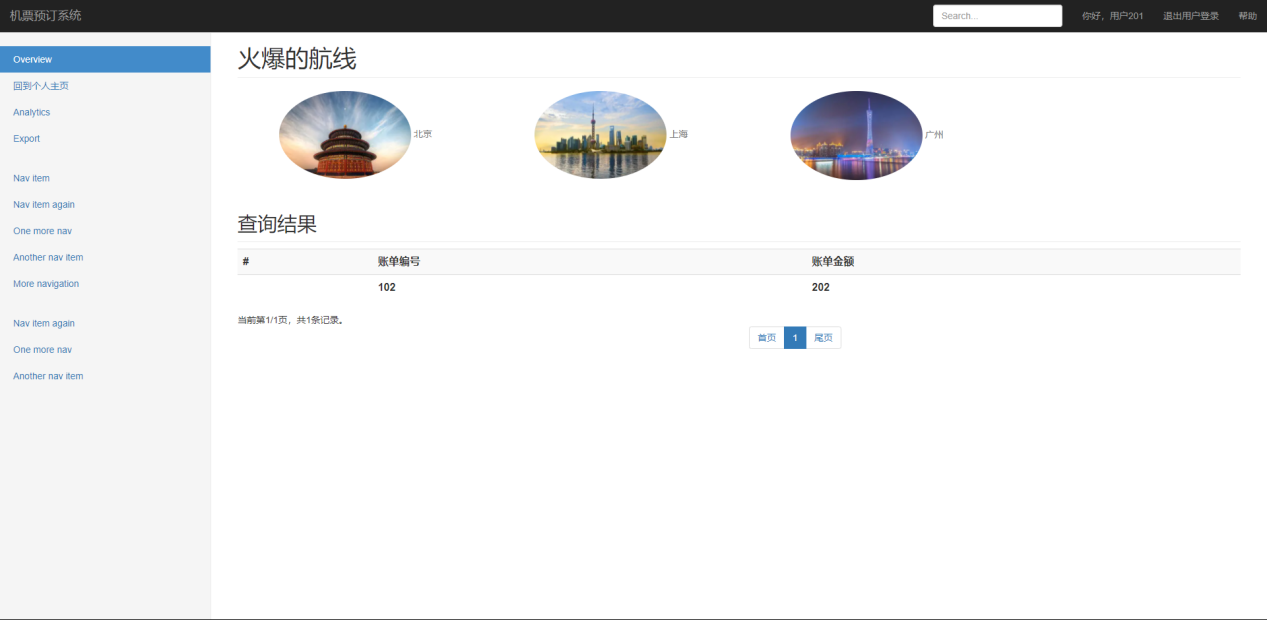


图4.24 账单信息页面

最后测试用户界面的退订页面，直接输入该订单的订单编号，最后查看订单编号，发现订单被成功删除了：

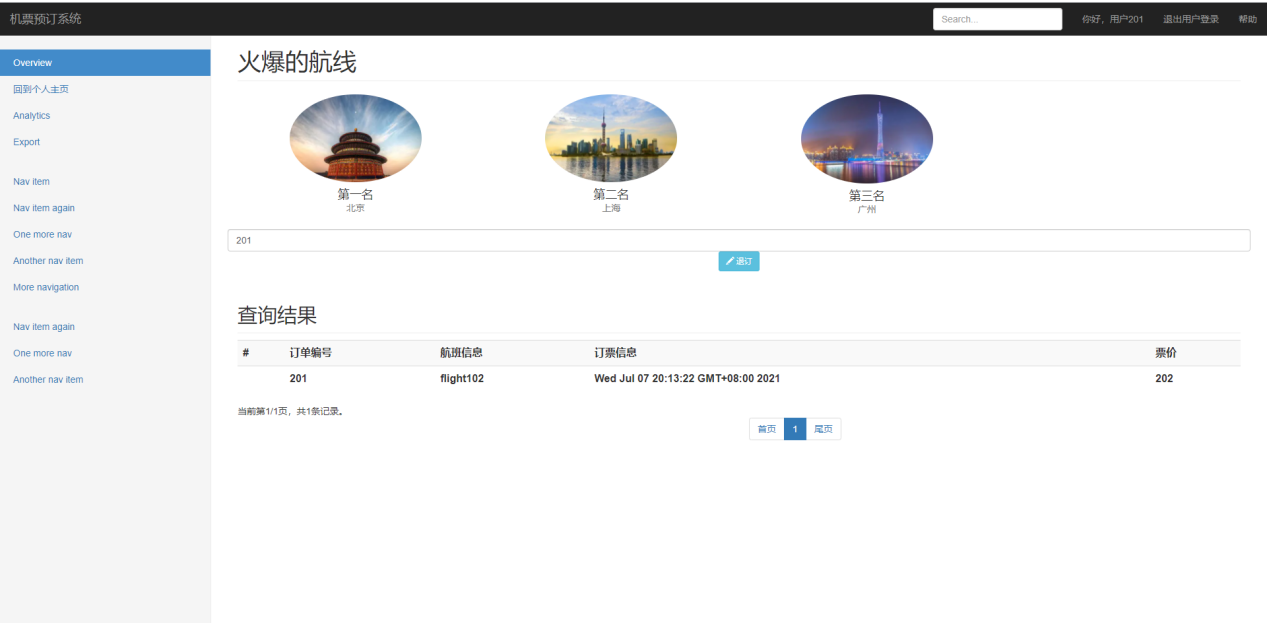


图4.25 退订页面

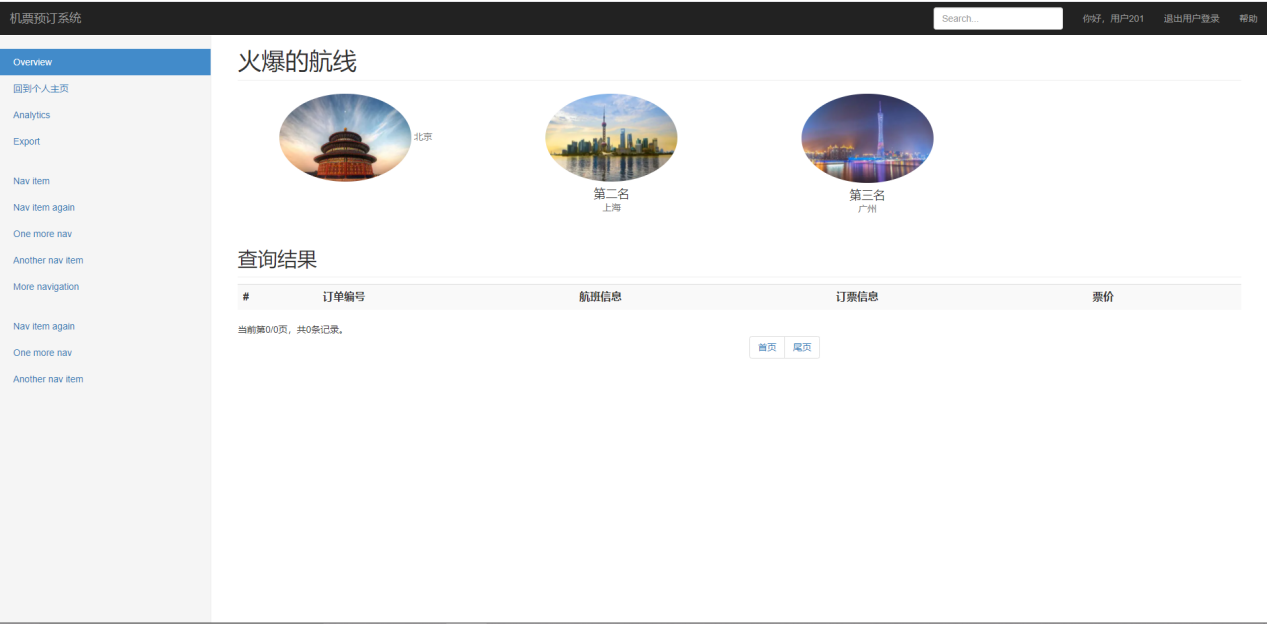


图4.26 成功退订页面验证

至此，所有的用户功能测试完成。

## 4.7 系统设计与实现总结

1. 完成了系统目标的总体设计与基本要求。
2. 对系统进行了需求分析，分析了功能需求，设计了数据字典以及数据流图等。
3. 对系统进行了总体设计，设计了功能模块组成以及业务流程等。
4. 对数据库进行了设计，绘制了ER图，设计了数据库的基本关系结构。
5. 根据系统的总体设计，利用ssm架构实现了前端与后端以及二者的交互过程，使得系统能够正常使用。
6. 在数据中使用sql语句完成了表的创建，完成了触发器的定义，建立了视图。
7. 实现了对关系表、尤其是航班信息表的增删查改功能。在查询航班信息时，实现了对于起飞时间的区间查询，不仅返回时间精确相同的航班，还返回其后一天内起飞、满足其他用户要求的航班。
8. 利用触发器实现了下订单时，账单信息、取票通知信息的创建，以及对应航班已订座位数加1。在退订单时要进行上述操作的逆操作，但是退订没有用触发器实现，尝试用后端实现，发现可以起到和触发器相同的效果。
9. 在开发过程中，使用了bootstrap的分页组件，完成了分页功能，并且使用了My97DatePicker包中的日期和时间选择插件，能够做到精准的插入航班的起飞时间和降落时间。

# 5 课程总结

数据库课程实验总体上分成了两个部分进行，一个是前期在educoder平台上的闯关任务，另一个是实验三制作一个完整的数据库系统并完整的实现出来。

第一个任务阶段难点是第一次编写sql语句，还不是很熟练，刚开始一直想直接在服务器上跑通，后来发现很长时间都找不到错误，最后在本地上先调试好了再往平台上提交成功率就高了不少。17关的查询语句训练让我提升了不少，印象最深刻的就是函数和触发器的编写，由于当时老师的课程还没有讲到那一章，我就去网上查阅相关资料，很快的就完成了关卡任务。在编写过程中，我发现了mysql语句相比于sql server的语法要求要宽松许多，自由度更大，当然我去网上查阅了资料后发现sql server的安全性做的比mysql要好很多，很少会发生删库无法恢复这种突发事件。在编写查询语句的时候开始只是想着把答案做出来就行，有的地方没有用规定的语句，后来按照题目要求修改了一遍，虽然结果不变，但是却让我知道怎样使用多种方法完成同样的查询目的。

第二个部分的任务是最困难的阶段了。数据库设计阶段我选择了机票预订系统作为题目，做出了ER图和数据流图并且设计好了数据字典并且考虑了相关的依赖，此外我专门花了4天时间在网上看视频学习SSM框架的使用，幸好这学期学会了java，在后端开发的过程中总的来说没有遇到特别多的麻烦，框架配置以及依赖包的版本问题困扰了我很久，等到整个框架搭建起来的时候已经是一周之后了，前端页面直接使用了bootstrap官网上的例子，官方文档给出了详细的使用方法和解释，于是我专门学习了一下如何添加一个分页功能，虽然看起来不起眼，但是实现过程却并不轻松。我还加入了详细时间选择模块，这里我开始准备使用bootstrap里面的，可是一直不能生效，最后我只能换成了My97DatePicker这个插件包完成了该项功能。页面跳转以及传参也一直是困扰我的问题，经常报错信息只有单纯的一个500，说传参出错，但是如何具体定位错误的位置就得我一个地方一个地方的慢慢试着打印信息找原因，此外，这个服务器每次启动一次都需要十几秒，调试起来也十分不方便，于是我在前端代码添加了一些代码加快了服务器的启动时间，帮助服务器快速定位了依赖资源位置。整个项目开发下来一共花费了我两个星期的时间，但是学会的东西还是很丰富的，既锻炼了我的程序开发能力，也锻炼了我对于一个小型项目工作模式的认识，并且结合了数据库方面的知识，个人成就感满满。

# 附录

drop database air\_ticket\_book\_management;

create database air\_ticket\_book\_management;

use air\_ticket\_book\_management;

-- 表一 管理员信息表

create table manager(

id int unique,

Username char(20) primary key,

Password char(20)

);

-- 表二 用户信息表

create table user\_info(

id int unique auto\_increment,

Username char(20) primary key,

Password char(20)

);

-- 表三 账单表

create table bills(

id int primary key auto\_increment,

P\_id int,

Pay int,

constraint person\_bill foreign key (P\_id) references user\_info(id) ON DELETE CASCADE ON UPDATE CASCADE

);

-- 表六 航班信息表

create table air\_info(

id char(20) primary key,

Takeoff\_time datetime,

Arrive\_time datetime,

origin\_stat char(20),

Term\_stat char(20),

price int,

bookedseats int,

totalseats int

);

-- 表四 订单表

create table book(

id int primary key auto\_increment,

P\_id int,

Air\_info char(20),

Booktime datetime,

Price int,

constraint air\_num foreign key (Air\_info) references air\_info(id),

constraint person\_book foreign key (P\_id) references user\_info(id) ON DELETE CASCADE ON UPDATE CASCADE

);

-- 表五 取票通知表

create table withdraw\_ticket(

id int primary key auto\_increment,

P\_id int,

Book\_id int,

Withdraw bool,

constraint booknumber foreign key (Book\_id) references book(id),

constraint personnumber foreign key (P\_id) references user\_info(id) ON DELETE CASCADE ON UPDATE CASCADE

);

-- 表七 航班座位表

create table air\_seat(

id int primary key auto\_increment,

Flight\_id char(20),

Seatnum int,

constraint flightid foreign key (Flight\_id) references air\_info(id)

);

set global time\_zone='+8:00';

DELIMITER $$

create trigger insert\_book

after insert

on book

for each row

begin

insert into bills(P\_id,Pay) values(new.P\_id,new.Price);

insert into withdraw\_ticket(P\_id,Book\_id,Withdraw) values(new.P\_id,new.id,false);

end;$$

DELIMITER ;

create view user as

select \*

from user\_info,air\_info,book,bill,withdraw\_ticket

where user\_info.id=book.P\_id and user\_info.id=bill.P\_id and user\_info=withdraw\_ticket.P\_id;

use air\_ticket\_book\_management;

insert into air\_seat values(100,null,10);