

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 机票预定系统 |
| 姓 名： | 孟彦康 |
| 专 业： | 计算机科学与技术（国际交流） |
| 班 级： | 校交1803班 |
| 学 号： | U201810824 |
| 指导教师： | 江胜 |

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

2021 年 6 月 28 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](#_Toc76425332)

[1.1 软件功能学习 1](#_Toc76425333)

[1.2 SQL练习 1](#_Toc76425334)

[1.3 数据库应用系统设计 1](#_Toc76425335)

[2软件功能学习 2](#_Toc76425336)

[2.1 任务要求 2](#_Toc76425337)

[2.2 完成过程 2](#_Toc76425338)

[2.2.1 环境配置 2](#_Toc76425339)

[2.2.2 数据备份 2](#_Toc76425340)

[2.2.3 增加用户以及设置权限 4](#_Toc76425341)

[2.3任务总结 5](#_Toc76425342)

[3 SQL练习 6](#_Toc76425343)

[3.1 任务要求 6](#_Toc76425344)

[3.1.1 建表 6](#_Toc76425345)

[3.1.2 数据更新 7](#_Toc76425346)

[3.1.3 查询 8](#_Toc76425347)

[3.1.4 了解系统的查询性能分析功能（选做） 9](#_Toc76425348)

[3.1.5 DBMS函数及存储过程（选做） 9](#_Toc76425349)

[3.2 完成过程 9](#_Toc76425350)

[3.2.1 建表 9](#_Toc76425351)

[3.2.2 数据更新 12](#_Toc76425352)

[3.2.3 查询 15](#_Toc76425353)

[3.2.4 了解系统的查询性能分析功能 28](#_Toc76425354)

[3.2.5 DBMS函数及存储过程 29](#_Toc76425355)

[3.3任务总结 30](#_Toc76425356)

[4 综合实践任务 31](#_Toc76425357)

[4.1系统设计目标 31](#_Toc76425358)

[4.2 需求分析 31](#_Toc76425359)

[4.2.1 功能需求 31](#_Toc76425360)

[4.2.2 性能需求 33](#_Toc76425361)

[4.2.3 数据流图 33](#_Toc76425362)

[4.2.4 数据完整性需求 33](#_Toc76425363)

[4.3 总体设计 34](#_Toc76425364)

[4.3.1 系统架构 34](#_Toc76425365)

[4.3.2 功能模块 34](#_Toc76425366)

[4.3.3 总体业务流程图 35](#_Toc76425367)

[4.4 数据库设计 35](#_Toc76425368)

[4.4.1 E-R图设计 35](#_Toc76425369)

[4.4.2 数据库逻辑结构设计 37](#_Toc76425370)

[4.5 详细设计与实现 39](#_Toc76425371)

[4.5.1 登录模块 39](#_Toc76425372)

[4.5.2 航班信息查询模块 40](#_Toc76425373)

[4.5.3 订票模块 41](#_Toc76425374)

[4.5.4 打印模块 42](#_Toc76425375)

[4.5.5 订单查询模块 42](#_Toc76425376)

[4.5.6 退票模块 43](#_Toc76425377)

[4.5.7 预定情况查询模块 44](#_Toc76425378)

[4.6 系统测试 45](#_Toc76425379)

[4.6.1 登录 45](#_Toc76425380)

[4.6.2 航班查询 45](#_Toc76425381)

[4.6.3 订票 47](#_Toc76425382)

[4.6.4 打印 51](#_Toc76425383)

[4.5.5 订单查询 52](#_Toc76425384)

[4.5.6 退票 54](#_Toc76425385)

[4.5.7 预定情况查询 56](#_Toc76425386)

[4.5.8 异常处理 56](#_Toc76425387)

[4.7 系统设计与实现总结 58](#_Toc76425388)

[5 课程总结 60](#_Toc76425389)

[5.1 工作总结 60](#_Toc76425390)

[5.2 心得体会 61](#_Toc76425391)

[5.3 展望 61](#_Toc76425392)

[附录 62](#_Toc76425393)

# 1 课程任务概述

## 1.1 软件功能学习

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。

## 1.2 SQL练习

创建数据库、创建表，如何完成数据更新、数据查询、了解系统查询性能分析功能、函数及存储过程和事务等相关任务。

## 1.3 数据库应用系统设计

拟定一个自己感兴趣的数据库应用系统题目，完成该小型数据库应用系统的设计与实现工作。主要内容包括：需求调研与分析、总体设计、数据库设计、详细设计与实现、测试等环节的工作。

# 软件功能学习

## 2.1 任务要求

完成下列1~2题，并在实践报告中叙述过程，可适当辅以插图（控制在A4三页篇幅以内）

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。

## 2.2 完成过程

### 2.2.1 环境配置

操作系统：Windows 10 家庭中文版 19042.1052，

数据库：MySQL 8.0.20

图形界面：MySQL Workbench

### 2.2.2 数据备份

1) 数据和日志文件的脱机备份

调用 mysqldump 程序来进行指定数据库中数据与结构向外部文本文件的保存过程。

如图 2.1所示，执行如下命令，输入数据库 root 密码后，进行完全数据备份：

.\mysqldump.exe airinfo -uroot -p > back.sql

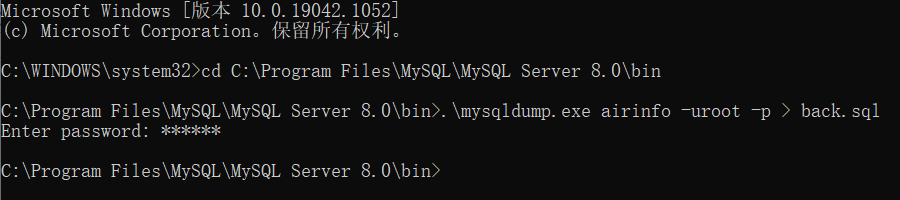


图 2.1 执行备份命令

导出的SQL文件如图 2.2所示。

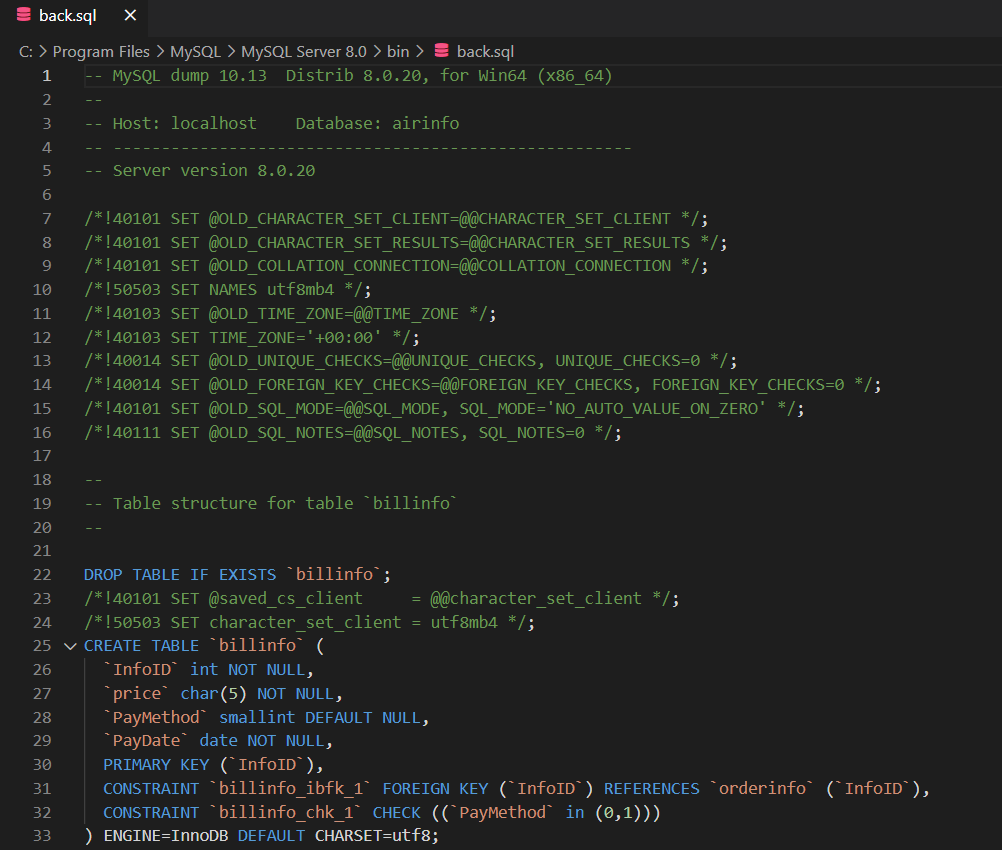


图 2.2 导出的 SQL文件

2) 系统的备份功能

打开 MySQL Workbench，并连接上数据库，执行 Data Export，如图 2.3所示。

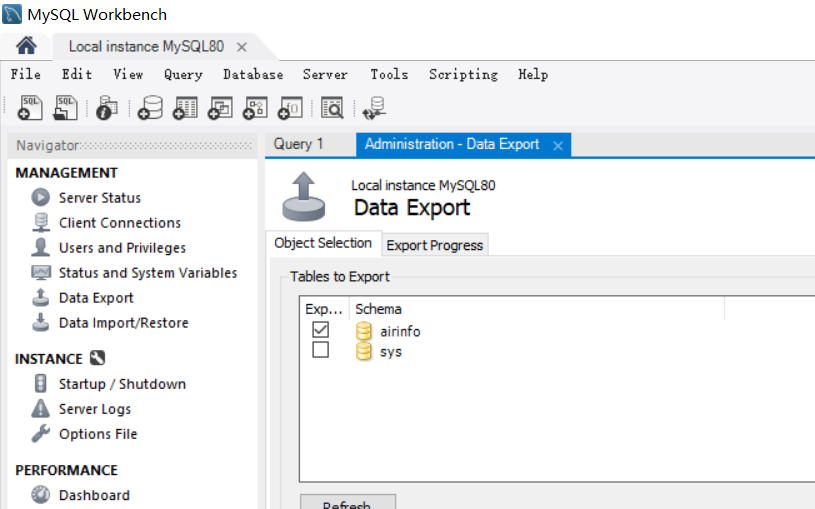


图 2.3 导出数据

执行导出任务成功，如图 2.4所示。

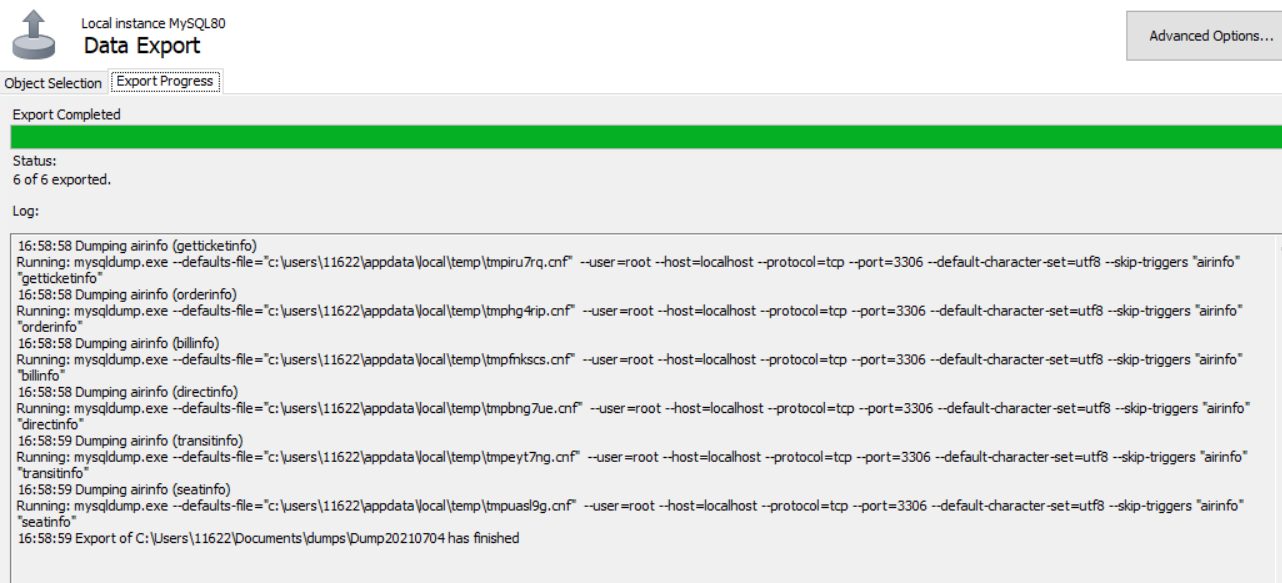


图 2.4 导出记录

导出的结果如图 2.5所示。

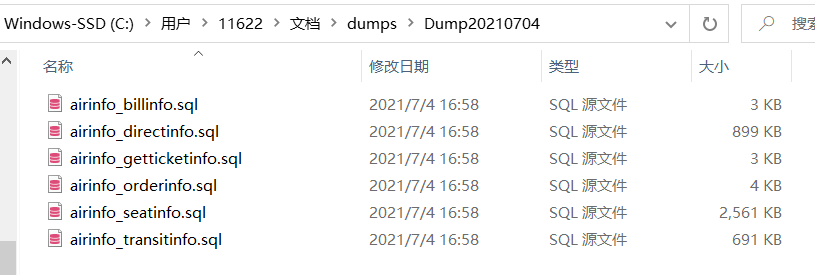


图 2.5 数据导出的结果

### 2.2.3 增加用户以及设置权限

首先创建一个实验用的用户，执行如下命令：

create user wcx@localhost IDENTIFIED BY 'password';

接下来对用户wcx授权，使其获得查询 airinfo 数据库的所有表，但不能 执行其他的操作，执行如下命令：

grant select on airinfo.\* TO 'wcx'@'localhost';

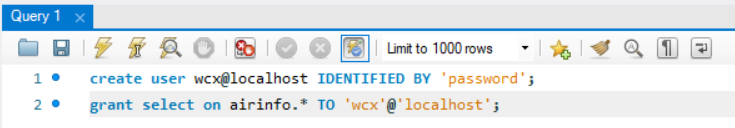


图 2.6 执行命令

接下来以用户wcx的身份进入数据库，如图 2.7所示。

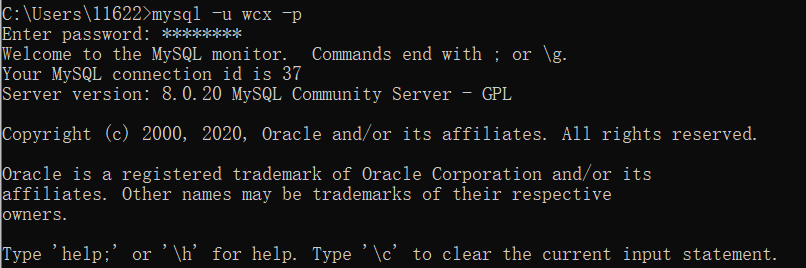


图 2.7 登入数据库

执行查询操作，成功，如图 2.8所示。

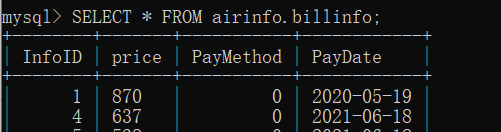


图 2.8 查询成功

执行插入操作，失败，如图 2.9所示。



图 2.9 插入失败

与预期的结果一致，查询成功但插入失败，失败时的输出为 command denied， 符合该用户的权限。

## 2.3任务总结

在这一个任务中，我验熟悉了数据库的基础操作和基础结构，由于是Windows系统，我选择了使用MySQL Workbench图形界面，这一个任务是对软件的学习，相对较简单。我学会了使用mysqldump进行备份，掌握备份的方法和技能对于今后的更深入更具体更现实的开发是不可或缺的技能，对于自己某些时候误删误改的数据常备份也是好习惯。

# 3 SQL练习

## 3.1 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。

### 3.1.1 建表

1. **根据以上背景，请在DBMS中创建这些关系，包括主码和外码的说明，并写出指定关系的建表SQL语句**

**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)

主码为隔离房间编号

1. **观察性实验**

验证在设置外码时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

1. **数据准备**

依据后续实验的要求，向上述表格中录入适当数量的实验数据，从而对相关的实验任务能够起到验证的作用。

### 3.1.2 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”。

### 3.1.3 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】，查询结果如图 3.1所示）

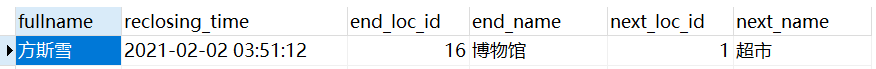


图 3.1查询结果

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

### 3.1.4 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。

### 3.1.5 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。
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。

## 3.2 完成过程

实验环境如下：

操作系统：Windows 10 家庭中文版 19042.1052，

数据库：MySQL 8.0.20

图形界面：MySQL Workbench

### 3.2.1 建表

通过编写对应的SQL语句来生成所需要的表，SQL语句如下：

create database covid19mon;

use covid19mon;

-- 表1 人员表(person)

CREATE TABLE person(

id int NOT NULL,

fullname char(20) NOT NULL,

telephone char(11) NOT NULL,

CONSTRAINT pk\_person PRIMARY KEY(id)

);

-- 表2 地点表(location)

CREATE TABLE location(

id int NOT NULL,

location\_name char(20) NOT NULL,

CONSTRAINT pk\_location PRIMARY KEY(id)

);

-- 表3 行程表（itinerary）

CREATE TABLE itinerary(

id int not null,

p\_id int,

loc\_id int,

s\_time datetime,

e\_time datetime,

CONSTRAINT pk\_itinerary PRIMARY KEY(id),

CONSTRAINT fk\_itinerary\_pid FOREIGN KEY(p\_id)

REFERENCES person(id),

CONSTRAINT fk\_itinerary\_lid FOREIGN KEY(loc\_id)

REFERENCES location(id)

);

-- 表4 诊断表（diagnose\_record）

CREATE TABLE diagnose\_record(

id int,

p\_id int,

diagnose\_date datetime,

result int,

CONSTRAINT pk\_diagnose\_record PRIMARY KEY(id),

CONSTRAINT fk\_diagnose\_pid FOREIGN KEY(p\_id)

REFERENCES person(id)

);

-- 表5 密切接触者表（close\_contact）

CREATE TABLE close\_contact(

id int,

p\_id int,

contact\_date datetime,

loc\_id int,

case\_p\_id int,

CONSTRAINT pk\_close\_contact PRIMARY KEY(id),

CONSTRAINT fk\_contact\_pid FOREIGN KEY(p\_id)

REFERENCES person(id),

CONSTRAINT fk\_contact\_lid FOREIGN KEY(loc\_id)

REFERENCES location(id),

CONSTRAINT fk\_contact\_caseid FOREIGN KEY(case\_p\_id) REFERENCES person(id)

);

-- 表6 隔离地点表（isolation\_location）

CREATE TABLE isolation\_location(

id int,

location\_name char(20),

capacity int,

CONSTRAINT pk\_isolation\_loc PRIMARY KEY(id)

);

-- 表7 隔离表（isolation\_record）

CREATE TABLE isolation\_record(

id int,

p\_id int,

s\_date datetime,

e\_date datetime,

isol\_loc\_id int,

state int,

CONSTRAINT pk\_isolation PRIMARY KEY(id),

CONSTRAINT fk\_isolation\_pid FOREIGN KEY(p\_id)

REFERENCES person(id),

CONSTRAINT fk\_isolation\_lid FOREIGN KEY(isol\_loc\_id)

REFERENCES isolation\_location(id)

);

SQL语句正确，见表成功，提示信息如图 3.2所示。

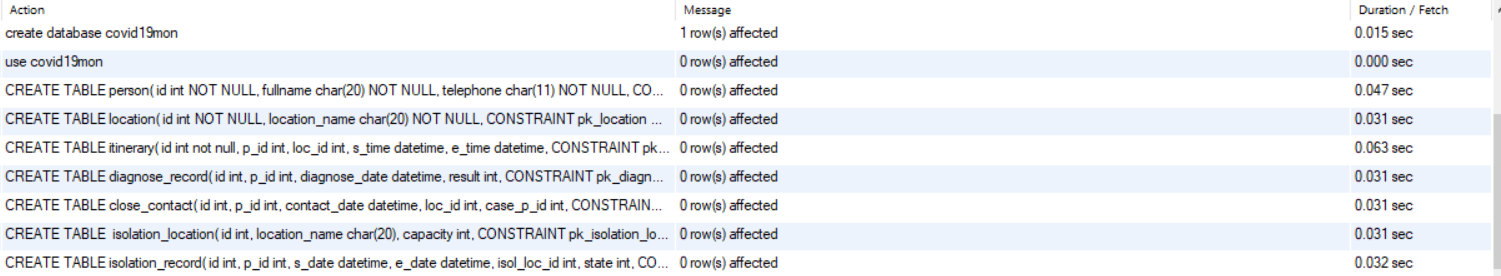


图 3.2 提示信息

建表的结果如图 3.3所示，可以看出七张表都已经存在。

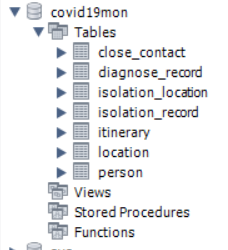


图 3.3 建表结果

### 3.2.2 数据更新

1. **SQL 语句增删改**

* **数据插入**

向人员表person插入以下3条数据:

-- ( 1, 张小敏,13907110001)

-- ( 2, 李大锤,18907110002)

-- ( 3, 孙二娘,13307100003)

对应的SQL语句如下：

use covid19mon;

INSERT person(id,fullname,telephone)

VALUES(1, '张小敏',13907110001);

INSERT person(id,fullname,telephone)

VALUES(2, '李大锤',18907110002);

INSERT person(id,fullname,telephone)

VALUES(3, '孙二娘',13307100003);

插入的结果如图 3.4所示，可以看出插入成功。

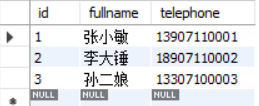


图 3.4 插入结果

* **数据删除**

-- 从人员表person中删除以下数据:

-- ( 2, 李大锤,18907110002)

语句为：DELETE FROM person where id=2;

执行结果如图 3.5所示，可以看出删除成功。



图 3.5 删除结果

* **修改数据**

将人员表person中1号人员的电话更改成13607176668，SQL语句为：

UPDATE person SET telephone='13607176668' WHERE id=1 AND fullname='张小敏';

执行结果如图 3.6所示，可以看出修改成功。



图 3.6 修改结果

1. **批处理操作**

首先建立新表location\_record\_2：

CREATE TABLE location\_record\_2(id int not null,p\_id int,loc\_id int,s\_time datetime, e\_time datetime);

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

insert into location\_record\_2 select \* from itinerary where loc\_id=2;

结果如图 3.7所示。

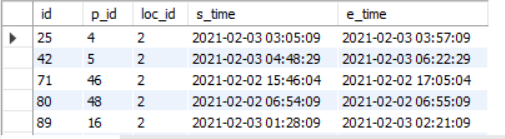


图 3.7 批处理结果

1. **数据导入导出**

使用 MySQL Workbench 的 Data Exprot 导出数据，然后新创建一个模式

new\_schema 并导入数据，得到如图 3.8所示的结果。

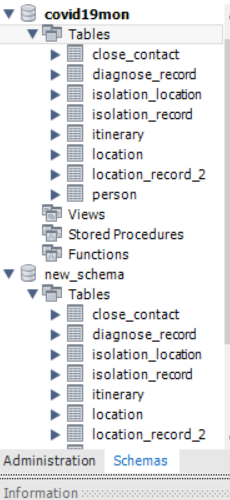


图 3.8 导入结果

1. **观察性实验**

MySQL中数据库表中如果没有设置primary key,在workbench中，无法直接编辑数据，必须设置PK和NN。

1. **触发器实验**

当隔离表（isolation\_record）中的某位隔离人员在诊断表（diagnose\_record）中的诊断结果（result）为1(新冠确诊)时，自动将隔离表中的隔离状态（state)改成3（转入医院）。

SQL语句如下：

DROP TRIGGER IF EXISTS trigger\_u;

DROP TRIGGER IF EXISTS trigger\_i;

DELIMITER ||

CREATE TRIGGER trigger\_u after update ON diagnose\_record FOR EACH ROW

BEGIN

IF

NEW.result = 1 THEN

UPDATE isolation\_record

SET isolation\_record.state = 3

WHERE

isolation\_record.p\_id = NEW.p\_id ;

END IF;

END||

CREATE TRIGGER trigger\_i after INSERT ON diagnose\_record FOR EACH ROW

BEGIN

IF

NEW.result = 1 THEN

UPDATE isolation\_record

SET isolation\_record.state = 3

WHERE

isolation\_record.p\_id = NEW.p\_id ;

END IF;

END||

DELIMITER;

### 3.2.3 查询

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

SELECT location\_name, count(\*) AS visitors FROM itinerary JOIN location ON itinerary.loc\_id = location.id

GROUP BY location.location\_name

HAVING count(\*)> 30

ORDER BY count(\*) DESC, location\_name ;

查询结果如图 3. 9所示。



图 3.9 查询结果1

1. 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

SELECT

isolation\_location.location\_name,

count(\*) AS number

FROM isolation\_record JOIN isolation\_location ON isolation\_record.isol\_loc\_id = isolation\_location.id WHERE isolation\_record.state = 1

GROUP BY

isolation\_location.location\_name

ORDER BY

count(\*) DESC,

isolation\_location.location\_name;

查询结果如图 3.10所示。

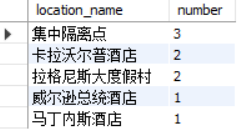


图 3.10 查询结果2

1. 查询行程表中连续行程中的一对行程地点

SELECT

person.id,

person.fullname,

person.telephone,

s\_I.e\_time AS reclosing\_time,

s\_loc.id AS loc1,

s\_loc.location\_name AS address1,

e\_loc.id AS loc2,

e\_loc.location\_name AS address2

FROM

itinerary AS s\_I,

itinerary AS e\_I,

location AS s\_loc,

location AS e\_loc,

person

WHERE

s\_I.p\_id = person.id

AND e\_I.p\_id = person.id

AND s\_I.e\_time = e\_I.s\_time

AND s\_I.loc\_id = s\_loc.id

AND e\_I.loc\_id = e\_loc.id

AND person.id > 30

ORDER BY

person.id,reclosing\_time;

查询结果如图 3.11所示。



图 3.11 查询结果3

1. 查询充珉瑶和贾涵山的行程情况

SELECT

person.fullname,

person.telephone,

location.location\_name,

itinerary.s\_time,

itinerary.e\_time

FROM

person

LEFT JOIN itinerary ON itinerary.p\_id = person.id

LEFT JOIN location ON location.id = itinerary.loc\_id

WHERE

person.fullname = '充珉瑶'

OR person.fullname = '贾涵山'

ORDER BY

person.id DESC,

itinerary.s\_time;

查询结果如图 3.12所示。



图 3.12 查询结果4

1. 查询地名中带有‘店’字的地点名称

SELECT

location.id,

location.location\_name

FROM

location

WHERE

location.location\_name LIKE '%店%'

ORDER BY

location.id;

查询结果如图 3.13所示。



图 3.13 查询结果5

1. 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

SELECT

person.fullname,

person.telephone

FROM

person

JOIN itinerary ON person.id = itinerary.p\_id

JOIN location ON location.id = itinerary.loc\_id

WHERE

location\_name = '活动中心'

AND s\_time < '2021-02-02 21:25:40' AND e\_time > '2021-02-02 20:05:40'

ORDER BY

person.fullname;

查询结果如图 3.14所示。



图 3.14 查询结果6

1. 查询正在使用的隔离区名（使用DISTINCT关键字）

SELECT

isolation\_location.location\_name

FROM

isolation\_location

WHERE

isolation\_location.id IN ( SELECT isolation\_record.isol\_loc\_id FROM isolation\_record WHERE isolation\_record.state = 1 )

ORDER BY

isolation\_location.id;

查询结果如图 3.15所示。

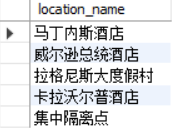


图 3.15 查询结果7

1. 查询有出行记录的人员（使用EXISTS关键字）

SELECT

person.fullname,

person.telephone

FROM

person

WHERE

EXISTS ( SELECT \* FROM itinerary WHERE itinerary.p\_id = person.id )

ORDER BY

person.id

LIMIT 30;

查询结果如图 3.16所示。



图 3.16 查询结果8

1. 查询没有达到地点“Today便利店”的人数（使用NOT EXISTS 关键字）

SELECT

count(\*) AS number

FROM

person

WHERE

NOT EXISTS (

SELECT\*

FROM

itinerary

JOIN location ON location.id = itinerary.loc\_id

WHERE

location.location\_name = 'Today便利店'

AND itinerary.p\_id = person.id

);

查询结果如图 3.17所示。



图 3.17 查询结果9

1. 查询去过所有地点的人员

SELECT

person.fullname

FROM

person

WHERE

NOT EXISTS (

SELECT\*

FROM

location

WHERE

NOT EXISTS ( SELECT \* FROM itinerary WHERE itinerary.loc\_id = location.id AND itinerary.p\_id = person.id )

)

ORDER BY person.fullname;

查询结果如图 3.18所示。

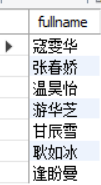


图 3.18 查询结果10

1. 新建一个视图，里面有每个隔离点的id,名称，已用房间，容量

DROP VIEW

IF EXISTS isolation\_location\_status;

CREATE VIEW isolation\_location\_status AS SELECT

isolation\_location.id,

isolation\_location.location\_name,

isolation\_location.capacity,

sum( if(isolation\_record.state = 1,1,0) ) AS occupied

FROM

isolation\_location

JOIN isolation\_record ON isolation\_location.id = isolation\_record.isol\_loc\_id

GROUP BY

isolation\_location.id

结果如图 3.19所示。

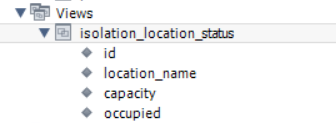


图 3.19 查询结果11

1. 从视图中查询还有空房间的隔离点的剩余房间数目

SELECT isolation\_location\_status.location\_name,( isolation\_location\_status.capacity - isolation\_location\_status.occupied ) AS available\_rooms

FROM

isolation\_location\_status

ORDER BY

isolation\_location\_status.id

查询结果如图 3.20所示。



图 3.20 查询结果12

1. 查询靳宛儿传染的人有哪些

SELECT

person.fullname,

person.telephone

FROM person

JOIN itinerary ON itinerary.p\_id = person.id,(

SELECT

itinerary.loc\_id,

itinerary.s\_time,

itinerary.e\_time

FROM

person

JOIN itinerary ON itinerary.p\_id = person.id

WHERE

person.fullname = '靳宛儿'

) AS TTable

WHERE

person.fullname != '靳宛儿'

AND itinerary.loc\_id = TTable.loc\_id

AND itinerary.s\_time < TTable.e\_time AND itinerary.e\_time > TTable.s\_time

ORDER BY

person.fullname;

查询结果如图 3.21所示。



图 3.21 查询结果13

1. 查询每个地点的密切接触者的数量

SELECT

location.location\_name,

count(\*) AS close\_contact\_number

FROM

location

JOIN close\_contact ON close\_contact.loc\_id = location.id

GROUP BY

location.id

ORDER BY

close\_contact\_number DESC,

location\_name;

查询结果如图 3.22所示。



图 3.22 查询结果14

1. 查询感染人数最多的用户的名字和感染人数

SELECT

close\_contact.case\_p\_id,

person.fullname,

count(\*) AS infected\_number

FROM

close\_contact

JOIN person ON person.id = close\_contact.case\_p\_id

GROUP BY

close\_contact.case\_p\_id

ORDER BY infected\_number DESC

LIMIT 1;

查询结果如图 3.23所示。



图 3.23 查询结果15

1. 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

SELECT

person.fullname,

count(\*) AS record\_number

FROM

person

JOIN itinerary ON itinerary.p\_id = person.id

WHERE

itinerary.e\_time >= '2021-02-02 10:00:00'

AND itinerary.s\_time <= '2021-02-02 14:00:00'

GROUP BY

person.id

ORDER BY

record\_number DESC,

person.fullname

LIMIT 3;

查询结果如图 3.24所示。

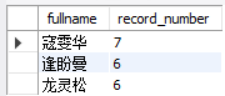


图 3.24 查询结果16

1. 房间数第2多的隔离点

SELECT

isolation\_location.location\_name,

isolation\_location.capacity

FROM

isolation\_location

WHERE

isolation\_location.capacity < (SELECT MAX(

isolation\_location.capacity)

FROM isolation\_location)

LIMIT 1;

查询结果如图 3.25所示。



图 3.25 查询结果17

### 3.2.4 了解系统的查询性能分析功能

选择第10个查询，执行下列explain语句。

explain SELECT

person.fullname

FROM

person

WHERE

NOT EXISTS (

SELECT

\*

FROM

location

WHERE

NOT EXISTS ( SELECT \* FROM itinerary WHERE itinerary.loc\_id = location.id AND itinerary.p\_id = person.id )

)

ORDER BY person.fullname;

结果如图 3.26所示。

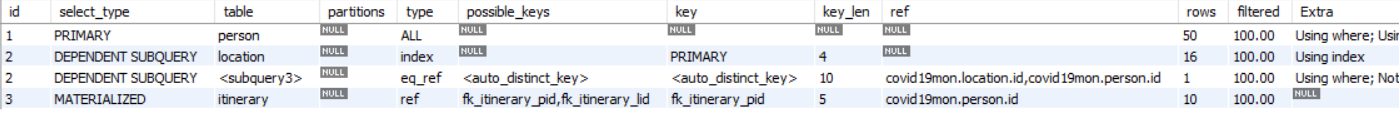


图 3.26 EXPLAIN结果

其中，id 代表操作的顺序，数字大的先执行，对于数字相同的从上到下的顺序执行。

select\_type 中，PRIMARY 代表一个需要 union 操作或者含有子查询的 select，

MATERIALIZED 代表物化视图子查询。

table 代表执行的表格。

type 代表查询结果类型，ALL 是全表扫描，ref 是对于每个来自前面的表的记录，所有匹配的行从这张表中取出。

possible\_key：查询可能用到的索引。

key：查询真正用到的索引。

ref：连接查询时显示驱动表的关联字段，

rows：执行计划中估算的扫描行数，不是精确值。

Extra：显示额外信息，如 Using temporary：使用了临时表存储中间结果，Using where：使用了 where 进行筛选，Distinct：使用了 distinct关键字等。

### 3.2.5 DBMS函数及存储过程

(1)依据人员编号计算其到达所有地点的次数(即行程表中的记录数)。

use covid19mon;

set global log\_bin\_trust\_function\_creators=1;

DELIMITER ;;

create function Count\_Records(x int)

returns int

BEGIN

declare number\_id int default 0;

set number\_id = (select count(\*) from itinerary where p\_id = x);

return number\_id;

end;;

DELIMITER ;

(2) 利用创建的函数，仅用一条SQL语句查询在行程表中至少有3条行程记录的人员信息，查询结果依人员编号排序。

select \*

from person

having Count\_Records(id) > 2

order by id;

## 3.3任务总结

通过这个任务，极大地增强了我对 SQL 的熟练程度，各个任务难度循序渐进，有启发意义，从最开始的简单单表多表查询，到最后的复杂连接查询嵌套查询，在练习中我再次巩固了大量的 SQL 用法和规则。

另外，我注意到，获得相同结果的 SQL 查询脚本的查询过程和性能会有很大的差别，在较大的数据基数上，这样的时间空间差距会进一步被扩大，因此优化查询脚本是必要的。

本任务让我对于一个数据库项目该如何考虑问题有了初步的认识，熟悉了如何抽象一个模型来建立数据之间的关系，在后面的实验中，这也将是我重点关注的。同时，本任务也让我对于非查询的应用，比如触发器等有了一定的了解，在后续的实验中可能会应用到这些。

# 4 综合实践任务

## 4.1系统设计目标

我设计的是机票预订系统，可以为机场订票和退票等业务操作提供支持，尽可能真实的模拟了机场的实际业务情况。用户进入系统之后，可以根据需求订购15天内的机票，同时持取票、退票、查看航班预订情况等功能。

**1、系统功能的基本要求：**

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。
8. 在程序中需要体现SQL和编程语言的结合。

**2、数据库要求：在数据库中至少应该包含下列数据表：**

1. 航班信息表；
2. 航班坐位情况表；
3. 旅客订票信息表；
4. 取票通知表；
5. 帐单。

设计一个B/S或C/S模式的系统实现上述功能。

## 4.2 需求分析

### 4.2.1 功能需求

本系统尽可能真实的模拟了机场的网上机票预定系统，每个功能都保证尽可能贴近现实，下面详细说明各项需求。

1）登录

用户能够使用自己的信息（用户名和密码）登录进入系统，然后执行自己的操作。不同用户的账号和密码保存至系统中，如果对应不上会提示错误。

2）航班信息查询

用户可以选择不同的出发地或者目的地，同时根据各种条件来进行筛选，比如出发日期，票价，以及航空公司等。

3）订票功能

用户在订票时需要输入姓、名、身份证号码，然后选择通过微信或者支付宝来进行支付，同时界面上要显示出用户选择的完整航班信息。

4）打印功能

用户在订完机票之后，可以点击机票打印按钮，输入对应的账单编号和取票密码，查询到自己所买机票的状态（已打印/未取票/已退订），选择进行打印。这里模拟了真实场景下购票的流程，将网上购票和取纸质票的流程分离，也降低了耦合。

5）订单查询功能

用户还可以直接通过自己的身份证号码来查询自己购买的所有机票，查看这些机票的详细信息，以及机票的状态（已打印/未取票/已退订）。这个功能使得用户查询历史机票更加方便，不必记住所有的订单编号和密码。

6）退票功能

用户需要输入身份证号码和订单号，匹配成功即可退票。只有未打印（未取票）的票才能够退票，打印之后就不能够再退票了。这里也模拟了真实场景，网上买票的话只有未取出的票才能在网上退，已经取了的票需要到机场线下退订。

7）预定情况查询功能

系统内存储了每个航班的座位情况，用户可以订购未满航班的座位，订购后剩余空座位会减少1。用户可以查询指定航班的座位情况，指定航班的座位情况通过饼图的形式直观展示，用户可以据此选择是否预定。这里机票的编号也模拟真实的飞机座位编号，采用字母+数字的组合。

功能如图 4.1所示。

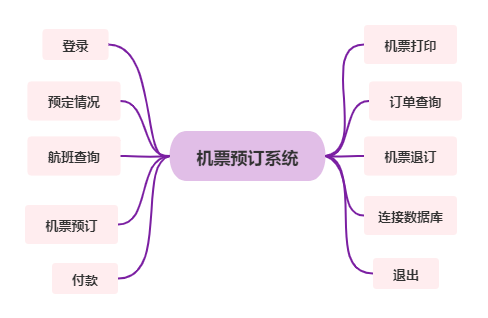


图 4.1 系统的各项功能

### 4.2.2 性能需求

我只使用较少的数据来进行了测试，测试时执行速度很快，没有考虑高并发场景，可以支持较小的数据需求。真实系统中应当加入并行的设计支持。

### 4.2.3 数据流图

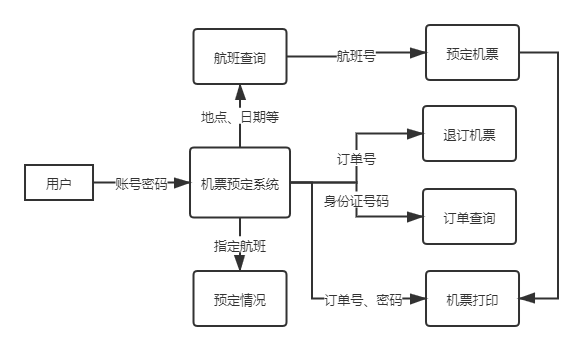


图 4.2 数据流图

### 4.2.4 数据完整性需求

这里只对数据完整性的需求给出一个大概的分析，具体如何转化为数据库的表示之后的章节给出。

1. 实体完整性约束

用户买的机票对应的订单，需要有唯一的订单编号能对它们进行区别，订单号是主码，其值应该非空且唯一。

1. 参照完整性约束

比如取票和账单都需要有对应的订单等。

1. 用户定义完整性约束
2. 支付方式只能选择微信或者支付宝；
3. 身份证号码是18位数字。

## 4.3 总体设计

### 4.3.1 系统架构

系统采用的是B/S结构，如图 4.3所示。



图 4.3 B/S架构图

系统总体使用Python编写，前端采用的是PyQt5作为组件，后端使用MySQL Workbench+MySQL实现，使用Pycharm作为设计工具。

### 4.3.2 功能模块

1）登录模块

用户能够使用自己的信息（用户名和密码）登录进入系统主界面，然后执行自己的操作。不同用户的账号和密码保存至系统中，如果对应不上会提示错误。

2）航班信息查询模块

本系统模拟的是一个特定的机场，有入港航班（飞到这个机场来的航班）和出港航班（从这个机场出发的航班），用户可以选择不同的出发地或者目的地，同时系统也支持根据各种条件来进行筛选，比如出发日期，票价，以及航空公司等。查询到的结果通过选项卡组件呈现，用户在这个组件里直接右击对应的航班，再点击“订票”，即可进入订票界面。

3）订票模块

用户在订票时需要输入姓、名、身份证号码，然后选择通过微信或者支付宝来进行支付，同时界面上会显示出用户选择的完整航班信息。点击确定完成订票。

4）打印模块

用户在订完机票之后，可以查看账单编号以及取票密码。然后用户可以进入主菜单中，点击机票打印按钮，输入对应的账单编号和取票密码，就能够查询自己所买机票的状态（已打印/未取票/已退订），然后可以选择进行打印。这里模拟了真实场景下购票的流程，只能打印一次，而且将网上购票和取纸质票的流程分离，也降低了耦合。

5）订单查询模块

用户还可以直接通过自己的身份证号码来查询自己购买的所有机票，查看这些机票的详细信息，以及机票的状态（已打印/未取票/已退订）。这个功能使得用户查询历史机票更加方便，不必记住所有的订单编号和密码。

6）退票模块

用户需要输入身份证号码和订单号，匹配成功即可退票。只有未打印（未取票）的票才能够退票，打印之后就不能够再退票了。这里也模拟了真实场景，网上买票的话只有未取出的票才能在网上退，已经取了的票需要到机场线下退订。

7）预定情况查询模块

系统内存储了每个航班的座位情况，用户可以订购未满航班的座位，订购后剩余空座位会减少1。用户可以查询指定航班的座位情况，指定航班的座位情况通过饼图的形式直观展示，用户可以据此选择是否预定。这里机票的编号也模拟了真实的飞机座位编号，采用字母+数字的组合。

### 4.3.3 总体业务流程图

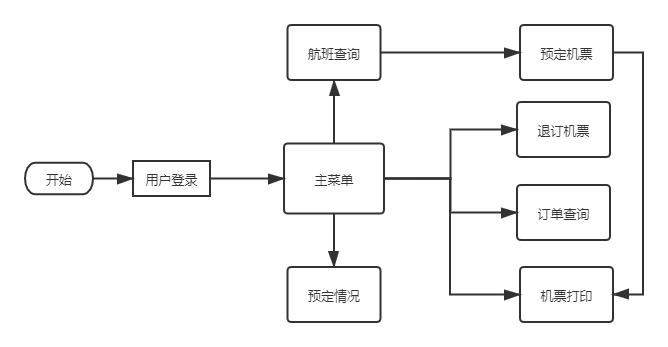


图 4.4 总体业务流程

## 4.4 数据库设计

### 4.4.1 E-R图设计

我采用MySQL Workbench来生成“乌鸦脚”式的E-R图。如图 4.5所示。

一对一以及一对多等联系都在上图当中清晰可见。具体为：

1）航班和航班座位表是一对一的关系，一个航班对应一个航班座位信息。

2）航班信息和订单信息是一对多的关系，一个航班信息可以对应多条订单信息。

3）订单信息和取票通知表是一对一的关系，一个订单信息记录对应于一条取票通知。

4）订单信息和账单也是一对一的关系，一条订单信息对应于一个账单信息。

这里的E-R图是一个基本的示意，实际建表的时候，考虑到了实际情况的复杂性，又区分了直达航班和中转航班，使得实际的航班信息表变成了两张表，这样E-R图会稍有不同，但是对应关系是一样的。这里为了便于理解对应关系，省去了中转航班表。

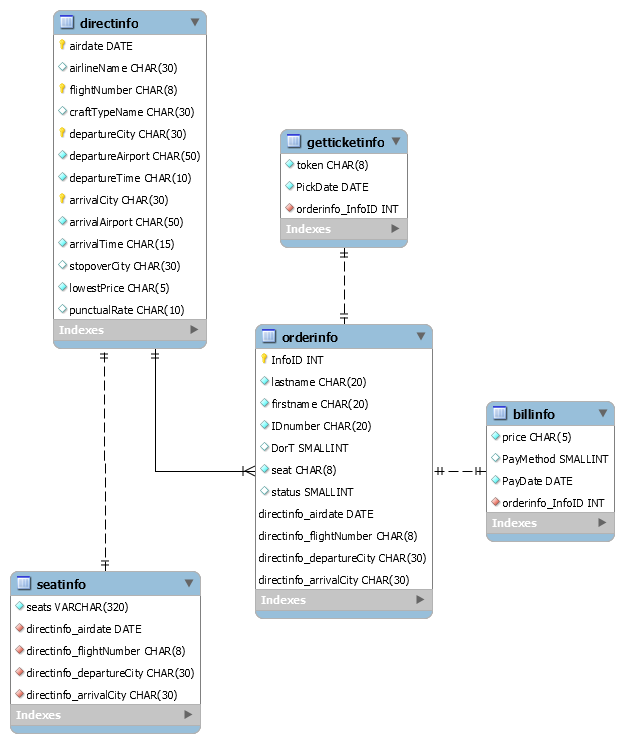


图 4.5 乌鸦脚式E-R图

### 4.4.2 数据库逻辑结构设计

1. 直达航班表DirectInfo

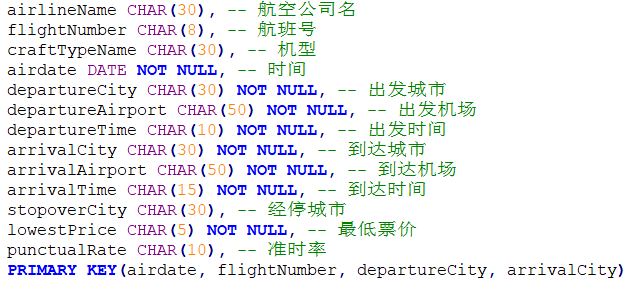


图 4.6 直达航班表

如图 4.6所示，主码为：（时间，航班号，出发城市，到达城市）。

非空的列为：时间，出发城市，出发机场，出发时间，到达城市，到达机场，到达时间，最低票价。

2. 中转信息表 TransitInfo

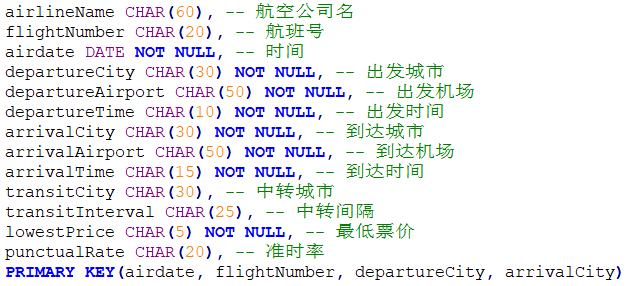


图 4.7 中转信息表

如图 4.7所示，主码为：（时间，航班号，出发城市，到达城市）。

非空的列为：时间，出发城市，出发机场，出发时间，到达城市，到达机场，到达时间，最低票价。

3. 航班座位信息表 SeatInfo

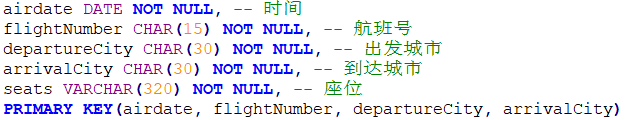


图 4.8 航班座位信息表

如图 4.8所示，主码为：（时间，航班号，出发城市，到达城市）。

非空的列为：时间，航班号，出发城市，到达城市，座位。

4. 旅订票信息表 OrderInfo

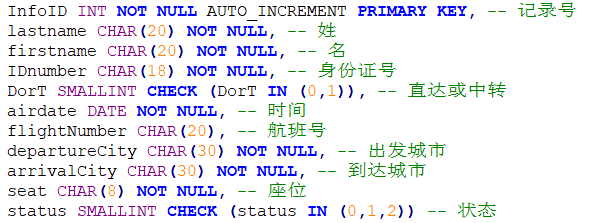


图 4.9 旅客订票信息表

如图 4.9所示，主码为记录号。

除航班号都为非空，且记录号自增，DorT 只能为 0 或 1,（0 直达，1 中转），status 只能为 0、1或2（0 未取票，1已取票，2已退票）。

5. 取票通知表 GetTicketInfo

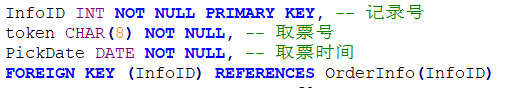


图 4.10 取票通知表

如图 4.10所示，主码为记录号，且来源自 OrderInfo 表中的记录号 InfoID。

全部列均为非空。

6. 账单 BillInfo

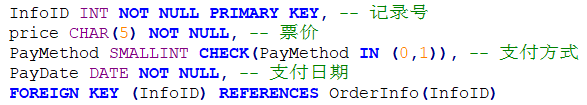


图 4.11 账单

如图 4.11所示，主码为记录号，且来源自 OrderInfo 表中的记录号 InfoID。

全部列均为非空，支付方式只能为 0 或 1（0微信，1支付宝）。

## 4.5 详细设计与实现

我的开发环境如下：

操作系统：Windows 10

前端开发框架：PyQt5

开发语言：Python + SQL

数据库：MySQL 8.0.20 for Win64 on x86\_64

下面给出各个模块的实现。

### 4.5.1 登录模块

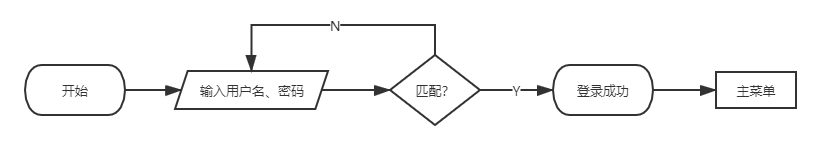


图 4.12 登录流程图

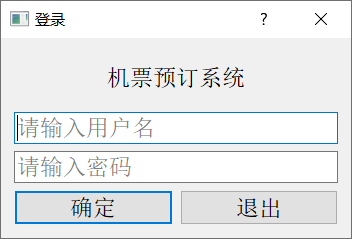


图 4.13 登录界面

用户点击确定后会在系统中查询用户名和密码，如果密码一致则验证通过，进入到主菜单中，如果密码不一致或者用户名根本不存在则会提示错误。用户点击退出则退出系统。

### 4.5.2 航班信息查询模块

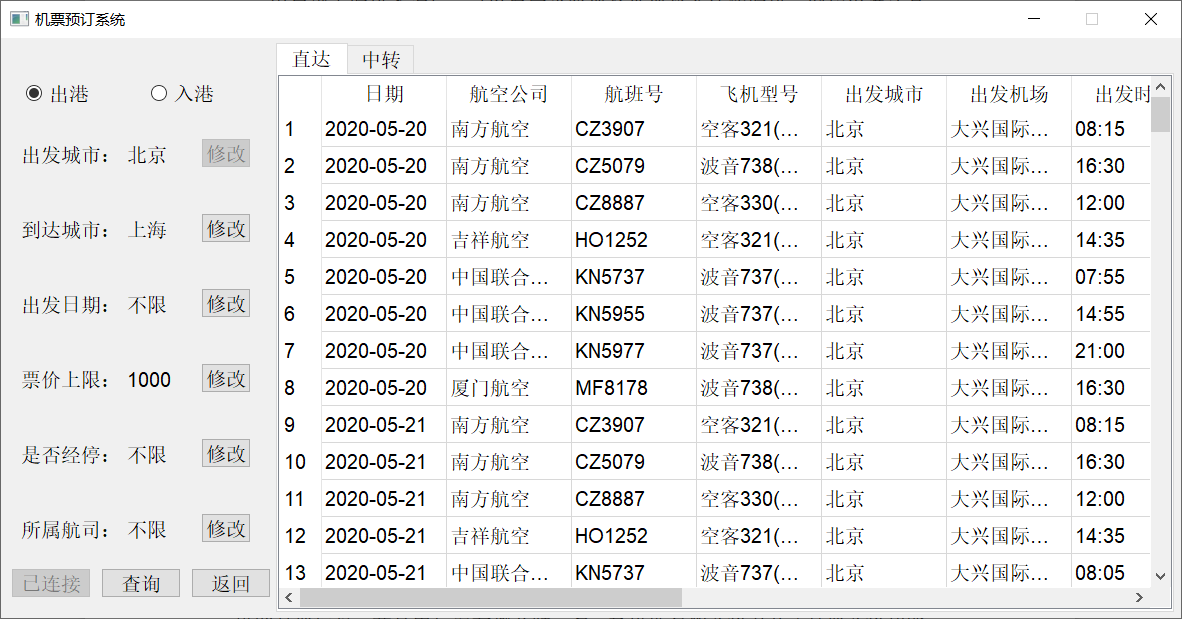


图 4.14 航班信息查询界面

用户进入航班查询后，可以自定义筛选条件选择合适的航班，例如选择不同的出发地或者目的地，出港还是入港，出发日期，票价上限，是否经停，所属航司等，系统会检测不在合理范围内的任何输入，并拒绝修改。查询到的结果通过选项卡组件呈现，用户在这个组件里直接右击对应的航班，再点击“订票”，即可进入订票界面。

### 4.5.3 订票模块

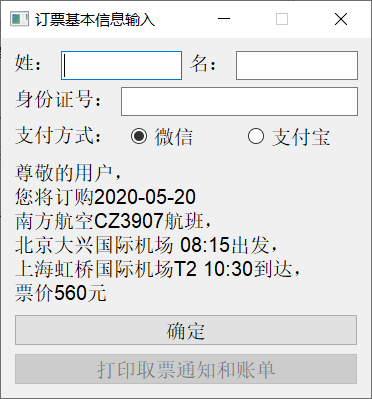


图 4.15 订票界面

用户在订票时需要输入姓、名、身份证号码，然后选择通过微信或者支付宝来进行支付，同时界面上会显示出用户选择的完整航班信息。点击确定完成订票。然后可以打印取票通知单和账单，获得账单编号和取票密码。

进行订票之后，旅客订票信息表将增加一条新的记录，订单状态为 0（未取票）。且取票通知表、账单也各自会增加一条新记录。航班座位信息表中座位对应位置也会改变。

### 4.5.4 打印模块

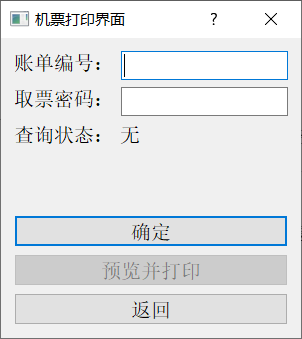


图 4.16 机票打印界面

输入对应的账单编号和取票密码，就能够查询自己所买机票的状态（已打印/未取票/已退订），然后可以选择进行打印。这里模拟了真实场景下购票的流程，只能打印一次，而且将网上购票和取纸质票的流程分离，也降低了耦合。

成功打印后，取票通知表 GetTicketInfo 中对应的记录将会删除，旅客订票信息表 OrderInfo 中对应的记录的 status 将会变成已取票。

### 4.5.5 订单查询模块

如图 4.17所示，用户可以直接通过自己的身份证号码来查询自己购买的所有机票，查看这些机票的详细信息，以及机票的状态（已打印/未取票/已退订）。这个功能使得用户查询历史机票更加方便，不必记住所有的订单编号和密码。

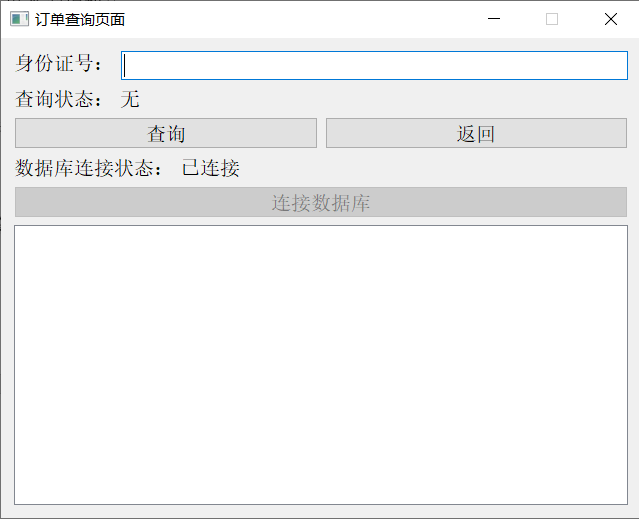


图 4.17 订单查询界面

### 4.5.6 退票模块

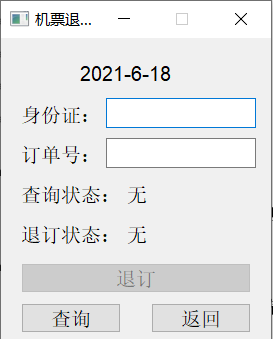


图 4.18 退票界面

用户需要输入身份证号码和订单号，匹配成功即可退票。只有未打印（未取票）的票才能够退票，打印之后就不能够再退票了。这里也模拟了真实场景，网上买票的话只有未取出的票才能在网上退，已经取了的票需要到机场线下退订。

取票通知表 GetTicketInfo 和账单 BillInfo 中对应的记录都会被删除，旅客订票信息表 OrderInfo 中对应的记录的 status 将会变成已退票。

### 4.5.7 预定情况查询模块

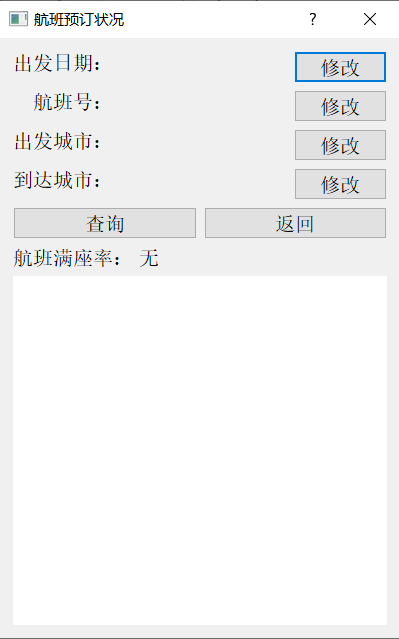


图 4.19 预定情况查询界面

系统内存储了每个航班的座位情况，用户可以订购未满航班的座位，订购后剩余空座位会减少1。用户可以查询指定航班的座位情况，指定航班的座位情况通过饼图的形式直观展示，用户可以据此选择是否预定。这里机票的编号也模拟了真实的飞机座位编号，采用字母+数字的组合。

## 4.6 系统测试

### 4.6.1 登录

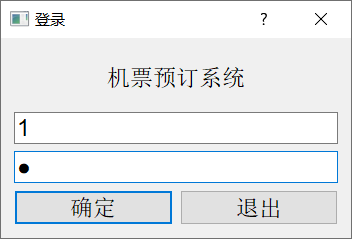


图 4.20 登录界面

输入用户名为 1，密码为 1，成功登录，进入主菜单。

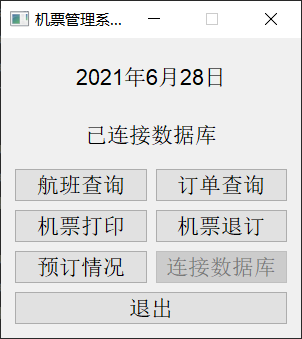


图 4.21 登录成功

### 4.6.2 航班查询

点击主菜单中的“航班查询”进入航班查询界面。

选择从北京到上海的航班，价格小于800，其他不限，得到的结果如图 4.22所示。



图 4.22 查询结果

再限定航空公司为南方航空，得到的结果如图 4.23所示。



图 4.23 查询结果

选择一个5月21日的航班进行订票，进入订票界面。

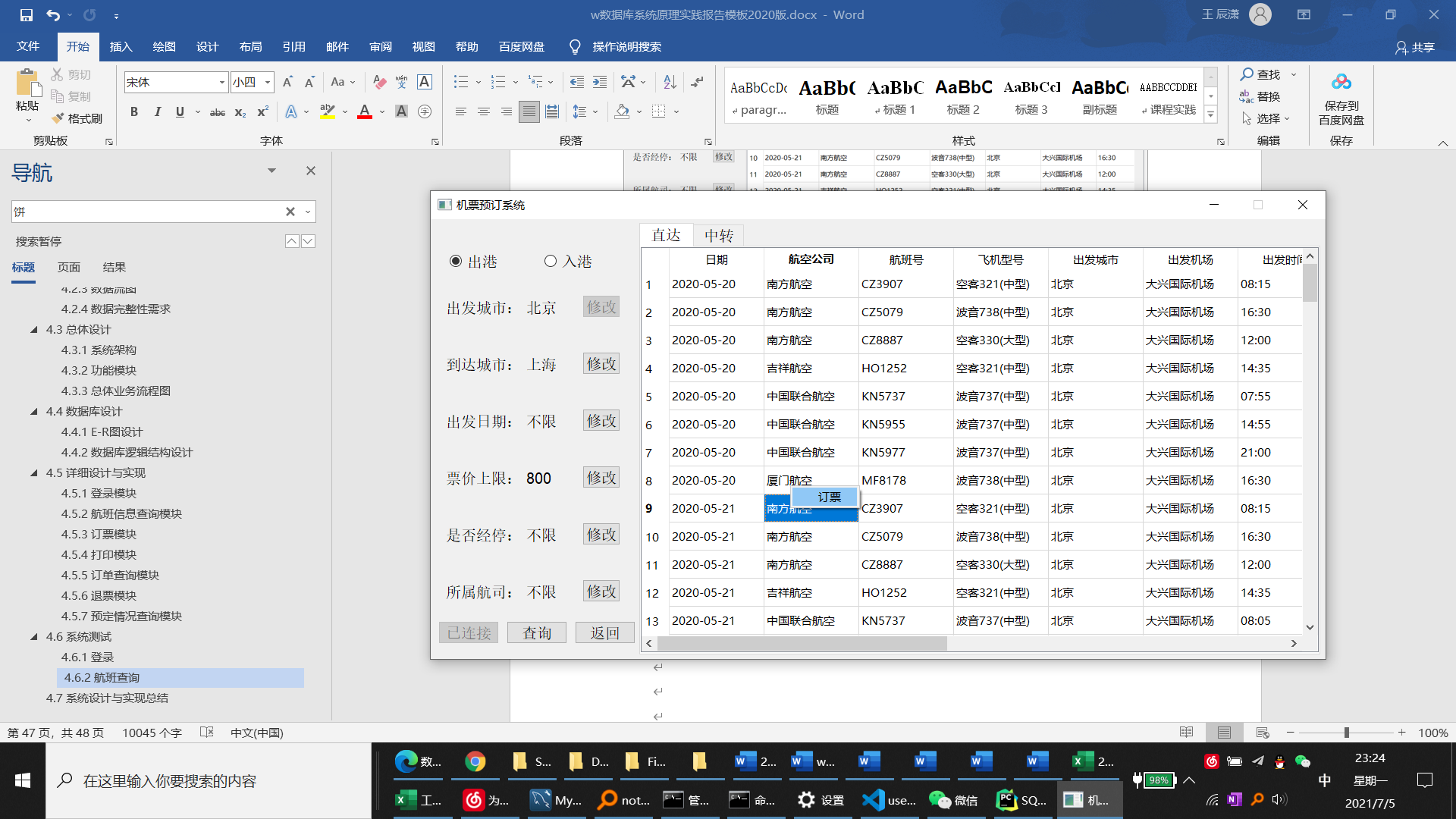


图 4.24 右击订票

### 4.6.3 订票

点击订票后会有确定的提示，如图 4.25所示。

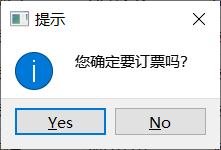


图 4.25 订票提示

进入订票界面后输入基本信息，选择支付方式，如图4. 26所示，可以看出，系统给出了完整的订票信息提示。



图 4.26 订票界面

点击确定，再次确认信息。

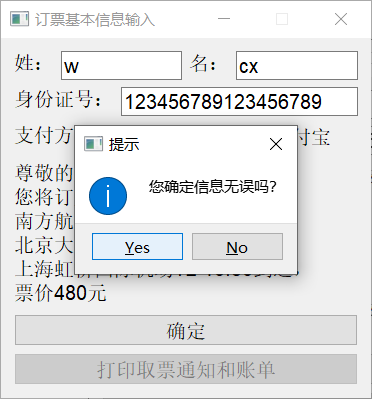


图 4.27 确认信息

确定后提示订票成功。

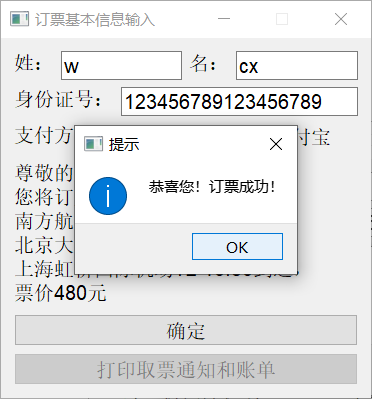


图 4.28 订票成功

然后点击打印取票通知和账单的按钮。



图 4.29 打印取票通知和账单

生成了取票通知单和账单，记住账单号15和取票密码hn3FIM。



图 4.30 取票通知单和账单

这时可以在MySQL Workbench中查看对应数据库的变化。取票表getticketinfo如图 4.31所示，可以看到，增加了一条记录，infoid对应账单号，token对应取票密码。

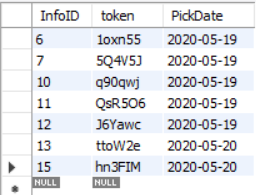


图 4.31 getticketinfo表

与此同时orderinfo表也增加了一条订单记录。

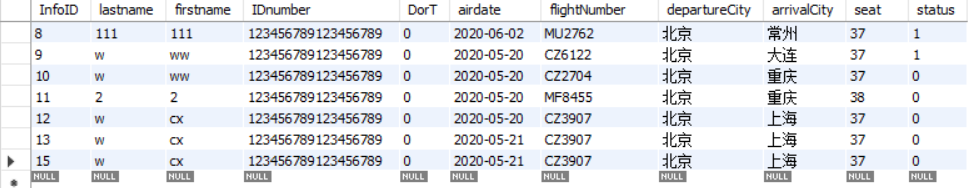


图 4.32 orderinfo表

### 4.6.4 打印

在机票打印界面输入刚才的订单编号和取票密码，然后点击确定进行查询。



图 4.33 查询

查询状态为成功，接下来点击“预览并打印”按钮进行打印。

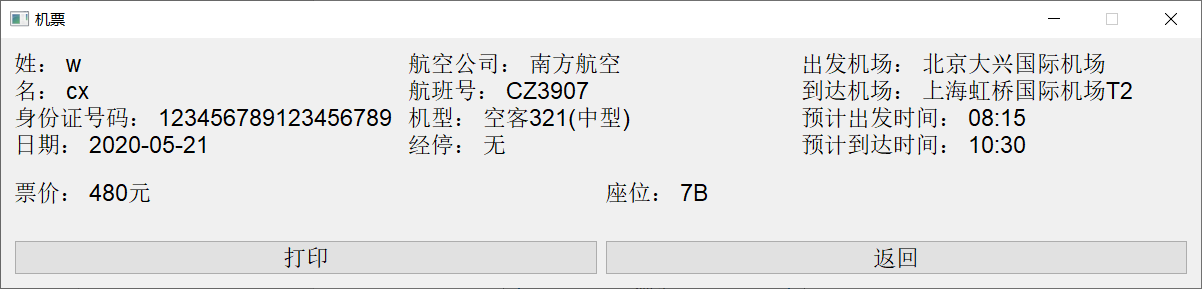


图 4.34 查询机票

机票如图 4.35所示，点击“打印”按钮进行打印。

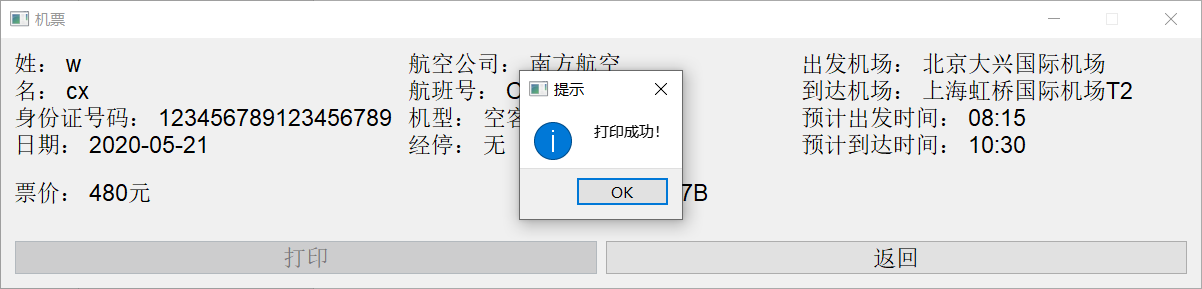


图 4.35 打印机票

提示打印成功，而且打印按钮变灰，不能二次打印。

这时orderinfo中对应的status变成了1。

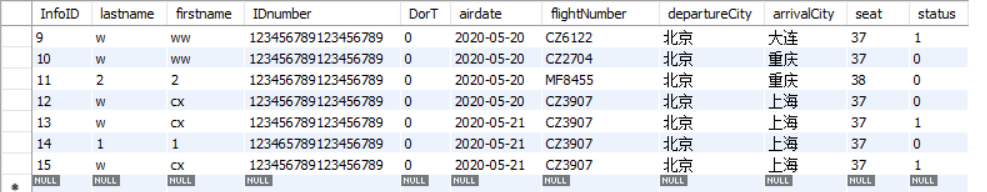


图 4.36 orderinfo表

### 4.5.5 订单查询

接下来打开主菜单中的订单查询界面，输入身份证号码123456789123465789，然后查询，发现了10条记录（因为之前用这个做过很多测试）。

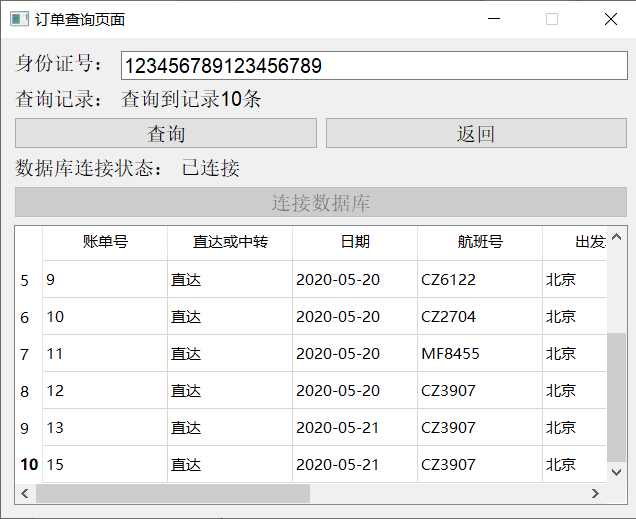


图 4.37 订单查询

可以看到最后一条记录，也就是刚才取票的订单，状态已经变成了已取票。

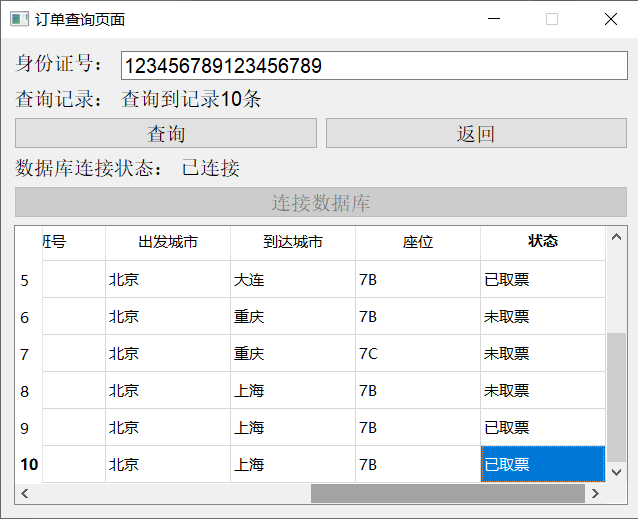


图 4.38 已取票状态

### 4.5.6 退票

刚才查询订单，发现了这个用户12号订单还未取票，尝试对这个订单进行退票。在主菜单点击“机票退订”按钮进入退票界面。

输入对应的身份证号码和订单号，进行查询。

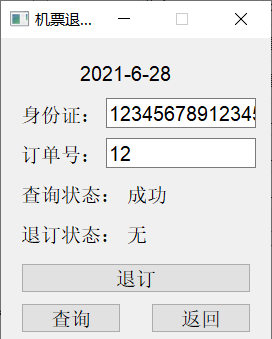


图 4.39退票界面

查询成功，点击“退订”按钮，进行退票。

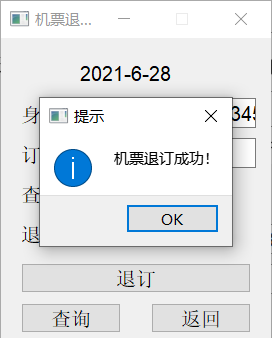


图 4.40 退订成功

可以看到，机票退订成功。界面的退订状态也变为成功。

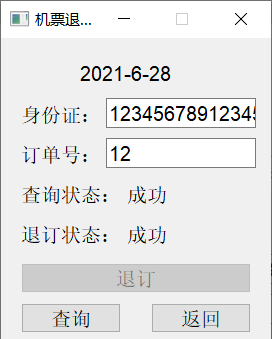


图 4. 41 界面的退订状态

再次进入订单查询界面，查看这个订单，发现状态变为已退票。

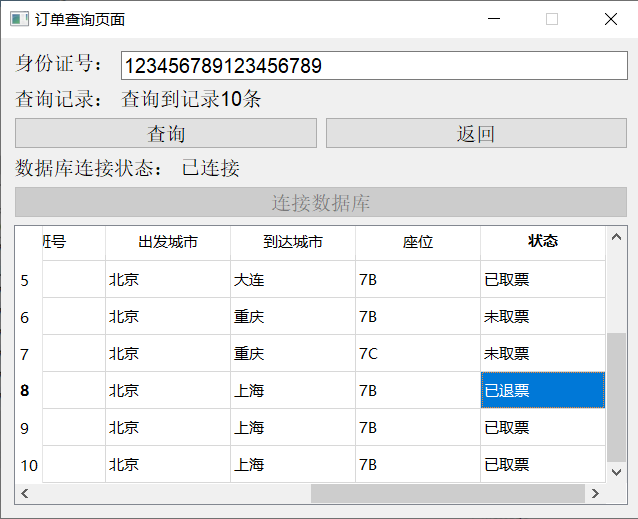


图 4.42 订单查询界面

说明成功退票。

### 4.5.7 预定情况查询

在主菜单点击“预定情况”按钮，进入预定情况查询界面。输入刚才预定的这个航班的信息，点击“查询”，界面上显示出了饼状图，可以直观的看到上座率和出售情况。

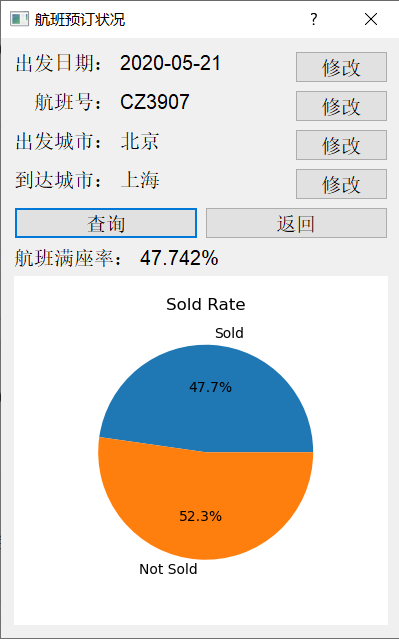


图 4.43 航班预定情况

### 4.5.8 异常处理

1. 身份证号码不是18位时，会提示出错。

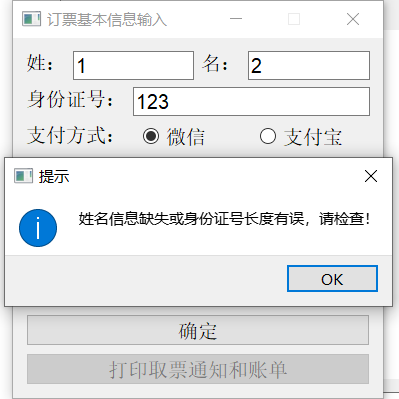


图 4.44 身份证号码错误

1. 已经取的机票不能退订

查询刚刚取过票的15号订单，然后尝试退订，发现退订失败。

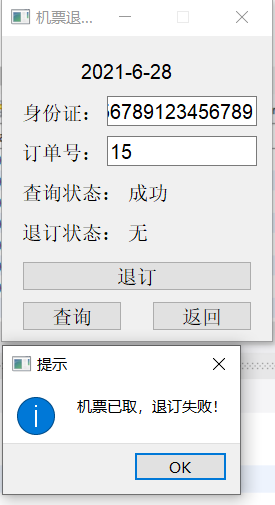


图 4.45 已经取的机票不能退订

## 4.7 系统设计与实现总结

在实验任务中，我设计了一个功能较为完备的机票预定系统，了解了一个数据库应用从设计到实现的大致流程，我主要完成了以下工作：

**1. 确定整体的架构**

整体使用B/S架构，我学会了客户端和服务器的通信过程。

**2. 完成数据库的设计**

根据题目要求，我细化了需求分析，为应用设计划分了各个不同的模块，并且设计ER图、数据流图等（使用MySQL WorkBench中的E-R设计模式和整体模型设计），并在MySQL中建相关的表和相关的完整性约束。

**3. 界面绘制**

使用PyQt5进行绘制，由于界面比较简单，直接使用了代码进行布局，但是如果写复杂界面还是需要使用Qt Designer。

**4. 逻辑实现**

我详细设计了各功能模块的界面和处理逻辑，实现了SQL和编程语言的结合使用，并增加了事务处理的操作。

**5. 测试、debug以及功能完善**

我导入数据之后，对每一个功能进行测试，发现了一些存在的bug，逐步完善功能，最终得到较为健壮的系统。

# 5 课程总结

## 5.1 工作总结

在本次实践课程中，我主要做了如下工作：

1) 掌握了 MySQL 及其 DBMS 的使用方法，知道了在 MySQL 中执行建表，导入数据，导出数据，等操作的方法。

2) 掌握了在 MySQL 上增加新用户并配置权限的方法，了解了数据库权限控制的方法，以及新建角色的方法。

3) 掌握了在 MySQL 中新建查询文件（.sql 文件）的方法，以及 MySQL执行增删查改语句的方法。

4) 掌握了 SQL 语言中，有关建表，建立视图，建立触发器，以及 UPDATE，INSERT INTO，DELETE 语句的使用方法。

5) 掌握了 SQL语言中，SELECT 语句的使用方法，掌握了单表查询，多表查询，嵌套查询等方法，掌握了各种子句使用方法，掌握了各类函数的使用方法，掌握了抽象数据查询需求，将其转换为 SQL查询语句的方法。

6) 了解了数据库性能分析的方法，学会了如何在 MySQL 查看查询的计划执行计划和实际执行计划的方法，学会了根据执行计划分查询性能的方法。

7) 掌握了数据库事务的使用方法，知道了 MySQL 处理执行出错事务的回滚机制以及其交互界面上的提示，了解了自定义函数的书写和执行方法。

8) 学会了根据实际生产生活中的事务以及处理事务的流程抽象出事务参与者对事务流程处理的需求的分析方法。并根据需求分析进一步抽象出一个概念模型的方法。

9) 掌握了数据库系统设计的需求分析方法，以及根据需求分析进行概念模型设计的方法。掌握了数据字典，数据流图，E-R 图等设计表示工具的使用方法，掌握了由 E-R 图进行关系数据库设计的方法，并根据实际情况进行必要的修改优化的方法。

10) 掌握了根据现实需求应用，设计相应功能数据库，表，视图，触发器的方法。熟练掌握了数据库编程的技术与方法。

11) 掌握了综合应用系统的设计，实现，调试，展示方法。熟悉了将系统目标与生产、生活中的实际需求较好的结合，能够用较为合理的数据结构及处理流程解决或优化现实生活中社会、健康、安全、法律或者文化中的某些问题的设计理念和思路。

## 5.2 心得体会

这是一个任务量相当大的实验，在完成的过程中，我收获巨大。系统从最初的版本到最终呈现的版本，中间经历了多次数据库扩展与修改，每一次扩展和修改都丰富或完善了整个数据库应用系统的功能，且在这个过程中，对于数据库设计的方法，我有了更深的理解，对于软件工程内中讲述的软件整个生命周期的管理，又有了新的理解。可以说，本次综合系统设计与实现工作虽然只是一次课程实践，但是它的意义不亚于一次完整的软件开发的工作。

课程综合实践全方面地培养了我的数据库编程能力，让我在课程只对SQL 一知半解的基础上实现了对数据库概念的深入了解，通过实践掌握的知识是深刻的，今后不论学习还是工作中，对于数据库系统的应用都不会如理论知识般束之高阁，而必将被我用作一种具有实践意义的技能。

## 5.3 展望

本次实验极大的激起了我对数据库的兴趣，我会进一步探索数据库中并发、事务等重要概念。暑假我将去蚂蚁集团的OceanBase部门实习，更加深刻的去理解数据库内核，去了解数据库的存储方式以及底层实现，并争取在这个完全国产自研的原生分布式关系数据库中贡献自己的代码。

谨此感谢老师在数据库系统综合实践的过程中对我的指正，以及对于实验设计方面上的指导，我一定再接再厉，将数据库相关技术作计算机行业学习者的根本技术加以深入学习应用。

# 附录

SQL练习部分的代码在对应章节已经给出，不再重复。下面列出综合实践任务中关键的部分代码。

class InputInfo(QWidget):

'''

订票时输入有关信息

'''

def \_\_init\_\_(self,allinfo,tag,seat):

super().\_\_init\_\_()

self.allinfo = allinfo # 购票信息

self.tag = tag # 直达或中转

self.date = '2021-06-28'

self.seat = seat

self.initUI()

def initUI(self):

'''

初始化界面

'''

self.setFont(QFont("Arial",12))

self.resize(370,360)

self.setFixedSize(self.width(),self.height())

self.setWindowTitle('订票基本信息输入')

self.hbox1 = QHBoxLayout()

# 姓

self.lb1 = QLabel(self)

#self.lb1.setGeometry(20,20,30,30)

self.lb1.setText('姓：')

self.lb1.setAlignment(Qt.AlignLeft)

self.hbox1.addWidget(self.lb1)

self.le1 = QLineEdit(self)

#self.le1.setGeometry(60,15,100,30)

self.hbox1.addWidget(self.le1)

# 名

self.lb2 = QLabel(self)

#self.lb2.setGeometry(200,20,30,30)

self.lb2.setText('名：')

self.lb2.setAlignment(Qt.AlignLeft)

self.hbox1.addWidget(self.lb2)

self.le2 = QLineEdit(self)

#self.le2.setGeometry(240,15,110,30)

self.hbox1.addWidget(self.le2)

self.hbox2 = QHBoxLayout()

# 身份证号码

self.lb3 = QLabel(self)

#self.lb3.setGeometry(20,65,80,30)

self.lb3.setText('身份证号：')

self.lb3.setAlignment(Qt.AlignLeft)

self.le3 = QLineEdit(self)

#self.le3.setGeometry(110,60,240,30)

self.hbox2.addWidget(self.lb3)

self.hbox2.addWidget(self.le3)

self.hbox3 = QHBoxLayout()

# 支付方式

self.lb4 = QLabel(self)

#self.lb4.setGeometry(20,105,80,30)

self.lb4.setText('支付方式：')

self.lb4.setAlignment(Qt.AlignLeft)

self.rb1 = QRadioButton('微信',self)

#self.rb1.setGeometry(130,100,80,30)

self.rb1.setChecked(True)

self.rb2 = QRadioButton('支付宝',self)

#self.rb2.setGeometry(230,100,80,30)

self.hbox3.addWidget(self.lb4)

self.hbox3.addWidget(self.rb1)

self.hbox3.addWidget(self.rb2)

# 提示信息

self.lb5 = QLabel(self)

#self.lb5.setGeometry(20,145,350,120)

# 直达

if self.tag == 0:

showtext = '尊敬的用户，\n您将订购' + \

self.allinfo[0] + '\n' + self.allinfo[1] + self.allinfo[2] +\

'航班，\n' + self.allinfo[4] + self.allinfo[5] + ' ' + self.allinfo[6] +\

'出发，\n' + self.allinfo[7] + self.allinfo[8] + ' ' + self.allinfo[9] +\

'到达，\n'

if self.allinfo[10]:

showtext += '经停' + self.allinfo[10] +'，'

showtext += '票价' + self.allinfo[11] + '元'

self.lb5.setText(showtext)

# 中转

elif self.tag == 1:

showtext = '尊敬的用户，\n您将订购' + self.allinfo[0] + '\n'

if self.allinfo[1] and self.allinfo[2]:

s1 = self.allinfo[1].split()

s2 = self.allinfo[2].split()

showtext += s1[0] + s2[0] + '和' + s1[1] + s2[1] + '航班，\n'

else:

showtext += '某航班，\n'

showtext += self.allinfo[3] + self.allinfo[4] + ' ' + self.allinfo[5] +\

'出发，\n' + self.allinfo[6] + self.allinfo[7] + ' ' + self.allinfo[8] +\

'到达，\n'

if self.allinfo[9] and self.allinfo[10]:

showtext += '中转' + self.allinfo[9] + self.allinfo[10] +'，\n'

else:

showtext += '某地中转，\n'

showtext += '票价' + self.allinfo[11] + '元'

self.lb5.setText(showtext)

# 确定

self.pb1 = QPushButton('确定',self)

self.pb1.setGeometry(20,275,330,30)

self.pb1.clicked.connect(self.infosure)

# 打印取票通知

self.pb2 = QPushButton('打印取票通知和账单',self)

self.pb2.setGeometry(20,315,330,30)

self.pb2.setEnabled(False)

self.pb2.clicked.connect(self.print\_get\_ticket\_and\_bill)

self.vbox = QVBoxLayout()

self.vbox.addLayout(self.hbox1)

self.vbox.addLayout(self.hbox2)

self.vbox.addLayout(self.hbox3)

self.vbox.addWidget(self.lb5)

self.vbox.addWidget(self.pb1)

self.vbox.addWidget(self.pb2)

self.setLayout(self.vbox)

self.show()

def check(self):

'''

检查输入信息的完整性

'''

if not (self.le1.text() and self.le2.text()): # 姓名不可缺失

return False

else:

m = re.compile(r'^\d{18}$').match(self.le3.text()) # 且身份证号长度必须为18位

if m:

return True

else:

return False

def infosure(self):

'''

输入信息确认

'''

reply = QMessageBox.information(self, '提示','您确定信息无误吗？',QMessageBox.No | QMessageBox.Yes,QMessageBox.Yes)

if reply == QMessageBox.Yes:

if self.check():

QMessageBox.information(self,'提示','恭喜您！订票成功！',QMessageBox.Ok)

self.pb1.setEnabled(False)

self.pb2.setEnabled(True)

token = self.generatetoken() # 生成取票密码

InfoID = self.insert\_order() # 插入新的旅客订票信息

self.all\_get\_ticket\_info = self.insert\_get\_ticket(InfoID, token) # 插入新的取票通知信息

self.all\_bill\_info = self.insert\_bill(InfoID) # 插入新的账单信息

self.take\_seat() # 将座位标记为0

else:

QMessageBox.information(self,'提示','姓名信息缺失或身份证号长度有误，请检查！',QMessageBox.Ok)

def take\_seat(self):

'''

占座

'''

self.airinfo = pymysql.connect(host='127.0.0.1',port=3306,user='root',password='666666',database='airinfo',charset='utf8')

cursor = self.airinfo.cursor()

# 直达

if self.tag == 0:

sql = 'SELECT seats FROM SeatInfo WHERE airdate = "%s" AND flightNumber = "%s" AND departureCity = "%s" \

AND arrivalCity = "%s";' % (self.allinfo[0], self.allinfo[2], self.allinfo[4], self.allinfo[7])

cursor.execute(sql)

s = cursor.fetchall()[0][0]

l = list(s)

l[self.seat[0]] = '0'

s = ''.join(l)

sql = 'UPDATE SeatInfo SET seats = "%s" WHERE airdate = "%s" AND flightNumber = "%s" AND departureCity = "%s" \

AND arrivalCity = "%s";' % (s, self.allinfo[0], self.allinfo[2], self.allinfo[4], self.allinfo[7])

# 中转

elif self.tag == 1:

sql = 'SELECT seats FROM SeatInfo WHERE airdate = "%s" AND flightNumber = "%s" AND departureCity = "%s" \

AND arrivalCity = "%s";' % (self.allinfo[0], self.allinfo[2], self.allinfo[3], self.allinfo[6])

cursor.execute(sql)

s = cursor.fetchall()[0][0]

l = list(s)

l[self.seat[0]] = '0'

l[self.seat[1]] = '0'

s = ''.join(l)

sql = 'UPDATE SeatInfo SET seats = "%s" WHERE airdate = "%s" AND flightNumber = "%s" AND departureCity = "%s" \

AND arrivalCity = "%s";' % (s, self.allinfo[0], self.allinfo[2], self.allinfo[3], self.allinfo[6])

try:

cursor.execute(sql)

self.airinfo.commit()

except:

self.airinfo.rollback()

cursor.close()

self.airinfo.close()

def insert\_order(self):

'''

插入旅客订票信息

'''

self.airinfo = pymysql.connect(host='127.0.0.1',port=3306,user='root',password='666666',database='airinfo',charset='utf8')

cursor = self.airinfo.cursor()

# 直达

if self.tag == 0:

sql = 'INSERT INTO OrderInfo VALUES (DEFAULT,"%s","%s",%d,%d,"%s","%s","%s","%s","%s",%d);' % \

(self.le1.text(), self.le2.text(), int(self.le3.text()), self.tag, self.allinfo[0], self.allinfo[2],\

self.allinfo[4], self.allinfo[7], str(self.seat[0]), 0) # 0 表示未取票

elif self.tag == 1:

sql = 'INSERT INTO OrderInfo VALUES (DEFAULT,"%s","%s",%d,%d,"%s","%s","%s","%s","%s",%d);' % \

(self.le1.text(), self.le2.text(), int(self.le3.text()), self.tag, self.allinfo[0], self.allinfo[2],\

self.allinfo[3], self.allinfo[6], str(self.seat[0])+' '+str(self.seat[1]), 0) # 0 表示未取票

try:

cursor.execute(sql)

self.airinfo.commit()

except:

self.airinfo.rollback()

sql = 'SELECT MAX(InfoID) FROM OrderInfo;'

cursor.execute(sql)

data = cursor.fetchall()

cursor.close()

self.airinfo.close()

return data[0][0] #返回当前InfoID

def insert\_get\_ticket(self,infoid,token):

'''

插入取票通知

'''

self.airinfo = pymysql.connect(host='127.0.0.1',port=3306,user='root',password='666666',database='airinfo',charset='utf8')

cursor = self.airinfo.cursor()

# 计算取票时间(-1)

d = datetime.datetime.strptime(self.allinfo[0],'%Y-%m-%d')

d = d - datetime.timedelta(days=1)

d = d.strftime("%Y-%m-%d")

sql = 'INSERT INTO GetTicketInfo VALUES (%d,"%s","%s");' % \

(infoid, token, d)

try:

cursor.execute(sql)

self.airinfo.commit()

except:

self.airinfo.rollback()

cursor.close()

self.airinfo.close()

data = [infoid,token,d]

return data

def insert\_bill(self,id):

'''

插入账单

'''

self.airinfo = pymysql.connect(host='127.0.0.1',port=3306,user='root',password='666666',database='airinfo',charset='utf8')

cursor = self.airinfo.cursor()

if self.rb1.isChecked():

method = 0

else:

method = 1

sql = 'INSERT INTO BillInfo VALUES (%d,%d,%d,"%s");' % \

(id, int(self.allinfo[11]), method, self.date)

try:

cursor.execute(sql)

self.airinfo.commit()

except:

self.airinfo.rollback()

cursor.close()

self.airinfo.close()

data = [id,int(self.allinfo[11]),method,self.date]

return data

def generatetoken(self):

'''

生成六位随机取票密码

'''

while 1:

token = ''

for i in range(6):

num = random.randint(0, 9)

letter = chr(random.randint(97, 122)) #取小写字母

Letter = chr(random.randint(65, 90)) #取大写字母

s = str(random.choice([num,letter,Letter]))

token += s

# 检查是否有重复密码

self.airinfo = pymysql.connect(host='127.0.0.1',port=3306,user='root',password='666666',database='airinfo',charset='utf8')

cursor = self.airinfo.cursor()

sql = 'SELECT token FROM GetTicketInfo WHERE token = "%s";' % token

cursor.execute(sql)

data = cursor.fetchall()

if not data:

return token

else:

pass

def print\_get\_ticket\_and\_bill(self):

'''

打印取票通知和账单

'''

self.hide()

self.pgtab = GetTicket\_Bill(self.all\_get\_ticket\_info, self.all\_bill\_info)

self.pgtab.show()

class GetTicket\_Bill(QMainWindow):

'''

取票通知单

'''

def \_\_init\_\_(self,info1,info2):

super().\_\_init\_\_()

self.info1 = info1

self.info2 = info2

self.initUI()

def initUI(self):

'''

初始化界面

'''

self.setFont(QFont("Arial",14))

self.resize(620,400)

self.setFixedSize(self.width(),self.height())

self.setWindowTitle('取票通知单和账单')

text1 = '机场取票通知单' + '\n\n' +\

'账单号: ' + str(self.info1[0]) + '\n' +\

'取票密码: ' + self.info1[1] + '\n' +\

'取票时间: ' + self.info1[2] + '\n'

# 左侧为取票通知单

self.lb1 = QLabel(self)

self.lb1.setGeometry(0,0,300,300)

self.lb1.setText(text1)

self.lb1.setAlignment(Qt.AlignCenter)

if self.info2[2] == 0:

method = '微信'

else:

method = '支付宝'

text2 = '账单' + '\n\n' +\

'账单号: ' + str(self.info2[0]) + '\n' +\

'金额: ' + str(self.info2[1]) + '\n' +\

'支付方式: ' + method + '\n' +\

'支付日期: ' + self.info2[3] + '\n'

# 右侧为取票通知单

self.lb2 = QLabel(self)

self.lb2.setGeometry(320,0,300,300)

self.lb2.setText(text2)

self.lb2.setAlignment(Qt.AlignCenter)

self.pb1 = QPushButton('返回',self)

self.pb1.setGeometry(90,320,120,50)

self.pb1.clicked.connect(self.back)

self.pb2 = QPushButton('打印',self)

self.pb2.setGeometry(390,320,120,50)

self.pb2.clicked.connect(self.Print)

def Print(self):

'''

打印

'''

screen = QApplication.primaryScreen()

pix1 = screen.grabWindow(self.lb1.winId())

pix1.save('取票通知单'+str(self.info1[0])+'.jpg')

screen = QApplication.primaryScreen()

pix2 = screen.grabWindow(self.lb2.winId())

pix2.save('账单'+str(self.info2[0])+'.jpg')

QMessageBox.information(self,'提示','打印成功！',QMessageBox.Ok)

def back(self):

'''

返回

'''

self.hide()

self.airsearch = AirSearch()

self.airsearch.show()

class AirSearch(QMainWindow):

'''

航班查询界面

'''

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

self.connectdb() #自动连接

def initUI(self):

'''

初始化界面

'''

#更改尺寸

self.setFont(QFont("Arial",12))

self.resize(1180,580)

self.setFixedSize(self.width(),self.height())

self.setWindowTitle('机票预订系统')

# 出港或入港

self.rb1 = QRadioButton('出港',self)

self.rb1.setGeometry(25,30,100,50)

self.rb1.setChecked(True)

self.rb1.pressed.connect(self.valid\_port)

self.rb2 = QRadioButton('入港',self)

self.rb2.setGeometry(150,30,100,50)

self.rb2.pressed.connect(self.valid\_port)

# 修改查询要求

self.lb1 = QLabel(self)

self.lb1.setGeometry(20,105,175,50)

self.lb1.setText('出发城市： 北京')

self.lb1.setAlignment(Qt.AlignLeft)

self.pb1 = QPushButton('修改',self)

self.pb1.setGeometry(200,100,50,30)

self.pb1.setEnabled(False)

self.pb1.clicked.connect(self.change\_query)

self.lb2 = QLabel(self)

self.lb2.setGeometry(20,180,175,50)

self.lb2.setText('到达城市： 阿尔山')

self.lb2.setAlignment(Qt.AlignLeft)

self.pb2 = QPushButton('修改',self)

self.pb2.setGeometry(200,175,50,30)

self.pb2.clicked.connect(self.change\_query)

self.lb3 = QLabel(self)

self.lb3.setGeometry(20,255,175,50)

self.lb3.setText('出发日期： 不限')

self.lb3.setAlignment(Qt.AlignLeft)

self.pb3 = QPushButton('修改',self)

self.pb3.setGeometry(200,250,50,30)

self.pb3.clicked.connect(self.change\_query)

self.lb4 = QLabel(self)

self.lb4.setGeometry(20,330,175,50)

self.lb4.setText('票价上限： 不限')

self.lb4.setAlignment(Qt.AlignLeft)

self.pb4 = QPushButton('修改',self)

self.pb4.setGeometry(200,325,50,30)

self.pb4.clicked.connect(self.change\_query)

self.lb5 = QLabel(self)

self.lb5.setGeometry(20,405,175,50)

self.lb5.setText('是否经停： 不限')

self.lb5.setAlignment(Qt.AlignLeft)

self.pb5 = QPushButton('修改',self)

self.pb5.setGeometry(200,400,50,30)

self.pb5.clicked.connect(self.change\_query)

self.lb6 = QLabel(self)

self.lb6.setGeometry(20,480,175,50)

self.lb6.setText('所属航司： 不限')

self.lb6.setAlignment(Qt.AlignLeft)

self.pb6 = QPushButton('修改',self)

self.pb6.setGeometry(200,475,50,30)

self.pb6.clicked.connect(self.change\_query)

self.pb7 = QPushButton('连接',self)

self.pb7.setGeometry(10,530,80,30)

self.pb7.clicked.connect(self.connectdb)

self.pb8 = QPushButton('查询',self)

self.pb8.setGeometry(100,530,80,30)

self.pb8.clicked.connect(self.query)

self.pb9 = QPushButton('返回',self)

self.pb9.setGeometry(190,530,80,30)

self.pb9.clicked.connect(self.back)

# 选项卡网格

self.tb = QTabWidget(self)

self.tb.setGeometry(275,5,900,570)

# 直达表格支持右键菜单

self.tb1 = QTableWidget()

self.tb1.setContextMenuPolicy(Qt.CustomContextMenu)

self.tb1.customContextMenuRequested.connect(self.GenerateMenuDirect)

# 中转表格支持右键菜单

self.tb2 = QTableWidget()

self.tb2.setContextMenuPolicy(Qt.CustomContextMenu)

self.tb2.customContextMenuRequested.connect(self.GenerateMenuTransit)

self.tb.addTab(self.tb1,"直达")

self.tb.addTab(self.tb2,"中转")

self.show()

def valid\_port(self):

'''

出港航班和入港航班选定后要限制按钮修改

'''

s = self.sender()

if s == self.rb1:

self.lb1.setText('出发城市： 北京')

self.pb1.setEnabled(False)

self.lb2.setText('到达城市： 阿尔山')

self.pb2.setEnabled(True)

elif s == self.rb2:

self.lb1.setText('出发城市： 阿尔山')

self.pb1.setEnabled(True)

self.lb2.setText('到达城市： 北京')

self.pb2.setEnabled(False)

def change\_query(self):

'''

修改查询信息

'''

s = self.sender()

if s == self.pb1:

text, ok = QInputDialog.getItem(self, '修改', '请输入出发城市：',citys)

if text not in citys:

QMessageBox.warning(self,'警告','输入不符合要求！',QMessageBox.Yes,QMessageBox.Yes)

elif ok:

self.lb1.setText('出发城市： ' + text)

elif s == self.pb2:

text, ok = QInputDialog.getItem(self, '修改', '请输入到达城市：',citys)

if text not in citys:

QMessageBox.warning(self,'警告','输入不符合要求！',QMessageBox.Yes,QMessageBox.Yes)

elif ok:

self.lb2.setText('到达城市： ' + text)

elif s == self.pb3:

text, ok = QInputDialog.getItem(self, '修改', '请输入出发时间：',time\_limit)

if text not in time\_limit:

QMessageBox.warning(self,'警告','输入不符合要求！',QMessageBox.Yes,QMessageBox.Yes)

elif ok:

self.lb3.setText('出发日期： ' + text)

elif s == self.pb4:

text, ok = QInputDialog.getItem(self, '修改', '请输入票价上限：',price\_limit)

if text not in price\_limit:

QMessageBox.warning(self,'警告','输入不符合要求！',QMessageBox.Yes,QMessageBox.Yes)

elif ok:

self.lb4.setText('票价上限： ' + text)

elif s == self.pb5:

op = ['不限','是','否']

text, ok = QInputDialog.getItem(self, '修改', '请输入是否中转：',op)

if text not in op:

QMessageBox.warning(self,'警告','输入不符合要求！',QMessageBox.Yes,QMessageBox.Yes)

elif ok:

self.lb5.setText('是否中转： ' + text)

elif s == self.pb6:

text, ok = QInputDialog.getItem(self, '修改', '请输入所属航司：',airline\_limit)

if text not in airline\_limit:

QMessageBox.warning(self,'警告','输入不符合要求！',QMessageBox.Yes,QMessageBox.Yes)

elif ok:

self.lb6.setText('所属航司： ' + text)

def connectdb(self):

'''

连接数据库

'''

try:

self.airinfo = pymysql.connect(host='127.0.0.1',port=3306,user='root',password='666666',database='airinfo',charset='utf8')

self.pb7.setText('已连接')

self.pb7.setEnabled(False)

except:

pass

def query(self):

'''

查询

'''

if not self.pb7.isEnabled():

# 先查直达表

cursor = self.airinfo.cursor()

sql = 'SELECT \* FROM DirectInfo WHERE departureCity = "' + self.lb1.text()[6:]\

+ '" AND arrivalCity = "' + self.lb2.text()[6:] + '"'

# 对非不限的项添加到sql语句中

if self.lb3.text()[6:] != '不限':

sql += ' AND airdate = "' + self.lb3.text()[6:] + '"'

if self.lb4.text()[6:] != '不限':

sql += ' AND lowestPrice <= ' + self.lb4.text()[6:]

if self.lb5.text()[6:] == '是': #是否经停仅对直达表有效

sql += ' AND stopoverCity IS NOT NULL'

elif self.lb5.text()[6:] == '否':

sql += ' AND stopoverCity IS NULL'

if self.lb6.text()[6:] != '不限':

sql += ' AND airlineName = "' + self.lb6.text()[6:] + '"'

sql += ';'

cursor.execute(sql)

data = cursor.fetchall()

self.tb1.setRowCount(len(data))

self.tb1.setColumnCount(13)

self.tb1.setHorizontalHeaderLabels(['日期','航空公司','航班号','飞机型号','出发城市','出发机场','出发时间'\

,'到达城市','到达机场','到达时间','经停','票价','预估准时率'])

# self.tw.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch) # 横向自适应伸缩模式

self.tb1.setEditTriggers(QAbstractItemView.NoEditTriggers) # 不允许用户编辑

for i in range(len(data)):

for j in range(13):

newItem = QTableWidgetItem(str(data[i][j]))

self.tb1.setItem(i,j,newItem)

cursor.close()

# 再查中转表

cursor = self.airinfo.cursor()

sql = 'SELECT \* FROM TransitInfo WHERE departureCity = "' + self.lb1.text()[6:]\

+ '" AND arrivalCity = "' + self.lb2.text()[6:] + '"'

# 对非不限的项添加到sql语句中

if self.lb3.text()[6:] != '不限':

sql += ' AND airdate = "' + self.lb3.text()[6:] + '"'

if self.lb4.text()[6:] != '不限':

sql += ' AND lowestPrice <= ' + self.lb4.text()[6:]

if self.lb6.text()[6:] != '不限':

sql += ' AND (airlineName LIKE "' + self.lb6.text()[6:] + '%" OR airlineName LIKE "%' +\

self.lb6.text()[6:] + '")'

sql += ';'

cursor.execute(sql)

data = cursor.fetchall()

self.tb2.setRowCount(len(data))

self.tb2.setColumnCount(13)

self.tb2.setHorizontalHeaderLabels(['日期','航空公司','航班号','出发城市','出发机场','出发时间'\

,'到达城市','到达机场','到达时间','中转城市','中转时间','票价','预估准时率'])

# self.tw.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch) # 横向自适应伸缩模式

self.tb2.setEditTriggers(QAbstractItemView.NoEditTriggers) # 不允许用户编辑

for i in range(len(data)):

for j in range(13):

newItem = QTableWidgetItem(str(data[i][j]))

self.tb2.setItem(i,j,newItem)

cursor.close()

else:

QMessageBox.warning(self,'警告','请先连接数据库！',QMessageBox.Yes,QMessageBox.Yes)

def GetSeat(self,info,tag):

'''

获取座位

'''

cursor = self.airinfo.cursor()

# 直达

if tag == 0:

sql = 'SELECT seats FROM SeatInfo WHERE airdate = "%s" AND flightNumber = "%s" AND \

departureCity = "%s" AND arrivalCity = "%s";' % (info[0],info[2],info[4],info[7])

# 中转

elif tag == 1:

sql = 'SELECT seats FROM SeatInfo WHERE airdate = "%s" AND flightNumber = "%s" AND \

departureCity = "%s" AND arrivalCity = "%s";' % (info[0],info[2],info[3],info[6])

cursor.execute(sql)

data = cursor.fetchall()

if data:

data = data[0][0]

else:

return [-1,-1]

# 直达

if tag == 0:

data = re.compile(r'\d+').search(data).group(0)

index = data.find('1',int(len(data)/4))

return [index, index]

# 中转

elif tag == 1:

data = re.compile(r'\d+').search(data).group(0)

index1 = data.find('1',int(len(data)/4))

index2 = data.find('1',int(len(data)/4\*3))#瞎选的

return [index1, index2]

def GenerateMenuDirect(self,pos):

'''

直达页面菜单

'''

row = self.tb1.rowCount()

row\_num = -1

# 获取当前行数，从0开始

for i in self.tb1.selectionModel().selection().indexes():

row\_num = i.row()

if row\_num < row:

menu = QMenu()

order = menu.addAction(u'订票')

action = menu.exec\_(self.tb1.mapToGlobal(pos))

if action == order:

allinfo = []

for i in range(13):

info = self.tb1.item(row\_num,i).text()

allinfo.append(info)

reply = QMessageBox.information(self, '提示','您确定要订票吗？',QMessageBox.No | QMessageBox.Yes,QMessageBox.Yes)

if reply == QMessageBox.Yes:

seat = self.GetSeat(allinfo,0) # 获取座位

if seat[0] > -1:

self.hide()

self.input = InputInfo(allinfo,0,seat)

self.input.show()

else:

QMessageBox.information(self, '提示','航班已售罄！',QMessageBox.Ok)

def GenerateMenuTransit(self,pos):

'''

中转页面菜单

'''

row = self.tb2.rowCount()

row\_num = -1

# 获取当前行数，从0开始

for i in self.tb2.selectionModel().selection().indexes():

row\_num = i.row()

if row\_num < row:

menu = QMenu()

order = menu.addAction(u'订票')

action = menu.exec\_(self.tb2.mapToGlobal(pos))

if action == order:

allinfo = []

for i in range(13):

info = self.tb2.item(row\_num,i).text()

allinfo.append(info)

reply = QMessageBox.information(self, '提示','您确定要订票吗？',QMessageBox.No | QMessageBox.Yes,QMessageBox.Yes)

if reply == QMessageBox.Yes:

seat = self.GetSeat(allinfo,1) # 获取座位

if seat[0] > -1:

self.hide()

self.input = InputInfo(allinfo,1,seat)

self.input.show()

else:

QMessageBox.information(self, '提示','航班已售罄！',QMessageBox.Ok)

def back(self):

'''

返回到选择页面

'''

self.hide()

self.us = user\_select.select\_window()

self.us.show()

def logout(self):

'''

注销

'''

self.hide()

self.login = login.login()

self.login.show()