



华中科技大学

数据库系统原理实践报告

综合设计题目： 机票预订系统

姓 名： 徐 永 鑫

学 院： 计算机科学与技术学院

专 业： 计算机科学与技术

班 级： 计算机卓越实验 1601 班

学 号： U201610002

指导教师： 谢 美 意

分数	
教师签名	

2019 年 6 月 12 日

任务书

1 软件功能学习

完成下列练习，并在实践报告中叙述过程，可适当辅以截图（篇幅控制在 3 页内）：

- 1) 练习 DBMS (SQL Server / DM / 其他功能相当的数据库产品) 的安装和使用。
- 2) 练习创建新用户并配置权限，要求“2 SQL 练习”部分的操作以新建的普通用户而非管理员用户的身份进行。
- 3) 练习数据库的备份：数据库文件的脱机备份 / DBMS 提供的备份功能。

2 SQL 练习

2.1 数据定义

- 1) 定义下列跟电影相关的基表，包括完整性约束的说明。

电影表【电影编号，电影名称，电影类型，导演姓名，电影时长（以分钟计），是否 3D，用户评分】

FILM(FID int, FNAME char(30), FTYPE char(10), DNAME char(30), length int, IS3D char(1), GRADE int)。

主码为电影编号，IS3D 取值为‘Y’表示是 3D 电影，‘N’表示不是，用户评分规定为 0~100 分之间或者为空值。

演员表【演员编号，演员姓名，性别，出生年份】

ACTOR(ACTID int, ANAME char(30), SEX char(2), BYEAR int)

主码为演员编号。

参演表【演员编号，电影编号，是否主角，用户对该演员在该电影中的评分】

ACTIN(ACTID int, FID int, ISLEADING char(1), GRADE int)

主码、外码请依据应用背景合理定义。ISLEADING 取值为‘Y’表示是，‘N’表示不是主角，也可能取空值，表示不太确定该演员在该电影中是否主角。

GRADE 规定为 0~100 分之间或者为空值。

电影院表【电影院编号，电影院名字，影院所在行政区，影院地址】

THEATER (TID int, TNAME char(20), TAREA char(20), ADDRESS char(30))

主码为电影院编号，影院所在行政区取值如“洪山区”、“武昌区”等等。

上映表【电影编号，影院编号，上映年份，上映月份】

SHOW(FID int, TID int , PRICE int, YEAR int , MONTH int)

假定一部电影在一家影院只上映一次，主码、外码请依据应用背景合理定义。

2) 观察性实验

验证在建立外码时是否一定要参照被参照关系的主码，并在实验报告中简述过程和结果。

2.2 数据更新

1) 数据导入导出

通过查阅 DBMS 资料学习数据导入导出功能，将给定数据文件中的数据导入到相应基表中，然后再将表中数据导出到指定的操作系统文件中。

- 2) 向电影表中插入记录（20197396, 新喜剧之王, 爱情, 周星驰, 93, N, 73）；
- 3) 将电影表中所有周星驰执导的影片类型改为喜剧；
- 4) 删除电影“新喜剧之王”；
- 5) 将演员表中的 90 后演员记录插入到一个新表 YOUNG_ACTOR 中。
- 6) 创建一个视图，该视图记录 80 后演员作为主角参演电影的情况，其中的属性包括：演员编号、演员姓名、出生年份、该演员作为主角参演的电影数量、这些电影的用户评分的最高分。

7) 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。

2.3 数据查询

请分别用一条 SQL 语句完成下列各个小题的查询需求：

- 1) 查询“后来的我们”这部电影在洪山区各家影院的 2017 年的上映情况，并按照上映月份的降序排列；
- 2) 查询所有没有参演演员信息的电影的基本信息；
- 3) 查询所有用户评分低于 80 分或者高于 89 分的电影编号、电影名称、导演姓名及其用户评分，要求 where 子句中只能有一个条件表达式；
- 4) 查询执导过动作片或惊悚片的导演的姓名，要求 where 子句中只能有一个条件表达式；
- 5) 查询所有“巴霍巴利王”系列的电影的编号、电影名称、上映电影院名称及其上映年月；
- 6) 查询每部电影的上映总次数；
- 7) 查询每个演员担任主角的电影中的平均用户评分；
- 8) 查询每个导演所执导的全部影片的最低和最高用户评分；

- 9) 查询至少执导过 2 部电影的导演及其执导电影的数量;
- 10) 查询至少有 2 部电影的用户评分超过 80 分的导演姓名及其执导过的影片数量、平均用户评分;
- 11) 查询至少执导过 2 部电影的导演以及跟这些导演合作过的演员的编号及姓名;
- 12) 查询所有直到 2017 年仍未上映的电影编号、电影名称、导演姓名;
- 13) 查询在各家电影院均上映过的电影编号;
- 14) 查询用户评分超过 90 分的电影的最早上映年月及相应的上映影院编号;
- 15) 查询在同一年月上映过‘30187395’号和‘26942631’号电影的影院编号;
- 16) 查询所有从未参演过用户评分 85 分以下电影的演员的编号、姓名;
- 17) 查询所有的演员的编号、姓名及其参演过的电影名称，要求即使该演员未参演过任何电影也要能够输出其编号、姓名;
- 18) 查询所有上映超过 3 次但没有用户评分的电影编号、名称。
- 19) 通过系统帮助文档学习系统关于时间、日期、字符串类型的函数，为电影表增加首映时间属性，然后查询将在下个月首映的电影信息。（选做）
- 20) 查询所有电影院信息，要求若影院所在行政区为洪山区，则在该列中显示“附近”，否则显示“遥远”。（选做）
- 21) 选择上述任务中某些较为复杂的查询语句，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。（选做）

2.5 事务特性

对以下两个事务，分别在不同的事务隔离级下并发执行，注意尝试各种可能的并发顺序，记录运行结果并分析原因。

2.6 存储函数、存储过程和触发器

- 1) 编写一个依据演员编号计算在其指定年份参演的电影数量的存储函数，并利用其查询 2017 年至少参演过 2 部电影的演员信息。
- 2) 建立每家影院的上映电影总数的统计表，并编写一个存储过程来更新该表。
- 3) 编写一个触发器，用于实现对电影表的完整性控制规则：当增加一部电影时，若导演的姓名为周星驰，则电影类型自动设置为“喜剧”。

3 数据库应用系统设计

自行选择所擅长的 DBMS 软件以及数据库应用系统（客户端程序或者网站）的程序开发工具，参考后面的题目例子，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。

下列题目作为选题背景参考，也可依据这些题目拟定一个自己感兴趣的具有类似工作量和复杂程度的课题。

题目 6：机票预定系统

1、系统功能的基本要求：

每个航班信息的输入。

每个航班的坐位信息的输入；

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；

旅客在飞机起飞前一天凭取票通知交款取票；

旅客能够退订机票；

能够查询每个航班的预定情况、计算航班的满座率。

2、数据库要求：在数据库中至少应该包含下列数据表：

航班信息表；

航班坐位情况表；

旅客订票信息表；

取票通知表；

帐单。

设计一个 B/S 或 C/S 模式的系统实现上述功能。

4 撰写课程实践报告

在课设规定的时间内，撰写并完成课程实践报告。

实践报告由 5 章组成，依次对应下列内容：

1 课程任务概述

简要陈述介绍本实践课程的各项任务要求。

2 软件功能学习

阐述第 1 部分任务的完成过程。

3 SQL 练习

阐述第 2 部分的完成过程。

4 应用系统设计

阐述第 3 部分的完成过程。

5 课程总结

逐条概括、总结此次课程实践的主要工作，阐述此次课程实践的心得体会，展望此次课程实践的有待改进和完善的工作。

其中，第 4 章进一步分 7 个小节，依次是：

4. 1 系统设计目标

4. 2 需求分析

4. 3 总体设计

4. 4 数据库设计

4. 5 详细设计与实现

4. 6 系统测试

4. 7 系统设计与实现总结

目 录

1 课程任务概述.....	1
2 SQL 基础功能学习与实践	2
2.1 任务要求.....	2
2.2 完成过程.....	2
2.3 任务总结.....	21
3 SQL 进阶功能学习与实践	23
3.1 任务要求.....	23
3.2 完成过程.....	23
3.3 任务总结.....	29
4 综合实践任务.....	30
4.1 系统设计目标.....	30
4.2 需求分析.....	30
4.3 总体设计.....	31
4.4 数据库设计.....	34
4.5 详细设计与实现.....	38
4.6 系统优化.....	51
4.6 系统安全性保证.....	51
4.7 系统测试.....	52
4.8 系统设计与实现总结.....	59
5 课程总结.....	60
附录.....	61

1 课程任务概述

本课程实验主要分为三个阶段，第一个阶段是对数据库软件功能进行学习。通过使用 SQL Server 或 DM 或 MySQL 等功能相当的数据库类型产品，学会安装、配置使用的方法。并完成数据定义、数据导入、数据更新、数据查询等数据库基本操作。

第二个阶段是对于数据库进阶技能的一些探究，包括如何使用数据库事务，事务的隔离级别对事务有何影响等。再有就是如何定义数据库的触发器、如何使用数据库的存储过程和存储函数。

在完成了前两个阶段的学习任务之后，需要基于某一个 DBMS 软件以及数据库应用系统的程序开发工具，进行小型数据库应用系统的设计与实现工作。从多个备选题中选出一个题目，进行 B/S 或 C/S 架构的设计，设计中需要分析需求、总体分析、详细设计、测试等。此外还需要考虑数据库设计的正确性问题，确认在并发过程中不会出错，且要具有简单安全性。由于具有针对数据库的 SQL 注入攻击和基于 Browser 脚本的 XSS 攻击，需要在一定程度上避免这些攻击。

总的来说，三次实验的递进层级清晰，便于我们从学习数据库的原理、使用到真正实践，由浅入深全面了解数据库。

2 SQL 基础功能学习与实践

2.1 任务要求

完成下列练习，并在实践报告中叙述过程，可适当辅以截图（篇幅控制在 3 页内）：

- 1) 练习 DBMS (SQL Server / DM / 其他功能相当的数据库产品) 的安装和使用。
- 2) 练习创建新用户并配置权限，要求“2 SQL 练习”部分的操作以新建的普通用户而非管理员用户的身份进行。
- 3) 练习数据库的备份：数据库文件的脱机备份 / DBMS 提供的备份功能。

2.2 完成过程

2.2.1 数据定义及观察性实验

- 0) 事先准备工作。

首先需要下载一个功能齐全的数据库，本文选择了 MySQL 8.0.15 版本作为需要使用的数据库，系统环境如表 2.1。

表 2.1 系统环境说明表

环境表项	说明
操作系统	MacOS Mojave 10.14
系统内核	Darwin Kernel Version 18.5.0
数据库及其版本	MySQL Ver 8.0.15
数据库引擎	InnoDB

其次需要创建一个普通用户来操纵数据库，和一个数据库模式来保存各个表。如代码所示，创建了一个名称为 yongxin，密码为 123456 的用户，以及一个名称为 movie 的模式；将该模式的所有权限通过 root 用户授权给 yongxin。

```
create user 'yongxin'@'localhost' identified by '123456';
create schema movie;
use movie;
grant all privileges on movie.* to yongxin@localhost;
show grants for yongxin@localhost; # 可以通过这个指令查看权限
```

- 1) 定义下列跟电影相关的基表，包括完整性约束的说明。

- **电影表【电影编号，电影名称，电影类型，导演姓名，电影时长（以分钟计），是否 3D，用户评分】**

FILM(FID int, FNAME char(30), FTTYPE char(10), DNAME char(30), length int, IS3D char(1), GRADE int)。

主码为电影编号，IS3D 取值为‘Y’表示是 3D 电影，‘N’表示不是，用户评分规定为 0~100 分之间或者为空值。

解题思路：只需要通过 Create table 指令建立表，并将相应的信息填入即可。注意到 MySQL 8.0.15 版本是不支持 Check 指令的，所以此功能暂时不进行实现，对于 IS3D 属性可以使用 enum(枚举类型)来进行实现。

SQL 语句：

```
create table film (
    fid int primary key,
    fname char(30),
    ftype char(10),
    dname char(30),
    length int,
    is3d enum('Y', 'N'),
    grade int
);
```

主码的关键词是 primary key，其他的关键词只要按照要求的类型定义即可。

- **演员表【演员编号，演员姓名，性别，出生年份】**

ACTOR(ACTID int, ANAME char(30), SEX char(2), BYEAR int)

主码为演员编号。

解题思路：此处与电影表的相似，不再赘述，只给出 SQL 语句。

SQL 语句：

```
create table actor (
    actid int primary key,
    aname char(30),
    sex char(2),
    byear int
);
```

- **参演表【演员编号，电影编号，是否主角，用户对该演员在该电影中的评分】**

ACTIN(ACTID int, FID int, ISLEADING char(1), GRADE int)

主码、外码请依据应用背景合理定义。ISLEADING 取值为‘Y’表示是，‘N’表示不是主角，也可能取空值，表示不太确定该演员在该电影中是否主角。GRADE 规定为 0~100 分之间或者为空值。

解题思路：此处用 enum(‘Y’, ‘N’)替代 char(1)表示 ISLEADING。利用关键词 foreign key(column_name1) references table_name(column_name2) 来表示当前表中的 column_name1 是参照 table_name 中 column_name2 的外码。

SQL 语句：

```
create table actin (
    actid int,
    fid int,
    isleading enum('Y', 'N'),
    grade int,
    foreign key(actid) references actor(actid),
```

```
    foreign key(fid) references film(fid)
);
```

可以看出 actid 参照了 actor 表的 actid, fid 参照了 film 的 fid。

- **电影院表【电影院编号，电影院名字，影院所在行政区，影院地址】**

THEATER (TID int, TNAME char(20), TAREA char(20), ADDRESS char(30))

主码为电影院编号，影院所在行政区取值如“洪山区”、“武昌区”等等。

SQL 语句:

```
create table theater (
    tid int primary key,
    tname char(20),
    tarea char(20),
    address char(30)
);
```

- **上映表【电影编号，影院编号，上映年份，上映月份】**

SHOW(FID int, TID int , PRICE int, YEAR int , MONTH int)

假定一部电影在一家影院只上映一次，主码、外码请依据应用背景合理定义。

SQL 语句:

```
create table showt (
    fid int,
    tid int,
    price int,
    year int,
    month int,
    foreign key(fid) references film(fid),
    foreign key(tid) references theater(tid)
);
```

2) 观察性实验

验证在建立外码时是否一定要参照被参照关系的主码，并在实验报告中简述过程和结果。

解题思路: 对于此实验，首先可以尝试是否可以插入不存在的参照码，然后可以尝试在参照存在的情况下，是否可以删除一个被参照的码。

1. 首先插入一条合法的电影记录

```
insert into film values(20190001, 'movie1', 'horror', 'Jason', 120, 'Y', 90);
```

2. 然后再插入一条合法的演员记录

```
insert into actor values(40, 'David', 'M', 1985);
```

3. 插入一条参照不存在演员编号且参照不存在电影编号的 actin 记录

```
insert into actin values(35, 20191234, 'N', 85); # failed
```

可以看到报错提示，参照了不存在的码，如图 2.1。

4. 插入一条参照存在演员编号且参照存在电影编号的 actin 记录，插入成功。

```
insert into actin values(40, 20190001, 'Y', 95); # succeeded
```

```
[mysql> insert into actin values(35, 20191234, 'N', 85);
ERROR 1216 (23000): Cannot add or update a child row: a foreign key constraint fails
[mysql> insert into actin values(40, 20190001, 'Y', 95);
Query OK, 1 row affected (0.00 sec)
```

图 2.1 参照码观察性实验测试结果图 1

5. 直接删除一位已经被参照的演员(即该演员编号已经存在于 actin 中)。

```
delete from actor where actid = 40;
删除失败，报错如图 2.2
```

```
[mysql> delete from actor where actid = 40;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('movie`.`actin`, CONSTRAINT `actin_ibfk_1` FOREIGN KEY (`actid`) REFERENCES `actor` (`actid`))
mysql> ]
```

图 2.2 参照码观察性实验测试结果图 2

6. 先删除 actin 中的记录，再分别删除该演员信息和电影信息，成功删除。

```
delete from actin where actid = 40;
delete from actor where actid = 40;
delete from film where fid = 20090001;
```

成功删除，结果如图 2.3

```
[mysql> delete from actin where actid = 40;
Query OK, 1 row affected (0.00 sec)

[mysql> delete from actor where actid = 40;
Query OK, 1 row affected (0.00 sec)

[mysql> delete from film where fid = 20190001;
Query OK, 1 row affected (0.01 sec)
```

图 2.3 参照码观察性实验测试结果图 3

2.2.2 数据更新及观察性实验

1) 数据导入导出

通过查阅 DBMS 资料学习数据导入导出功能，将给定数据文件中的数据导入到相应基表中，然后再将表中数据导出到指定的操作系统文件中。

解题思路：其次观察文件存放数据的格式，发现格式类似于.csv 的保存格式，即列之间用逗号隔开，行之间用换行符隔开。以演员表为例，导入文件的格式如下：



图 2.4 电影表导入语句与格式说明

此外，由于 MySQL 的安全设置，还需要在配置文件 my.cnf 中添加如下语句：

```
secure_file_priv = /Users/yongxinxu/Codes/
```

语句说明了准许导入文件的路径，使得用户可以从本地的此路径下导入想要的文件。

其余四张表的导入与其类似，再次只给出 SQL 语句。

SQL 语句：

```
load data infile "/Users/yongxinxu/Codes/film.txt" into table film fields
terminated by ',' optionally enclosed by "" lines terminated by '\n';
load data infile "/Users/yongxinxu/Codes/cinema.txt" into table theater fields
terminated by ',' lines terminated by '\n';
load data infile "/Users/yongxinxu/Codes/actin.txt" into table actin fields
terminated by ',' optionally enclosed by "" lines terminated by '\n';
load data infile "/Users/yongxinxu/Codes/onshow.txt" into table showt fields
terminated by ',' optionally enclosed by "" lines terminated by '\n';
```

2) 向电影表中插入记录 (20197396, 新喜剧之王, 爱情, 周星驰, 93, N, 73);

解题思路：插入记录的语句是 insert into table_name values (col1, col2, col3)，只需要根据此格式插入即可。

SQL 语句：

```
insert into film values(20197396, '新喜剧之王', '爱情', '周星驰', 93, 'N', 73);
```

结果验证：

利用 Select 语句对插入的结果进行验证，或者也可以根据 MySQL 插入之后多少行被影响的反馈得到信息，结果如图 2.5。

Query OK, 1 row affected (0.00 sec) → 插入语句影响的行数

```
mysql> select * from film where fid = 20197396; → 验证查询
+-----+-----+-----+-----+-----+-----+
| fid   | fname | ftype | dname | length | is3d | grade |
+-----+-----+-----+-----+-----+-----+
| 20197396 | 新喜剧之王 | 爱情 | 周星驰 | 93 | N | 73 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

图 2.5 电影表插入结果验证

3) 将电影表中所有周星驰执导的影片类型改为喜剧；

解题思路：利用更新语句，将导演名为周星驰的电影的电影种类全部改成喜剧即可。

SQL 语句：

```
update film set ftype = '喜剧' where dname = '周星驰';
```

结果验证：

利用 Select 语句查询导演名为周星驰的电影即可。如图 2.6

```
[mysql]> select * from film where fname = '周星驰';
+-----+-----+-----+-----+-----+
| fid   | fname        | ftype   | dname    | length | is3d | grade |
+-----+-----+-----+-----+-----+
| 20197396 | 新喜剧之王      | 喜剧     | 周星驰    | 93      | N     | 73    |
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

图 2.6 电影表更新结果验证

- 4) 删除电影“新喜剧之王”;

解题思路: 简单易用删除语句即可。

SQL 语句:

```
delete from film where fname = '新喜剧之王';
```

结果验证:

```
[mysql]> delete from film where fname = '新喜剧之王'; → 删除语句执行
Query OK, 1 row affected (0.01 sec)
```

```
[mysql]> select * from film where fname = '新喜剧之王'; → 删除后结果查询
Empty set (0.00 sec)
```

图 2.7 电影表删除结果验证

- 5) 将演员表中的 90 后演员记录插入到一个新表 YOUNG_ACTOR 中。

解题思路: 创建一个新的表 YOUNG_ACTOR，拥有和普通演员表一样的表项，再通过 insert into 语句加条件实现。

SQL 语句:

```
insert into young_actor select * from actor where byear >= 1990 and byear < 2000;
```

结果验证:

```
+-----+-----+-----+
| actid | fname        | sex   | byear |
+-----+-----+-----+
| 1185637 | 神木隆之介    | 男    | 1993 |
| 1274224 | 周冬雨       | 女    | 1992 |
| 1318811 | 杉咲花       | 女    | 1997 |
| 1320169 | 卡拉·海沃德  | 女    | 1998 |
| 1321661 | 杰克·奎德    | 男    | 1992 |
| 1325127 | 迪丽热巴    | 女    | 1992 |
| 1327806 | 奥利维亚·库克 | 女    | 1993 |
| 1327869 | 陈晓雪       | 女    | 1991 |
| 1328390 | 泰伊·谢里丹  | 男    | 1996 |
| 1339588 | 刘芮麟       | 男    | 1990 |
| 1342488 | 村上虹郎    | 男    | 1997 |
| 1345908 | 林允         | 女    | 1996 |
| 1348583 | 育乃介       | 男    | 1993 |
| 1349142 | 池田依来沙  | 女    | 1996 |
| 1366380 | 谈善言       | 女    | 1990 |
| 1369068 | 森崎温       | 男    | 1990 |
| 1381769 | 苏米特·古拉蒂 | 男    | 1990 |
| 1383684 | 桑贾娜·桑吉  | 女    | 1996 |
| 1387920 | 玛丽·梅曾巴克 | 女    | 1994 |
| 1389281 | 乔希·丹福德  | 男    | 1990 |
+-----+-----+-----+
20 rows in set (0.00 sec)
```

图 2.8 新表插入结果验证

如图 2.8, 满足要求(90 后)的演员都已经被记录到这个新的 YOUNG_ACTOR 表中, 对比插入的数据发现, 既不遗漏, 也无重复。

6) 创建一个视图

该视图记录 80 后演员作为主角参演电影的情况, 其中的属性包括: 演员编号、演员姓名、出生年份、该演员作为主角参演的电影数量、这些电影的用户评分的最高分。

解题思路: 视图作为一个虚拟存在的表, 也可以通过 Select 语句进行条件筛选, 由于需要得到电影的用户评分的最高分, 需要进行多表的 Join, 具体来说是对 actor 演员表和 actin 表表演表还有 film 表进行连接; 以及在 Join 后的表上对(演员, 是否主演)进行集函数统计, 需要用到的集函数是统计 Count(主演)得到主演的次数, 以及 Max(分数)得到最高的评分。

SQL 语句:

```
create view v1 as
select actor.actid as aid, actor.aname as fname, actor.byear as byear,
count(actin.isleading) as cnt, max(film.grade) as highest
from (actor left join actin on actor.actid = actin.actid and actin.isleading =
'Y') left join film on (actin.fid = film.fid)
group by actor.actid, actin.isleading
having actor.byear >= 1980 and actor.byear < 1990;
select * from v1;
```

结果验证:

直接利用 Select 语句进行验证即可

```
|mysql> select * from v1;
+-----+-----+-----+-----+
| aid | fname | byear | cnt | highest |
+-----+-----+-----+-----+
| 1007512 | 郑欣宜 | 1987 | 0 | NULL |
| 1013975 | 瓦内莎·约翰逊 | 1980 | 0 | NULL |
| 1022652 | 格蕾塔·葛韦格 | 1983 | 0 | NULL |
| 1027395 | 泰莎·汤普森 | 1983 | 0 | NULL |
| 1028166 | 山田孝之 | 1983 | 0 | NULL |
| 1037118 | 满岛光 | 1985 | 0 | NULL |
| 1045454 | 松田龙平 | 1983 | 0 | NULL |
| 1049993 | 内哈·迪胡皮阿 | 1980 | 1 | 88 |
| 1050791 | 松田翔太 | 1985 | 0 | NULL |
| 1054453 | 斯嘉丽·约翰逊 | 1984 | 0 | NULL |
| 1054454 | 娜塔莉·波特曼 | 1981 | 0 | NULL |
| 1163626 | 佩蒂塔·维克斯 | 1985 | 0 | NULL |
| 1182849 | 特曼娜·芭蒂亚 | 1989 | 0 | NULL |
| 1201851 | 大卫·吉雅西 | 1980 | 0 | NULL |
| 1215997 | 杰森·格里菲斯 | 1980 | 0 | NULL |
| 1229458 | 安努舒卡·谢蒂 | 1981 | 1 | 70 |
| 1237330 | 德米·垂斯·格罗索 | 1981 | 0 | NULL |
| 1240639 | 加利克·哈尔拉莫夫 | 1980 | 0 | NULL |
| 1248294 | T·J·米勒 | 1981 | 0 | NULL |
| 1255571 | 丽娜·维特 | 1984 | 0 | NULL |
| 1263714 | 萨巴·卡玛尔 | 1984 | 1 | 88 |
```

图 2.9 视图验证结果

7) 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。

解题思路：首先建立一个没有主码的表，然后插入一系列相同的重复元组，对插入结果进行查看；再对某一个条件进行修改，查看修改后的结果，对比是否所有元组都被修改了；最后对某一个条件进行删除，查看删除后的结果，是否所有元组都被删除了。

结果展示：

1. 创建表；

```
create table test_table (
    col1 int,
    col2 char(2),
    col3 int,
    col4 char(3)
);
```

2. 插入 4 条相同记录的元组；

```
insert into test_table values(1, 'a', 3, 'bc');
```

插入结果如图 2.10

```
[mysql> select * from test_table;
+-----+-----+-----+
| col1 | col2 | col3 | col4 |
+-----+-----+-----+
|     1 | a    |     3 | bc   |
|     1 | a    |     3 | bc   |
|     1 | a    |     3 | bc   |
|     1 | a    |     3 | bc   |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

图 2.10 插入后结果

3. 分别按第三列和第一列进行查询修改

```
update test_table set col2 = 'b' where col3 = 3;
update test_table set col2 = 'f' where col1 = 1;
```

修改结果分别如图 2.11 和 2.12，可以看出，所有符合条件的元组都被修改了。换句话说，所有重复的元组都被修改了。

```
[mysql> update test_table set col2 = 'b' where col3 = 3;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

[mysql> select * from test_table;
+-----+-----+-----+
| col1 | col2 | col3 | col4 |
+-----+-----+-----+
|     1 | b    |     3 | bc   |
|     1 | b    |     3 | bc   |
|     1 | b    |     3 | bc   |
|     1 | b    |     3 | bc   |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

图 2.11 按第三列查询，修改第二列后的结果

```

[mysql> update test_table set col2 = 'f' where col1 = 1;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

[mysql> select * from test_table;
+-----+-----+-----+-----+
| col1 | col2 | col3 | col4 |
+-----+-----+-----+-----+
| 1 | f | 3 | bc |
| 1 | f | 3 | bc |
| 1 | f | 3 | bc |
| 1 | f | 3 | bc |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

图 2.12 按第一列查询，修改第二列后的结果

4. 按某一列的条件删除元素

```
delete from test_table where col2 = 'f';
```

如图 2.13，所有重复的，符合条件的元组都被删除了。

```

[mysql> delete from test_table where col2 = 'f';
Query OK, 4 rows affected (0.01 sec)

[mysql> select * from test_table;
Empty set (0.00 sec)

```

图 2.13 删除重复元素结果

2.2.3 数据查询及 3 个选做题

- 1) 查询“后来的我们”这部电影在洪山区各家影院的 2017 年的上映情况，并按照上映月份的降序排列；

解题思路：由于设计到电影明细、上映的明细以及影院的明细，涉及到多表查询，这里使用了子查询的思想，首先先获得后来的我们的电影编号，再在上映表中查到这部电影的上映信息，在上映表和影院表中找到符合这个电影编号，并且在洪山区的数据行，最后按照上映月份分组排序即可。

SQL 语句：

```

select *
from showt join theater on showt.tid = theater.tid
where year = 2018 and theater.tarea = '洪山区' and
showt.fid = (select fid
from film
where fname like '%后来的我们%')
order by showt.month desc;

```

结果展示：由于 2017 没有满足条件的，这里手动把题干改成了 2018 年，输出结果如图 2.14 所示。

```

mysql> select *
-> from showt join theater on showt.tid = theater.tid
-> where year = 2018 and theater.tarea = '洪山区' and
-> showt.fid = (select fid
-> from film
-> where fname like '%后来的我们%')
-> order by showt.month desc;

```

图 2.14 查询第一题结果

- 2) 查询所有没有参演演员信息的电影的基本信息；

解题思路：需要用到差集的思想，首先找到参演表中参演过电影的演员的 id 集合，再在所有演员中查找 id 不再前述集合的演员即可。

SQL 语句：

```
select *
  from film
 where fid not in
(
  select fid
    from actin
   group by fid
  having count(distinct actin.actid) > 0
);
```

- 3) 查询所有用户评分低于 80 分或者高于 89 分的电影编号、电影名称、导演姓名及其用户评分，要求 where 子句中只能有一个条件表达式；

解题思路：可以用 between 语句，也可以利用特殊性用除法的余数不等于 8，此处两种方法都给出展示。

SQL 语句(方法一)：

```
select fid, fname, dname, grade
  from film
 where grade between (80, 89);
```

SQL 语句(方法二)：

```
select fid, fname, dname, grade
  from film
 where grade div 10 <> 8;
```

- 4) 查询执导过动作片或惊悚片的导演的姓名，要求 where 子句中只能有一个条件表达式；

解题思路：用 in 谓词就可以解决问题

SQL 语句：

```
select dname
  from film
 where ftype in ('动作', '惊悚');
```

- 5) 查询所有“巴霍巴利王”系列的电影的编号、电影名称、上映电影院名称及其上映年月；

解题思路：这个题目本质上是用到了模糊查询，用到 like 谓词

SQL 语句：

```
select showt.fid as fid, film.fname as fname, theater.tname as tname,
showt.year as tyear, showt.month as tmonth
  from (film join showt on film.fid = showt.fid) join theater on showt.tid =
theater.tid
 where fname like '%巴霍巴利王%';
```

结果展示：

查询结果如图 2.15 展示，通过结合数据库导入数据可知，结果正确，所有巴霍巴利王系列电影都在查询结果中。

```
mysql> select showt.fid as fid, film.fname as fname, theater.tname as tname, showt.ye
ar as tyear, showt.month as tmonth
-> from (film join showt on film.fid = showt.fid) join theater on showt.tid = the
ater.tid
[ -> where fname like '%巴霍巴利王%';
]
+-----+-----+-----+-----+
| fid | fname | tname | tyear | tmonth |
+-----+-----+-----+-----+
| 26420932 | 巴霍巴利王2: 终结 | 紫星影城 | 2017 | 2 |
| 26420932 | 巴霍巴利王2: 终结 | 博影时代影城 | 2017 | 1 |
| 26420932 | 巴霍巴利王2: 终结 | 横店电影城 | 2017 | 2 |
| 26430636 | 巴霍巴利王1 | 耀莱成龙国际影城 | 2018 | 12 |
| 26430636 | 巴霍巴利王1 | 正华银兴影城 | 2018 | 12 |
| 26430636 | 巴霍巴利王1 | 卢米埃影城 | 2018 | 12 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

图 2.15 查询第五题结果

- 6) 查询每部电影的上映总次数；

解题思路：将上映表和电影表自然连接后用 Count 集函数统计电影次数即可。

SQL 语句：

```
select film.fid as fid, film.fname as fname, count(fid) as cnt
from showt join film on showt.fid = film.fid
group by fid;
```

结果展示：

```
+-----+-----+-----+
| fid | fname | cnt |
+-----+-----+-----+
| 4920389 | 头号玩家 | 4 |
| 5330387 | 青年马克思 | 4 |
| 25881611 | 战神纪 | 4 |
| 26384741 | 淹灭 | 4 |
| 26420932 | 巴霍巴利王2: 终结 | 3 |
| 26430636 | 巴霍巴利王1 | 3 |
| 26588783 | 冰雪女王3: 火与冰 | 4 |
| 26640371 | 犬之岛 | 3 |
| 26661189 | 脱单告急 | 4 |
| 26683723 | 后来的我们 | 4 |
| 26691361 | 21克拉 | 4 |
| 26769474 | 香港大营救 | 4 |
| 26774933 | 幕后玩家 | 3 |
| 26868408 | 黄金花 | 4 |
| 26924141 | 低压槽：欲望之城 | 13 |
| 26935777 | 玛丽与魔女之花 | 3 |
| 26942631 | 起跑线 | 4 |
| 30187395 | 午夜十二点 | 4 |
+-----+-----+-----+
18 rows in set (0.00 sec)
```

图 2.16 查询第六题结果

- 7) 查询每个演员担任主角的电影中的平均用户评分；

解题思路：根据演员的 id，连接演员表和参演表，并统计平均用户评分即可。

SQL 语句：

```
select actor.actid as actid, actor.aname as fname, avg(actin.grade) as
avg_grade
from actor join actin on actor.actid = actin.actid and actin.isleading = 'Y'
group by actid;
```

结果展示：

如图 2.17，这里只展示了部分的结果。

actid	aname	avg_grade
1274628	井柏然	57.0000
1274224	周冬雨	91.0000
1274287	田壮壮	66.0000
1355411	曲哲明	67.0000
1358221	刘启恒	96.0000
1275538	苏小明	63.0000
1318927	邹倚天	69.0000
1374201	张子贤	76.0000
1314249	施予斐	65.0000
1392358	李剑青	65.0000
1274297	徐峥	53.0000
1313574	王丽坤	80.0000
1322085	王砚辉	84.0000
1313011	段博文	87.0000
1831194	任达华	67.0000
1313742	于和伟	90.0000
1317092	朱珠	92.0000
1322795	赵达	56.0000
1313761	谢楠	69.0000
1381851	鲍晓	91.0000
1337710	帕拉巴斯	86.0000
1350818	拉纳·达格巴提	51.0000

图 2.17 查询第七题结果

- 8) 查询每个导演所执导的全部影片的最低和最高用户评分；

解题思路：在上映表中获得导演名字，在用导演名字分组的情况下，用最高和最低集函数统计评分即可。

SQL 语句：

```
select dname, min(grade) as lowwest, max(grade) as highest
from film
group by dname;
```

结果展示：

如图 2.18，统计出了这些导演的电影中的最低和最高用户评分。

```
mysql> select dname, min(grade) as lowwest, max(grade) as highest
-> from film
[ -> group by dname;
+-----+-----+
| dname | lowwest | highest |
+-----+-----+
| 任鹏远 | 60 | 89 |
| 哈乌·佩克 | 64 | 64 |
| 哈斯朝鲁 | 36 | 36 |
| 亚历克斯·嘉兰 | 73 | 73 |
| 阿列克谢·特斯蒂斯林 | 44 | 44 |
| 何念 | 52 | 88 |
| 柯孟融 | 56 | 56 |
| 刘若英 | 58 | 58 |
| 刘一君 | 44 | 44 |
| 张家辉 | 47 | 47 |
| 米林宏昌 | 0 | 0 |
| 张孝爽 | 0 | 0 |
+-----+-----+
12 rows in set (0.00 sec)
```

图 2.18 查询第八题结果

- 9) 查询至少执导过 2 部电影的导演及其执导电影的数量；

解题思路：在电影表中选择导演名称，和电影数量，用导演名称作为分组条件，条件是 Count(fid)>=2。

SQL 语句：

```
select dname, count(fid)
from film
group by dname
having count(fid) >= 2;
```

结果展示：

如图 2.19，统计出了这些导演及他们参演的电影数量。

```
mysql> select dname, count(fid)
-> from film
-> group by dname
[ -> having count(fid) >= 2;
+-----+
| dname | count(fid) |
+-----+
| 任鹏远 |        4 |
| 何念  |        4 |
+-----+
2 rows in set (0.00 sec)
```

图 2.19 查询第九题结果

- 10) 查询至少有 2 部电影的用户评分超过 80 分的导演姓名及其执导过的影片数量、平均用户评分；

解题思路：建立子查询，首先根据导演姓名分组，查询超过 80 分的，且多余 2 部电影的导演姓名。父查询中连接电影和参演表，并在父查询中用 in 谓词 where 判断是否满足子查询的条件，再根据导演姓名分组。

SQL 语句：

```
select dname, count(*), avg(actin.grade)
from film join actin on film.fid = actin.fid
where dname in (
    select dname
    from film
    where grade > 80
    group by dname
    having count(*) >= 2
)
group by dname;
```

结果展示：

```
+-----+
| dname | count(*) | avg(actin.grade) |
+-----+
| 何念  |      52 |       74.2500 |
+-----+
1 row in set (0.00 sec)
```

图 2.19 查询第十题结果

- 11) 查询至少执导过 2 部电影的导演以及跟这些导演合作过的演员的编号及姓名;

解题思路: 涉及到三张表, 首先在子查询中查到至少执导过 2 部电影的导演, 然后在父查询中把它作为子条件, 用 in 谓词查找; 父查询中电影、演员、参演三表自然连接, 最终选择需要的项输出即可。

SQL 语句:

```
select film.dname as dname, actor.actid as actid, actor.aname as fname
from (film join actin on film.fid = actin.fid) join actor on actor.actid =
actin.actid
where film.dname in
(select dname
from film
group by dname having count(fid) >= 2);
```

由于结果过多, 不具有代表性, 此题不展示结果。

- 12) 查询所有直到 2017 年仍未上映的电影编号、电影名称、导演姓名;

解题思路: 由于可能上映时间为 NULL, 所以不能直接通过查找 2017 年之后时间的电影, 这里的做法是利用类似差集的思想, 查找 2017 年及以前上映的电影, 在全集中减掉它, 即谓词 not in 的方法。

SQL 语句:

```
select distinct film.fid as fid, film.fname as fname, film.dname as dname,
showt.year as earliest_time
from film left join showt on film.fid = showt.fid
where film.fid not in (
    select showt.fid
    from showt
    group by showt.fid
    having min(showt.year) <= 2017
);
```

结果展示:

不存在上映时间为 NULL 的电影, 展示结果如图 2.20

fid	fname	dname	earliest_time
4920389	头号玩家	任鹏远	2018
25881611	战神纪	哈斯朝鲁	2018
26384741	湮灭	亚历克斯·嘉兰	2018
26430636	巴霍巴利王 1	任鹏远	2018
26640371	犬之岛	何念	2018
26661189	脱单告急	柯孟融	2018
26683723	后来的我们	刘若英	2018
26691361	21克拉	何念	2018
26769474	香港大营救	刘一君	2018
26774033	幕后玩家	任鹏远	2018
26924141	低压槽：欲望之城	张家辉	2018
30187395	午夜十二点	张孝爽	2018

图 2.20 查询第十二题结果

- 13) 查询在各家电影院均上映过的电影编号;

解题思路: 这题需要用到双重否定的思想。在各家电影院上映过的同义是没有在任何一家电影院上映，这些电影院不在没有上映的电影院中。

SQL 语句:

```
select fid
from film
where not exists (
    select*
    from theater
    where not exists
    (
        select *
        from showt
        where theater.tid = showt.tid and film.fid = showt.fid
    )
);
```

结果展示:

查询结果如图 2.21 所示

```
mysql> select fid
-> from film
-> where not exists (
->     select*
->     from theater
->     where not exists
->     (
->         select *
->         from showt
->         where theater.tid = showt.tid and film.fid = showt.fid
->     )
-> );
+-----+
| fid |
+-----+
| 26924141 |
+-----+
1 row in set (0.00 sec)
```

图 2.21 查询第十三题结果

- 14) 查询用户评分超过 90 分的电影的最早上映年月及相应的上映影院编号;

解题思路: 这题首先在最底层的查询中查找用户评分超过 90 分的电影，然后选择出上映年月最早的电影编号，最后与上映表连接即可。

SQL 语句: (由于超过 90 的满足条件的没有，此处改成了超过 80 分)

```
select *
from showt join
    (select tb1.fffid as fffid, min(tb1.syear * 100 + tb1.smonth) as sstime
     from
        (select film.fid as ffid, showt.tid as std, showt.year as syear,
        showt.month as smonth
         from film join showt on film.fid = showt.fid
         where film.grade > 80) as tb1
     group by tb1.fffid) as tb2
on showt.fid = tb2.fffid and (showt.year * 100 + showt.month) = tb2.sstime;
```

结果展示：

查询结果如图 2.22 所示

<i>fid</i>	<i>tid</i>	<i>price</i>	<i>year</i>	<i>month</i>	<i>ffffid</i>	<i>sstime</i>
4920389	2	25	2018	4	4920389	201804
4920389	6	33	2018	4	4920389	201804
4920389	13	27	2018	4	4920389	201804
4920389	7	25	2018	4	4920389	201804
26640371	5	25	2018	7	26640371	201807
26942631	4	23	2017	4	26942631	201704

图 2.22 查询第十四题结果

- 15) 查询在同一年月上映过 ‘30187395’ 号和 ‘26942631’ 号电影的影院编号；

解题思路：首先查找上映过 ‘26942631’ 号电影的月份和年份，然后再上映 ‘30187395’ 号电影的影院中查找相同的月份和年份即可。

SQL 语句：

```
select tid
from showt
where fid = 30187395 and (year, month) in
(
    select year, month from showt
    where fid = 26942631
);
```

- 16) 查询所有从未参演过用户评分 85 分以下电影的演员的编号、姓名；

解题思路：在演员表和参演表的自然连接中选择满足不存在有 85 分以下电影记录的演员的编号、姓名。用到 Not Exist 谓词。

SQL 语句：

```
select actor.actid, actor.aname
from actin join actor on actin.actid = actor.actid
where not exists
(select *
from film
where grade < 85 and actin.fid = fid);
```

由于结果很多，此处不展示查询结果。

- 17) 查询所有的演员的编号、姓名及其参演过的电影名称，要求即使该演员未参演过任何电影也要能够输出其编号、姓名；

解题思路：对于即使该演员未参演过任何电影也要能够输出其编号、姓名的要求，用到 left join 即可。

SQL 语句：

```
select actor.actid, actor.aname, film.fname
```

```
from (actor left join actin on actor.actid = actin.actid) left join film on
actin.fid = film.fid;
```

- 18) 查询所有上映超过 3 次但没有用户评分的电影编号、名称。

解题思路: 没有用户评分用 grade is null 来表示，在子查询中首先查询上映超过 3 次的电影。

SQL 语句:

```
select fid, fname
from film
where grade is null and fid in
(select fid
from showt
group by fid
having count(*) > 3);
```

由于查询结果是空集，再次不显示。

- 19) 通过系统帮助文档学习系统关于时间、日期、字符串类型的函数，为电影表增加首映时间属性，然后查询将在下个月首映的电影信息。(选做)

解题步骤: 首先需要添加一个表示首映时间的列，这里只需要日期就够了，所以用 date 即可。

```
alter table film add fstime date default null;
```

然后再给某一些电影更新首映时间。

```
update film set fstime = date '2019-05-13' where fname like '%香港大营救%';
```

然后查找下一个月首映的电影，下一个月的表示方法应该是 1、年份相同，取出月份，比较它们的差是否小于等于 1，2、年份不同且当月份为 12 月时，需要看是否是第二年的一月份或当年的 12 月份。

SQL 语句:

```
select *
from film
where (year(curdate()) = year(fstime) and (month(fstime) - month(curdate()))
between 0 and 1))
or
(year(curdate()) - year(fstime) = 1 and month(curdate()) = 12 and month(fstime
= 1));
```

结果展示:

查询结果如图 2.23 所示

```

mysql> select *
-> from film
-> where (year(curdate()) = year(fstime) and (month(fstime) - month(curdate()) between 0 and 1))
-> or
[ -> (year(curdate()) - year(fstime) = 1 and month(curdate()) = 12 and month(fstime)
= 1));
+-----+
| fid | fname           | ftype | dname      | length | is3d | grade | fstime
|-----+
| 26769474 | 香港大营救    | 动作   | 刘一君     | 95     | N    | 44    | 2019-05
|-13 00:00:00 |
+-----+

```

图 2.23 查询第十九题结果

- 20) 查询所有电影院信息，要求若影院所在行政区为洪山区，则在该列中显示“附近”，否则显示“遥远”。(选做)

解题思路：在 MySQL 中利用选择中列的 if as 语句即可。

SQL 语句：

```
select *, if (theater.tarea = '洪山区', '附近', '遥远') as far
from theater;
```

结果展示：

查询结果如图 2.24 所示

```

mysql> select *, if (theater.tarea = '洪山区', '附近', '遥远') as far
-> from theater;
+-----+
| tid | tname           | tarea      | address          | far
|-----+
| 1   | 紫星影城         | 江岸区    | 台北路 153号    | 遥远
| 2   | 博影时代影城     | 楚口区    | 航空路 13号    | 遥远
| 3   | 中影国际影城     | 洪山区    | 光谷步行街 C区 3楼 | 附近
| 4   | CGV影城          | 洪山区    | 光谷步行街 4期 8号 楼 3楼 | 附近
| 5   | 正华银兴影城     | 洪山区    | 民族大道 118号 时间广场三楼 | 附近
| 6   | 幸福蓝海影城     | 洪山区    | 雄楚大道 888号 4楼 | 附近
| 7   | 华夏国际影城     | 洪山区    | 路喻 726号 鲁巷广场七楼 | 附近
| 8   | 巨幕影城          | 洪山区    | 光谷资本大厦 4楼 | 附近
| 9   | 罗莱成龙国际影城 | 洪山区    | 路喻光谷国际广场 2期 | 附近
| 10  | 卢米埃影城        | 武昌区    | 楚河汉街 凯德 1818广场 7楼 | 遥远
| 11  | 华谊兄弟影院      | 黄陂区    | 黄陂大道 387号 黄陂广场 C座 | 遥远
| 12  | 恒大影城          | 黄陂区    | 盘龙城巨龙大道与兴龙路交汇 | 遥远
| 13  | 横店电影城        | 东湖区    | 常青花园地铁站汇和城五楼 | 遥远
+-----+
13 rows in set (0.02 sec)

```

图 2.24 查询第二十题结果

- 21) 选择上述任务中某些较为复杂的查询语句，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。(选做)

注：说在最前面，这里是根据网络博客，利用了老师上课说的 S、C、SC 表，自己加以实验写成的。

1、课程表的定义如下：(共 100 条数据)

```
create table Course(
c_id int PRIMARY KEY,
name varchar(10)
)
```

2、学生表的定义如下：(共 70000 条数据)

```
create table Student(
id int PRIMARY KEY,
name varchar(10)
)
```

3、学生成绩表定义如下：(共 70 万条数据)

```
create table SC(
    sc_id int PRIMARY KEY,
    s_id int,
    c_id int,
    score int
)
```

查询的目标是要查找到语文考 100 分的学生，下面给出了第一种查询方法。

```
select s.* from Student s where s.s_id in (select s_id from SC sc where sc.c_id = 0 and sc.score = 100 )
```

执行时间是 30248.271s

究其原因，可以查看查询计划。

Explain

```
select s.* from Student s where s.s_id in (select s_id from SC sc where sc.c_id = 0 and sc.score = 100 )
```

可以得到以下的表格：

表 2.2 原始查询计划表格

Id	Select_type	Table	Type	Possible_keys	Key	Key_len	Ref	Rows	Extra
1	PRIMARY	s	All	NULL	NULL	NULL	NULL	70007	Using where
2	DEPENDENT SUBQUERY	sc	All	NULL	NULL	NULL	NULL	770011	Using where

经过观察，可以发现没有用到索引，type 全是 ALL，那么首先改进的是建立一个索引，建立索引的字段当然是在 where 条件的字段。

```
CREATE index sc_c_id_index on SC(c_id);
CREATE index sc_score_index on SC(score);
```

再次执行上述查询语句，时间为: 1.054s，再次查看查询计划表格，得到如表 2.3 的结果。

表 2.3 建立索引后的查询计划表格

Id	Select_type	Table	Type	Possible_keys	Key	Key_len	Ref	Rows	Extra
1	PRIMARY	s	All	NULL	NULL	NULL	NULL	70007	Using where
2	DEPENDENT SUBQUERY	sc	Ref	sc_s_id_index, sc_score_index, sc_c_id_index	sc_score_index, 5	const	8	Using where	

可以看到经过简历索引后的查询利用了索引，优化了许多。

最后，可以利用下面的指令查看 MySQL 查询器的优化计划

```
explain extended
select s.* from Student s where s.s_id in (select s_id from SC sc where sc.c_id = 0 and sc.score = 100 )
```

优化后的计划如下：最后的执行结果为 0.1s

```

SELECT
    `YSB`.`s`.`s_id` AS `s_id`,
    `YSB`.`s`.`name` AS `name`
FROM
    `YSB`.`Student` `s`
WHERE
    < in_optimizer > (
        `YSB`.`s`.`s_id` ,< EXISTS > (
            SELECT
                1
            FROM
                `YSB`.`SC` `sc`
            WHERE
                (
                    (`YSB`.`sc`.`c_id` = 0)
                    AND (`YSB`.`sc`.`score` = 100)
                    AND (
                        < CACHE > (`YSB`.`s`.`s_id`) = `YSB`.`sc`.`s_id`
                    )
                )
        )
    )
)

```

2.3 任务总结

对于本阶段的任务，主要熟悉了 MySQL 的基础功能以及基本配置应用。包括数据的定义，包括创建表，主键，索引，外键等；数据的导入，通过从外界的格式化文件读取到表中的方法进行数据导入；数据的查询，包括单表查询、集函数查询、多表连接查询、嵌套查询等等方法解决各种查询问题。

遇到的问题主要有三个：

问题一：对于 MySQL 的操作不熟练。举一个典型的例子来说，在导入数据时，在网上查到多种方法，有 load local file、load file 等，在仔细阅读了语法规范以后写出语句也不能导入，显示不具备这样的权限(不信任那个文件夹下的原始数据文件)。

解决方案：首先还是借助 MySQL 的文档，得到基本的语法摘要。像此处距离的 load local file、load file，以及 terminated by 等间隔符的详细选项。再语法不出现错误之后，根据 MySQL 的错误提示，在 Stack Overflow 上进行查询原因。此处发现了 MySQL 在我电脑上默认的信任文件夹为空，需要通过在配置文件 my.cnf 中进行设置文件目录。配置成功后如图 2.25 所示，方可正常导入。

```

mysql> show variables like 'secure_file_priv';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| secure_file_priv | /Users/yonginxu/Codes/ |
+-----+-----+
1 row in set (0.00 sec)

```

图 2.25 配置导入文件目录结果

总结：结合官方文档和著名论坛解决自己的问题。

问题二：不知道 MySQL 命令行如何进行查询计划查询，以及如何进行基础的优化。

解决方案：查看了网上的各种博客，包括国内外各大计数论坛，并私信了博

主，知道了用 Explain 可以看到原始的查询计划。用 Explain Extended 可以查询到优化后的计划。

总结：查询优化非常重要，从实验结果也可以看到，从 30000+秒优化到 0.1 秒，虽然数据库一般都会进行自动的优化，但是也要尽量的写出更优化的查询语句。

问题三：会因为看错查询功能而写错查询语句。由于描述本身比较复杂，有可能因为疏忽或者漏看，写错查询语句。

解决方案：与同学交流查询思路，从而发现错误。

总结：这个不是在数据库本身上犯的错误，但是也是一个非常关键的错误。实际应用中，很可能因为对需求的错误分析，或者理解错误，也写出了错误的查询语句，造成查询结果与要求背道而驰，这是非常严重的错误，最好组内讨论。

3 SQL 进阶功能学习与实践

3.1 任务要求

- 1) 理解并且实践数据库关于事务相关的原理，不同隔离集的情况下会对数据库事务具有哪些影响；
- 2) 理解并实践存储函数和存储过程，体会它们之间的区别；
- 3) 理解并实践触发器，实现功能。

3.2 完成过程

- 1) 对以下两个事务，如图 3.1 所示，分别在不同的事务隔离级下并发执行，注意尝试各种可能的并发顺序，记录运行结果并分析原因。

已知表 R(A, B)中的数据如右表所示，
现有并发执行的事务 T1 和 T2 如下：

T1: BEGIN TRANSACTION;
 UPDATE R SET B=22 WHERE A='C1';
 INSERT INTO R VALUES ('C4', 0);
 UPDATE R SET B=38 WHERE A='C1';
 COMMIT;

T2: BEGIN TRANSACTION;
 SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
 SELECT SUM(B) FROM R;
 SELECT AVG(B) FROM R;
 COMMIT;

请给出 T2 中两个 SELECT 语句所有可能的查询结果。(6 分)

A	B
C1	40
C2	50
C3	60

图 3.1 题目一配图

解题思路：可以看到，在 T2 执行的过程中的第一句话，重新设置了隔离级别为读提交。所以这之后整个过程中，T2 都是处于 Read Committed 的隔离级别的。主要分两种情况讨论。

第一种情况是初始两个事务的隔离级别都是 Serializable，那么两个事务必须按照可串行化的顺序串行执行。那么会出现两种结果。

1. 执行顺序如下：

```
# both
set session transaction isolation level serializable;
show variables like '%isolation%';

T1:
start transaction;
T2:
start transaction;
T1:
update R set B = 22 where A = 'C1';
insert into R values('C4', 0);
update R set B = 38 where A = 'C1';
commit;
T2:
set session transaction isolation level read committed;
select sum(B) from R;
select avg(B) from R;
commit;
```

图 3.2 Serializable 下结果一

执行后的结果是 SUM 为 148, AVG 为 37。

2. 执行顺序如下：

```
# both
set global transaction isolation level serializable;
show variables like '%isolation%';

T1:
start transaction;
T2:
start transaction;
T2:
set session transaction isolation level read committed;
select sum(B) from R;

select avg(B) from R;
commit;
T1:
update R set B = 22 where A = 'C1';
insert into R values('C4', 0);
update R set B = 38 where A = 'C1';
commit;
```

图 3.3 Serializable 下结果二

执行后的结果是 SUM 为 150, AVG 为 50。

在隔离级别为非 Serializable，即为{Read Uncommitted, Read Committed, Read Repeatable}中的一个时，这时候主要看 T1 的语句在 T2 两个 Select 之前，两个 Select 语句之间或者是两个 Select 语句之后，共产生三种不同的结果。

1. 两个 Select 语句都在最开始执行时。

```
# both
set session transaction isolation level xxxx;
show variables like '%isolation%';

T1:
start transaction;
T2:
start transaction;
T2:
set session transaction isolation level read committed;
select sum(B) from R;
select avg(B) from R;
commit;
T1:
update R set B = 22 where A = 'C1';
insert into R values('C4', 0);
update R set B = 38 where A = 'C1';
commit;
```

图 3.4 非 Serializable 下结果一

此时，SUM 为 148, AVG 为 37。与 Serializable 的第一种结果是相同的。实际上这种结果本质就是串行执行，相当于先 T2 执行完，提交之后再执行 T1 的多条语句。

同样的，第二种结果就是先执行完 T1，再执行 T2。

2. 两个 Select 都语句在 T1 事务之后。

```
# both
set session transaction isolation level serializable;
show variables like '%isolation%';

T1:
start transaction;
T2:
start transaction;
T2:
set session transaction isolation level read committed;
select sum(B) from R;
T1:
update R set B = 22 where A = 'C1';
insert into R values('C4', 0);
update R set B = 38 where A = 'C1';
commit;
T2:
select avg(B) from R;
commit;
```

图 3.5 非 Serializable 下结果二

此时，SUM 为 148，AVG 为 37。

3. 两个 Select 分别在 T1 事务的前后。

```
# both
set session transaction isolation level serializable;
show variables like '%isolation%';

T1:
start transaction;
T2:
start transaction;
T2:
set session transaction isolation level read committed;
select sum(B) from R;
T1:
update R set B = 22 where A = 'C1';
insert into R values('C4', 0);
update R set B = 38 where A = 'C1';
commit;
T2:
select avg(B) from R;
commit;
```

图 3.6 非 Serializable 下结果三

此时，SUM 为 150，AVG 为 37。

总的来说，此题需要分事务的隔离级别是 Serializable 或者是非 Serializable 两种大情况。

对于 Serializable，总共有两种可能的结果。

对于非 Serializable，总共有三种可能的结果。

- 2) 编写一个依据演员编号计算在其指定年份参演的电影数量的存储函数，并利用其查询 2017 年至少参演过 2 部电影的演员信息。

解题思路：存储函数本质上是通过传入若干参数，通过 Select 语句，获得一个结果的一种函数。本题主要就是利用指定年份和演员编号，得到这一年这个演员的电影数量。

SQL 语句：

```
delimiter $$  
drop function if exists get_actin_cnt $$  
create function get_actin_cnt(act_id int, year_in int) returns int  
deterministic  
begin  
    declare cnt int default 0;  
    select count(fid) into cnt  
    from actin  
    where actid = act_id and fid in (select distinct fid from showt where  
year = year_in)  
    group by actid  
    return cnt;  
end$$  
delimiter ;
```

如上面的语句，主要是定义了一个函数 get_actin_cnt，以一个特定的演员编号和特定年份作为参数，返回一个 int。

函数的主题首先定义了一个 cnt 存储结果，之后只需要通过 select 语句，将结果存入 cnt，并返回 cnt 即可。

利用存储函数进行测试和查询：

```
select get_actin_cnt(1108861, 2017);
```

用上述的语句测试 2017 年编号为 1108861 的演员参演的电影数量，对比数据库的表中知道结果正确。

利用其查询 2017 年至少参演过 2 部电影的演员信息的 SQL 语句如下：

```
select * from actor  
where get_actin_cnt(actid, 2017) >= 1;
```

测试结果是 Empty Set，根据 Select 语句和数据库表，知道结果正确。

- 3) 建立每家影院的上映电影总数的统计表，并编写一个存储过程来更新该表。

解题思路：与存储函数不同的是，存储过程可以返回完整的一张表，并且可以对表进行过程型的操作，比如说 Insert、Update、Delete 等。本题首先需要建立一个没加影院上映电影总数的统计表，建立新表的 SQL 语句如下：

```
create table show_count(  
    tid int,  
    cnt int
```

```
);
```

这个存储过程不需要接收参数，此处将通过它的完整 SQL 语句进行说明。

SQL 语句：

```
1. delimiter $$  
2. drop procedure if exists update_show_count_table $$  
3. create procedure update_show_count_table()  
4. begin  
5. declare theater_id int;  
6. declare fid_cnt int;  
7. declare done int default 0;  
8. -- 定义游标  
9. declare rs_cursor cursor for  
10. select theater.tid, count(fid) from theater join showt on theater.tid  
= showt.tid  
11. group by theater.tid;  
12. declare continue handler for not found set done = 1;  
13. open rs_cursor;  
14. cursor_loop:LOOP  
15. fetch rs_cursor into theater_id,fid_cnt; -- 取数据  
16. if done = 1 then  
17. leave cursor_loop;  
18. end if;  
19. -- 更新表  
20. if theater_id not in (select tid from show_count) then  
21. insert into show_count values(fid_cnt, theater_id);  
22. else update show_count set show_count.cnt = fid_cnt where  
show_count.tid = theater_id;  
23. end if;  
24. end loop cursor_loop;  
25. close rs_cursor;  
26. end $$  
27. delimiter ;
```

如上述的语句，第三行定义了这个存储过程，第 5~6 行定义了这个存储过程的 2 个参数，这两个参数也对应了新创建表的元素，第 7 行定义了结束标志。第 9~18 行定义了游标，第 10 行得到每一个游标获得的影院 ID，以及对应上映的次数。第 19 行开始是利用游标每一次移动获得的 5~6 行的数据，如果这个影院 ID 已经在新表中，则更新它上映的数量，否则插入(影院 ID，上映电影数量)的新元组。

结果测试：

```
select * from show_count;  
insert into showt values(4920389, 1, 28, 2018, 4);  
call update_show_count_table();  
select * from show_count;
```

上述是新插入上映电影的例子，然后通过调用定义的存储过程，比较变化。

插入元素之前：

```
|mysql> select * from show_count;
+-----+-----+
| tid | cnt |
+-----+-----+
| 1   | 5   |
| 2   | 6   |
| 3   | 4   |
| 4   | 5   |
| 5   | 8   |
| 6   | 5   |
| 7   | 7   |
| 8   | 6   |
| 9   | 7   |
| 10  | 5   |
| 11  | 4   |
| 12  | 5   |
| 13  | 9   |
+-----+-----+
13 rows in set (0.00 sec)
```

图 3.7 插入元素之前结果

可以看到影院 ID 为 1 的电影院总共上映了 5 部电影。然后通过插入元素给影院 ID 为 1 的电影院新上映了一部电影。调用存储过程后再次查询结果如图 3.8 所示。

```
|mysql> call update_show_count_table();
Query OK, 0 rows affected (0.01 sec)

|mysql> select * from show_count;
+-----+-----+
| tid | cnt |
+-----+-----+
| 1   | 6   |
| 2   | 6   |
| 3   | 4   |
| 4   | 5   |
| 5   | 8   |
| 6   | 5   |
| 7   | 7   |
| 8   | 6   |
| 9   | 7   |
| 10  | 5   |
| 11  | 4   |
| 12  | 5   |
| 13  | 9   |
+-----+-----+
13 rows in set (0.00 sec)
```

图 3.8 插入元素之后结果

可以看到，结果符合预期。

- 4) 编写一个触发器，用于实现对电影表的完整性控制规则：当增加一部电影时，若导演的姓名为周星驰，则电影类型自动设置为“喜剧”。

解题思路：MySQL 的触发器语句比较标准，由于此题比较简单，直接展示触发器实现语句。

SQL 语句：

```
delimiter $$  
drop trigger if exists trg1 $$  
create trigger trg1 before insert  
on film for each row  
begin  
    if new.dname = '周星驰' then  
        set new.ftype = '喜剧';  
    end if;  
end$$
```

```
delimiter ;
```

定义了一个名称为 trg1 的触发器，当新插入的导演名称为周星驰时，则把新插入的电影种类改为喜剧。

测试结果：

```
insert into film values(20197396, '新喜剧之王', '爱情', '周星驰', 93, 'N',  
73, date '2019-05-15');  
select * from film where dname = '周星驰';
```

测试语句如上，插入一条新记录，为周星驰的电影，种类为爱情，之后再查看结果。

```
mysql> select * from film where dname = '周星驰';  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| fid | fname | ftype | dname | length | is3d | grade | fstime |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| 20197396 | 新喜剧之王 | 喜剧 | 周星驰 | 93 | N | 73 | 2019-05-15 00:00:00 |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.01 sec)
```

图 3.9 插入元素之后结果

如图 3.9 所示，电影类别被改为了喜剧。

3.3 任务总结

本次实验更多的是体验到了数据库的一些进阶操作，包括 MySQL 的事务，存储函数、存储过程以及触发器。

通过实验，我体会到了不同事务的隔离级别之间有怎样的区别。不仅如此，我还尝试了不同事物隔离级别所带来的不同的问题。包括重复查询带来的问题、不可重复读、幻读等等。

存储函数主要用于利用一些常用操作进行查询，本质上与视图很类似。但它的粒度要更细一点，根据一些参数得到一个值。

存储过程在存储函数的基础上，增加了对表返回的支持，并且支持利用游标、对表进行增删改等操作，操作更加丰富，但是编程的复杂度也增加了。游标是我个人很喜欢的一个特性，它仿佛让你进入了数据库内部的代码，可以一行行移动进行筛选。

触发器可以将负责存储的数据库和后台服务器分开，将存储和计算放在一起。减小了服务器的压力，同时也可以保证正确性。也是数据库一个非常好的特性。

4 综合实践任务

4.1 系统设计目标

随着经济、科技和社会的发展，飞机已经成了非常常见的出行工具之一，也有许多小公司开始运营自己的航空公司，本工程需要实现一个便于操作管理且稳定性强的轻量化机票预订系统，该系统需要支持管理员和用户登录；支持航班信息和飞机信息的录入、查询、订票、支付、退票等操作。此外，系统还可以装在物联网机器上以便于用户取票。

4.2 需求分析

首先，这个机票预订系统需要支持两种不同的使用者使用，普通用户和管理员用户。对于普通用户，需要可以查询机票、预订机票、查看订单、退订机票和取票等功能；对于管理员，具有统计订单(包括预定情况，航班满座率，收入信息等)、航班信息添加(增加新的航班)、修改(遇到气候等问题等的航班调整或新飞机、航班信息录入)和删除(飞机退役)的功能。此外，由于航班添加、修改、删除的存在，需要保障用户的订单级联地受到修改和删除航班的影响。

根据上述需求，画出功能需求的示意图如图 4.1：

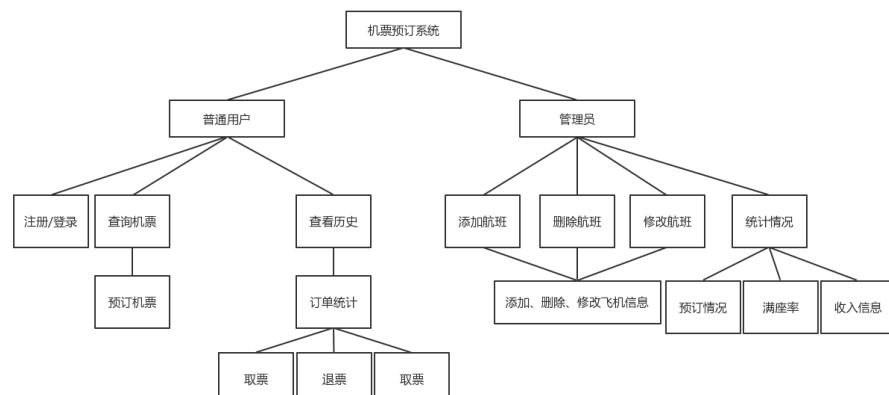


图 4.1 机票预订系统功能示意图

核心的数据流图有用户订票/退票数据流图和航班信息修改(包括添加和删除)数据流图。

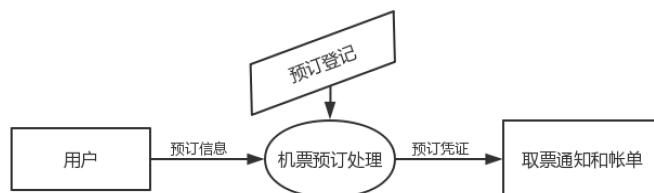


图 4.2 用户订票数据流图

如图 4.2，用户提交预订信息后，对信息进行处理，若登记成功则打印取票通知和账单。对于退票，处理的流程相似，将在详细设计中详述。

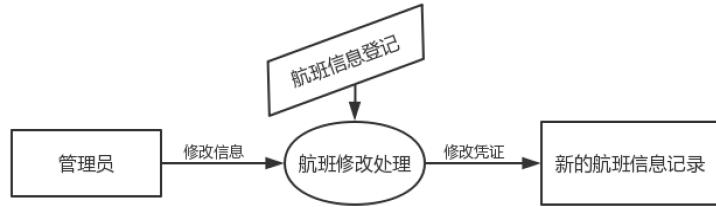


图 4.3 管理员修改航班信息数据流图

如图 4.3，管理员修改航班信息后，经过处理，若航班信息重新等级成功，则获得新的航班信息记录。

系统总体需求列表如表 4.1 所示。

表 4.1 系统总体需求

需求名称	需求描述
无错误需求	不能在订票或修改航班信息上出现错误，保持收支相等。
安全性需求	保证安全性，不能被 SQLI、XSS 等方式攻击
多线程需求	因为 B 端会有多个客户同时用，要求多线程且无误。
扩展性需求	便于后期增加新的功能。

4.3 总体设计

根据需求设计了的机票预订系统，如图 4.4。

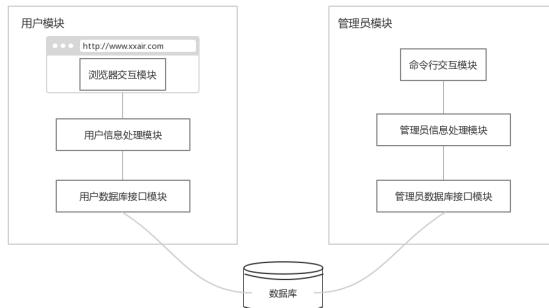


图 4.4 机票预订系统

整个系统采用了 B/S 的架构，用户操作位于浏览器；管理员操作位于服务器处，分别用用户信息处理模块和管理员信息处理模块处理操作，下层分别用用户和管理员数据库接口模块提供到数据库的接口，B/S 的核心架构示意图如图 4.5。

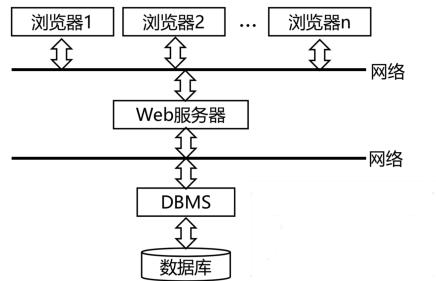


图 4.5 B/S 架构示意图

核心即浏览器通过 HTTP 向 Web 服务器发送请求，服务器通过与数据库交互返回处理的结果。

在数据库端：

1. 由于存在用户和管理员两个不同的角色，所以需要两个权限不同的用户，分别用于用户连接和管理员连接。
2. 需要根据需求和 E-R 图得到关系模型，建立数据库的各张表，完成用户定义完整性约束。
3. 创建一些触发器、视图、存储函数、事件等用于帮助后台部分对数据库信息进行调用。
4. 对于经常使用的一些属性建立索引。

在 Web 服务器(后台部分)：

1. 需要进行一些配置，包括数据库的端口、名称、用户名、密码、模式名、表名等等内容进行配置。测试连接性与执行 SQL 语句正确性。
2. 建立机制初步防止 SQLI, (SQL 注入) 攻击。
3. 定义一些公共处理的函数，比如定点数的运算的一些函数，数据库连接器等供用户和管理员调用。
4. 分别定义用户和管理员专用的一些函数接口，并加以实现，进行测试。
5. 为了保证并发下的正确性，需要通过设置事务进行控制。优化程序时还需要设置不同的事务隔离级别。
6. 利用命令行工具进行单元测试和进程测试。

在前端部分：

1. 利用 HMTL 和 PHP 还有部分 JavaScript 进行页面的组件设计、页面的跳转逻辑设计。
2. 利用 Session 或 Cookie 进行无状态连接下的状态信息保存。由于本系统中需要保存的东西不多，都在数据库中，所以只需要设置维护用户名的连接 Session 即可。
3. 对页面进行优化，如对于选择起飞地和目的地，可以通过 SQL 语句查询动态的获得已经存在的航班，节约用户操作的时间。
4. 对 XSS 等攻击进行防范，防止黑客在 GET 请求中私自添加 JavaScript 代码，导致收到攻击。
5. 对页面布局等进行优化，添加一些 CSS 组件优化页面体验。

测试部分：

1. 首先对数据库进行测试，利用 Select、Insert、Update 语句等，判断用户和管理员是否具有对应权限，是否能够正确对表进行操作。
2. 然后对数据库-后端进行单元测试和集成测试，在单元测试时判断每一个功能是否正确，提示错误是否正确。
3. 最后将前端连接到后端上，进行集成测试，测试页面逻辑，返回结果是否符合设计要求。

根据需求分析，以及以上的总体设计，本实验所使用的环境如表 4.2 所示：

表 4.2 系统环境一览表

环境表项	说明
操作系统	MacOS Mojave 10.14
系统内核	Darwin Kernel Version 18.5.0
数据库及其版本	MySQL Ver 8.0.15
数据库引擎	InnoDB
Web 服务器	Apache Web Server
编程语言前端	PHP 内嵌 HTML+JavaScript
编程语言后端	PHP 7.3
数据库连接器	PHP 7.3 mysqli
浏览器	Chrome 74.0

注：其中 Web 服务器首先使用了 PHP Storm 的服务器，后来发现它对资源支持有限，并且对于循环次数要求非常严格，所以换成了 Apache 的服务器
MySQL 的部分配置如表 4.3 所示：

表 4.3 MySQL 环境配置一览表

配置项	值	说明
secure_file_priv	/Users/yonginxu/Codes/	数据安全导入地址
bind-address	127.0.0.1	绑定的 IP 地址(本地)
port	3306	绑定的端口号
default_authentication_plugin	mysql_native_password	默认地连接方式用密码

注：在 MySQL 默认的连接方式是秘钥加密，需要改成用密码加密的方式，才能够使用 MySQLI 进行连接。

Apache 服务器的部分配置如表 4.4 所示：

表 4.4 Apache 服务器环境配置一览表

配置项	值	说明
DocumentRoot	"/Library/WebServer/Documents"	服务器文档运行地址
php7_module	libexec/apache2/libphp7.so	装载 PHP7 的动态库
ServerAdmin	127.0.0.1:80	服务启动地址
DirectoryIndex	Index.html	主页位置

4.4 数据库设计

机票预订系统的 E-R 图设计如图 4.6:

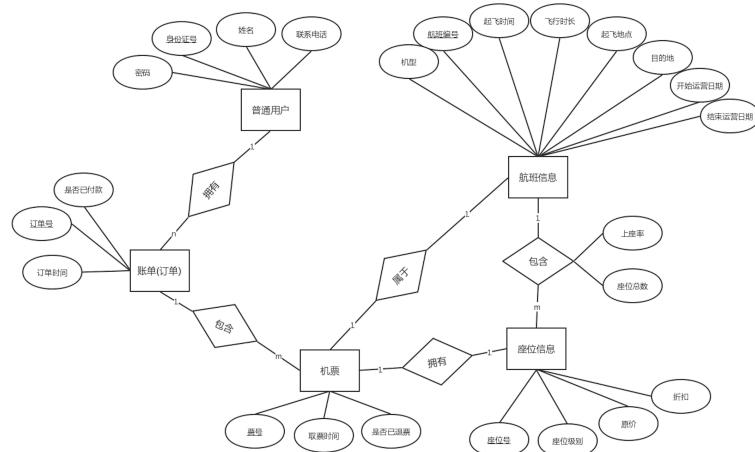


图 4.6 机票预订系统 E-R 图

根据 E-R 图，可以设计数据库中的表。本系统的核心内容包括在 5 张表中，分别是用户信息记录表(User table)、航班信息记录表(Flight table)、座位信息记录表(Seat table)、订单信息记录表(Order table)和机票信息记录表(Ticket table)。

在此基础上，本题对存储信息进行了简化。由于座位信息里包含了座位号和座位级别，若对于每天的每一次飞行都进行每个座位号的记录，则座位信息表过于庞大。若作为信息里不记录是否已经被预订，则在订票的时候判断座位是否被预订的计算复杂度太高。所以此处取消了座位信息表，综合它到航班信息表中，增加了飞行日志表，记录了每个航班的每个航程。改进后的 E-R 图如图 4.7 所示：

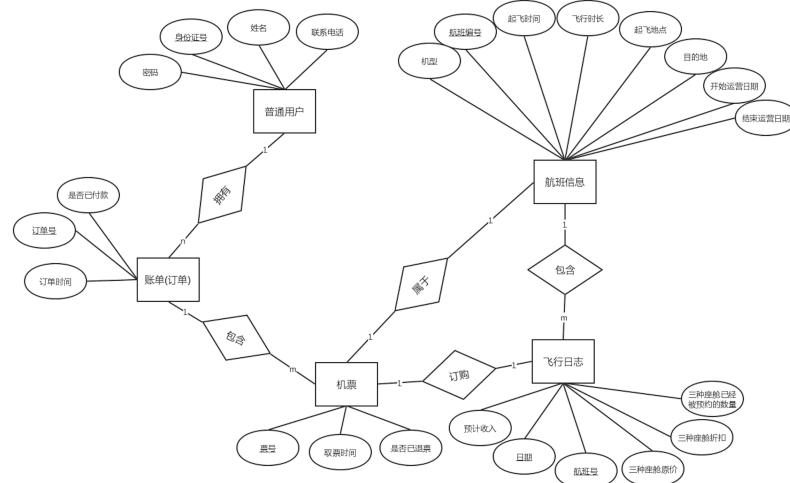


图 4.7 机票预订系统 E-R 图(改进版)

可以看到，飞行日志表(Flying Dates Table)以(日期, 航班号)为主码，记录了飞行的计划，此外，还包含了三种座舱的折扣和原价，以供修改。另有被预约数量和预计收入以便统计。

用户信息记录表主要记录可以登录此系统的用户信息，表项及其说明如下：

表 4.5 用户信息记录表的表项及其说明

表项名称	说明
U_ID	主码，用户 ID
U_PASSWORD	用户登录的密码
U_NAME	用户的姓名
U_TELEPHONE	用户的联系方式
U_BALANCE	用户余额

航班信息记录表主要记录航班的一些明细信息，表项及其说明如下：

表 4.6 航班信息记录表的表项及其说明

表项名称	说明	表项名称	说明
F_ID	主码，航班 ID	DEPART_PLACE	外码，出发地点
F_TYPE	航班类型	ARRIVE_PLACE	外码，目的地
DEPART_TIME	航班出发时间	FSEAT_NUMBER	头等舱座椅数
DURATION	航行时长	CSEAT_NUMBER	商务舱座椅数
		ESEAT_NUMBER	经济舱座椅数

飞行日志表记录了飞机的飞行时间以及值飞航班。此外还包含了当日已经被预订的座位，以及预计的营收。

表 4.7 飞行日志记录表的表项及其说明

表项名称	说明
F_date	1/2 主码, 飞行日期
F_FID	1/2 主码，外码，航班编号
F_discount	经济舱的折扣
F_cdiscount	商务舱的折扣
F_fdiscount	头等舱的折扣
e_price	经济舱票价
c_price	商务舱票价
f_price	头等舱票价

订单信息记录表记录了每一笔用户订购机票的明细内容，这里要注意，一个订单可以对应多张机票。因为存在着单程票、往返票、多程票的可能。

注：将订单是否支付与机票是否取消分开，有利于多程票的统计，也避免了既保存机票的支付与取票信息，且又保存订单支付、取票信息的冗余。

表 4.8 订单信息记录表的表项及其说明

表项名称	说明
O_ID	主码，订单 ID
O_UID	外码，订单所属的用户
O_TIME	订单创建的时间
O_COST	订单的总费用
O_PAID	订单是否已支付

机票信息是基于订单信息产生的，一个订单信息可能对应着多个机票信息记录。机票信息记录主要记录了一些用户订购机票的细节信息。

表 4.9 机票信息记录表的表项及其说明

表项名称	说明
T_ID	主码，机票 ID
T_OID	外码，机票所属的订单 ID
T_CANCELED	机票是否已经被取消
T_TOKEOFFTIME	飞机的起飞时间
T_GOTTIME	机票被取走的时间
T_FID	外码，机票所对应的航班
T_SEATCLASS	机票的座位档次
T_PRICE	购买时的价格

另外，系统还设计了一系列的辅助表格。辅助表 1 用于快速查看地址和其对应的机场码(3 位)的对应。

表 4.10 机场码信息表表项及其说明

表项名称	说明
AP_CODE	主码，机场码，如(PVG)
AP_NAME	机场名称，如上海浦东机场
AP_CITY	机场所在城市，如上海

为了方便管理员对每个航班每次飞行的上座率、营收进行统计，也为了方便用户订票时判断余票时更加方便，设计了根据飞行日期和航班编号得到的三种级别的座位总数，每个级别在当天已经预订的座位数，以及预计营收进行了视图创建，视图的 SQL 语句如下：

```
drop view if exists v1;
create view v1 as
select Flying_date.f_date, Flying_date.f_FID, e_taken, c_taken, f_taken,
revenue,
FSEAT_NUMBER, ESEAT_NUMBER, CSEAT_NUMBER
from Flying_date join Flight_t on Flying_date.f_FID = Flight_t.F_ID;
```

视图的模拟展示结果如图 4.8 所示：

```
mysql> select * from v1;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| f_date | f_FID | e_taken | c_taken | f_taken | revenue | FSEAT_NUMBER | ESEAT_NUMBER | CSEAT_NUMBER |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2019-05-02 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-03 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-04 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-05 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-06 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-07 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-08 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-09 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-10 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
| 2019-05-11 | 350 | 0 | 0 | 0 | 0.00 | 10 | 30 | 260 |
```

图 4.8 视图模拟展示结果图

由于预订一张票需要在飞行日志中添加相应一种级别的座椅占用量，同时需要按照价格增加预计营收，而退票需要减少相应一种级别的座椅占用量，同时需要按照价格减少预计营收，所以此处设计了一个触发器，当增加了一个座椅占用时，对应计算价格增加营收；而当减少了一个座椅占用时，对应计算价格减少营收。对应 SQL 语句如下：

```
delimiter $$  
drop trigger if exists trg1 $$  
create trigger trg1 before update  
on Flying_date for each row  
begin  
    if new.e_taken = old.e_taken + 1 then  
        set new.revenue = old.revenue + old.f_ediscount / 100 * old.e_price;  
    elseif new.e_taken = old.e_taken - 1 then  
        set new.revenue = old.revenue - old.f_ediscount / 100 * old.e_price;  
    elseif new.c_taken = old.c_taken + 1 then  
        set new.revenue = old.revenue + old.f_cdiscount / 100 * old.c_price;  
    elseif new.c_taken = old.c_taken - 1 then  
        set new.revenue = old.revenue - old.f_cdiscount / 100 * old.c_price;  
    elseif new.f_taken = old.f_taken + 1 then  
        set new.revenue = old.revenue + old.f_fdiscount / 100 * old.f_price;  
    elseif new.f_taken = old.f_taken - 1 then  
        set new.revenue = old.revenue - old.f_fdiscount / 100 * old.f_price;  
    end if;  
end$$  
delimiter ;
```

另外，为了防止有人很早订票，但是迟迟不付款，让别人无法在订票的情况下发生，设计了一些事件，在每天凌晨 1 点自动清理掉前一天没有付款的订单。这利用了数据库事件的方法。分别把三种级别的座椅的未付款订单清理，并将用户的订单设置为已取消状态，总共设置了 4 个事件，分别对应把头等舱、商务舱和经济舱的订单清理，还有一个事件用于将用户订单取消。

以头等舱未付款订单为例，对应的 SQL 语句如下：

```
Drop EVENT IF EXISTS temp_event;  
delimiter $$  
CREATE EVENT IF NOT EXISTS temp_event  
ON SCHEDULE EVERY 1 DAY STARTS DATE_ADD(CURDATE(), INTERVAL 1 DAY), INTERVAL 1 HOUR  
ON COMPLETION PRESERVE ENABLE  
DO update Flying_date set e_taken = e_taken - 1 where (f_FID, f_date) in (select  
T_FID,date(T_TOKEOFFTIME) from Order_t join Ticket_t on ticket_t.T_OID = Order_t.O_ID where  
O_PAID = false and T_CANCELED = false and T_SEATCLASS = 'C');  
$$  
delimiter ;
```

剩余 3 个事件原理相似，此处不再展示。

4.5 详细设计与实现

4.5.1 后端公共类设计与实现

数据库端的实现部分基本已经在 4.4 部分阐释完成。本部分将直接从后端设计部分进行说明，到最后前端的 UI 和逻辑设计。

后端首先需要定义一些公共需要用到的函数，此处的公共指的是用户和管理员都需要的一些函数，本文设计与实现的类如表 4.11 所列出：

表 4.11 公共类包含内容说明表

类名称	包含内容	说明
Config	表名称，属性名称，视图名称	用于增删改查时调用
DBConnector	MySQL 连接创建于基本查询语句	提供连接器
Decimal2P	2 位顶点小数的加减乘计算	处理 2 位定点小数的类
Flight	航班信息的构造函数和 <code>toString</code>	方便创建航班信息

根据表 4.11 所列出的 4 个类，将逐一实现这些函数。

首先是 Config 配置文件类，这个类相当于是一个命名空间，在这个命名空间下存储了所有数据库表的表名成，每个表里每个属性的名称，创建了的视图的名称，以及数据库服务器的 IP 地址，端口号，用户的名称和密码。

具体实现时首先创建一个命名空间：

```
namespace config;
```

然后对在这个命名空间内创建多个类，这些类仅仅用于定义常量供使用者调用。这些类的信息如表 4.12 中所示：

表 4.12 Config 命名空间包含类说明表

类名称	包含内容	说明
DB_info	数据库 IP 端口用户等	数据库基本信息类
Code_City	机场代码表名称，属性名称	机场代码表信息类
User_table	用户表名称，属性名称	用户表信息类
.....	数据库中其他表名称，属性名称	其他表的信息类
Flying_date	飞行日志表名称，属性名称	飞行日志表信息类
View	视图名称	数据库视图信息类

以数据库信息和机场代码表类为例，实现的基本模式如下：

```
final class DB_info { // Basic information about Database
    public const SERVER_ADDRESS = "127.0.0.1";           // Server address of the database
    public const DATABASE_ADMIN_NAME = "seller";           // Admin user use this user name to login in to the DB
    public const DATABASE_ADMIN_PSW = "123456";           // Password of the admin user
    public const DATABASE_NAME = "ticket";                 // The name of the schema
    public const DATABASE_COSTUMER_NAME = "customer";     // Customer use this user name to login in to the DB
    public const DATABASE_COSTUMER_PSW = "123456";         // Password of customers
}

final class Code_CITY { // Code to Address and City
    public const NAME = "code_addr";                      // Name of the table
    public const CODE = "AP_CODE";                         // primary key, @datatype: char(3)
    public const AP_NAME = "AP_NAME";                     // @datatype: char(50)
    public const CITY = "AP_CITY";                        // @datatype: char(20)
}
```

下面对于数据库连接器进行设计与实现，具体实现的成员函数如表 4.13 所示：

表 4.13 数据库连接器类成员函数说明表

成员函数名称	接收参数	实现内容
构造函数	是否管理员	根据配置文件连接并显示状态
析构函数	无	关闭连接
根据城市获得机场码	城市的名称	返回城市拥有的所有机场代码
.....	其他一些基本函数连接
获得整张表	表名称	打印整张表用于 debug

其中，构造函数通过传入参数是否管理员，选择 Config 命名空间中的数据库用户，尝试对数据库进行连接，如果连接成功，则显示成功，否则出现了配置错误或者服务器未打开的情况，需要进行问题检查，利用 `mysqli_connect_error()` 进行错误显示。流程如图 4.9 所示：

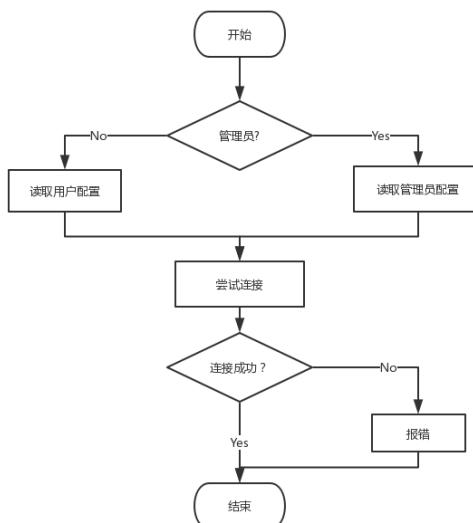


图 4.9 数据库连接器构造函数流程图

析构函数仅仅直接关闭连接，由于 PHP 内存安全，不需要主动回收内存。

获得整张表和根据城市获得机场代码主要由 PHP 向 MySQL 发送数据库查询语句请求，MySQL 执行请求并返回结果而完成。在 PHP 使用 MySQLi 且连接成功的情况下，一个普通的 SQL 语句执行的伪代码如下：

```

$test_query = "show tables";           // 首先创建一条查询语句，这里以 show tables 为例
$result = $this->link->query($test_query); // 利用连接的 query 执行这条查询语句，保存在 result
while(list($table_name) = $result->fetch_row()) // fetch_row 获得每一行，list() 进行模式匹配
    printf("%s <br />", $table_name);      // 一行一行输出即可(这里只匹配了一个表名)
$result->free();                      // 释放空间
  
```

以查看所有表为例，首先创建一个查询语句，然后利用连接器(这里起名为 link)进行 query 操作，利用模式匹配将结果一行行取出，最后释放空间即可。

DBConnector 中的基本函数可以进行单元测试，也可以被用户和管理员调用。

Decimal2P 类用来处理精度为 2 的定点小数，由于浮点数是用 IEEE754 的方式进行存储的，它不是精确的，可以用非常简单的一个方法进行测试，在 PHP 中输入

```
echo (0.58 * 100)
```

可以得到结果是 57，并不是 58，在金钱交易上，绝对不能容忍的就是交易金额双方不对等。所以需要进行专门的定点小数处理函数。由于 PHP 中并不自带定点小数的处理方法，此处自己设定了处理精度为 2 的定点小数的类，它包含的成员函数如表 4.14 所示：

表 4.14 Decimal2P 类成员函数说明表

成员函数名称	接收参数	实现内容
构造函数	金额的字符串	存小数部分和整数部分,格式错误(不是两位小数)则报错
Compare	第二个金额的字符串或 Decimal2P 类	比较两个金额的大小
Plus	同上	两个金额相加
Mius	同上	两个金额相减
Multiply	同上	两个金额相乘
Multiply_discount	折扣整数	乘折扣得到金额
__toString	无	金额转字符串

在处理两位精度的定点小数时，Decimal2P 时刻保存着整数部分和小数部分，对于加法时，首先将小数部分乘 100 加整数部分，算出十进制左移两位的两个加数，相加后将结果右移两位即可。减法的原理与加法相同。

乘法执行的流程图如图 4.10，与加减法不同的是需要记得结果除以 100，因为两个乘数相乘之前乘的 100 被会被扩大为 10000 倍。

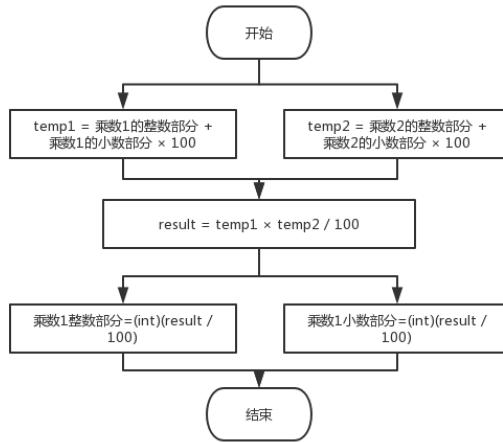


图 4.10 两位定点小数乘法流程图

此外，`__toString()`主要用于输出，打印金额时只需要拼接字符串，在整数部分和小数部分之间加上小数点，返回组合之后的字符串即可。

由于 Flight 类比较简单，仅仅是提供了构造函数和 `toString`，再次不专门介绍。

4.5.2 后端用户操作接口设计与实现

后端部分需要给用户和管理员所有的 UI 操作提供接口，根据自底向上的原则，先设计后端有利于单元测试，在确保后端正正确的情况下，接入前端进行集成测试会变得更加可靠。

对于用户操作接口，后端设计并实现了如表 4.15 的成员函数：

表 4.15 后端用户操作类成员函数说明表

成员函数名称	接收参数	实现内容
Create_Account	用户名、密码等	按照填写信息创建账户
Login_Account	用户输入的账号和密码	验证账户是否登录成功
Search_Tickets	出发地和目的地城市	按照用户输入的出发地和目的地查询航班
Order_Tickets	航班日期、座位级别等	按照用户选择进行订票
Pay_for_Orders	订单号	根据用户选择的订单号进行付款操作
Take_Ticket	票号	根据用户选择的票号进行取票操作
Cancel_Ticket	票号	根据用户的选进行取消票的操作
Lookup_History	无	查询用户历史订单
Add_Balance	金额	根据用户输入增加余额

其中，Create_Account 和 Login_Account 函数是静态函数，可以允许用户在不创建用户操作对象时直接实例化调用。而剩余的其他函数是普通成员函数，必须要在用户创建了自己的操作对象之后才能进行调用。

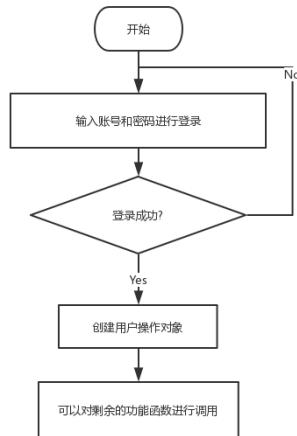


图 4.11 后端函数功能调用逻辑

如图 4.11，在用户成功登录之后，系统会为用户生成属于他的操作对象，通过这个操作对象，用户可以进行查询历史，查询航班，订购机票等剩余的操作。在实际前端进行运行的时候，需要通过 Session 或 Cookie 进行状态保存，用来标识当前用户的操作对象。具体将在 4.5.4 和 4.5.5 部分进行详细阐述，当前只关注后端的逻辑。

由于前端返回的信息可能不合法；或者操作数据库时可能执行发生了错误；亦或是服务器过于繁忙，不能返回错误。所以本文设计了如表 4.16 的一些异常。

表 4.16 后端用户操作类异常说明表

异常编号	异常名称	实现内容
0	InvalidInput	输入格式不正确等
1	NameTooLong	姓名过长
2	PswTooLong	密码过长
3	TelTooLong	电话号码过长
4	InsertAcconutFailed	创建用户失败，服务器忙等
5	AccountNotExist	账户不存在
6	IDInvalidFormat	账户格式不正确
7	SrcPlaceNotExist	起飞地不存在
8	DstPlaceNotExist	降落地点不存在
9	NoTargetFlight	不存在这样的航班
10	ServerBusy	服务器忙
11	TooLatetoDo	操作过时
12	AlreadyCanceled	票已经被取消了
13	AlreadyPaid	票已经付款
14	AlreadyGot	票已经被取过了
15	NotEnoughBalance	余额不足
16	CouldNotFindOrder	找不到这个订单
17	HaventPaid	还没有付款
18	TooEarly	过早执行操作
19	SeatsSoldOut	无余票
20	CouldNotFindTicket	无法找到这个票

本文的基本的查询(Select)语句的执行模式在 4.5.1 中已经进行了阐述，这里还要说明对于增(Insert)、删(Delete)和改(Update)，本文的执行模式，如图 4.12：

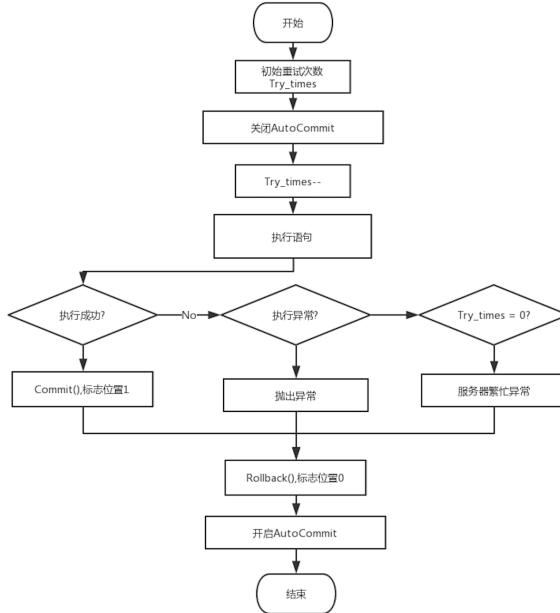


图 4.12 增删改一般模式

首先要关闭 Auto Commit，然后执行查询语句，如果执行成功则 Commit，否则 Rollback，执行异常时需要抛出相应的异常。此外，设定一个重试的次数，如果超过次数则服务器繁忙，防止过度占用资源。

下面，以情况较为复杂的创建账户和预订机票为例，进行详细说明。

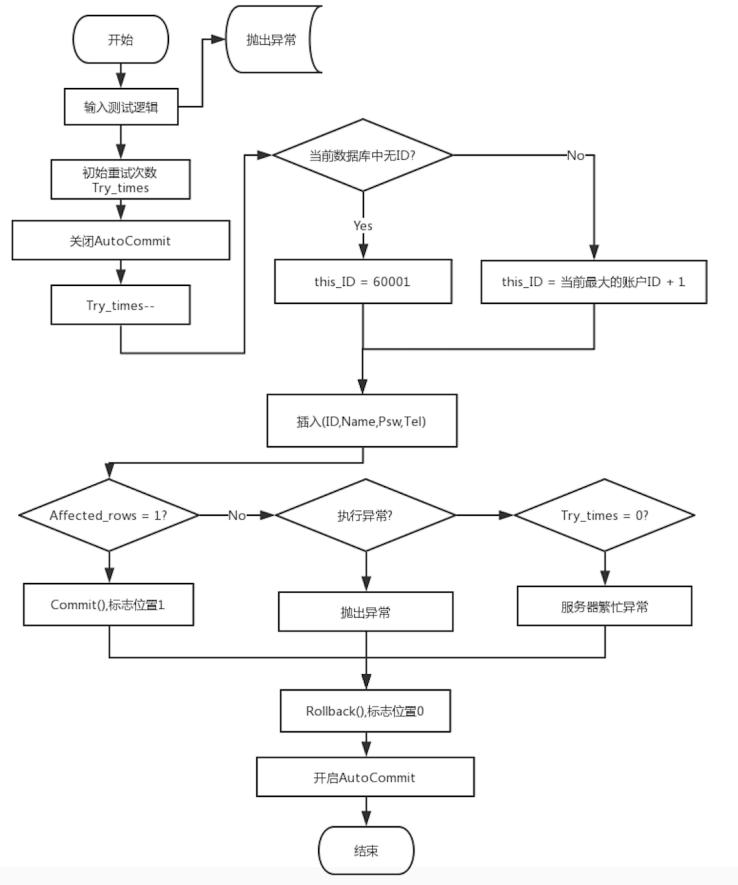


图 4.13 插入账户逻辑

如图 4.13，插入账户首先需要测试用户的输入是否合理，比如姓名是否过长，密码是否符合要求等，若失败了需要引起相应的异常。否则获得数据库中最大的 ID，若没有则当前 ID=60001，否则当前 ID=最大 ID+1，然后进行数据库插入，插入结果判断遵循图 4.12 的数据库插入一般模式，如果成功则 Commit，失败则回滚，超过次数则抛出服务器繁忙异常。之后的登录必须要让用户用这个 ID 登录，因为存在人可能会重名的可能，所以 ID 作为主码成为用户登录的标识。

预订机票则更加复杂，预订机票需要经过的大致流程如图 4.14 所示。

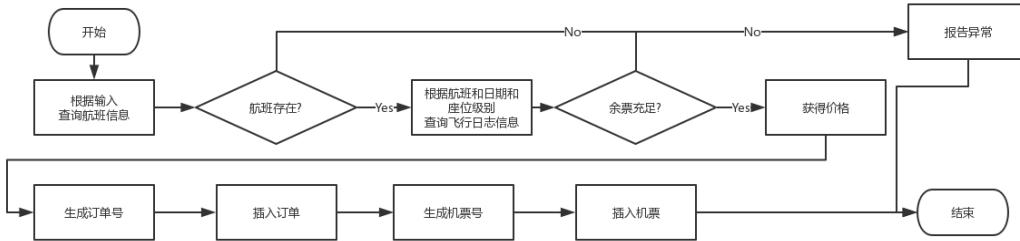


图 4.14 忽略并发异常等的预订机票流程图(结合图 4.12 的事务模式图后成为完整的流程图)

需要根据输入信息判断航班是否存在，再根据航班的日期和作为级别查询余票是否充足，都成功了则获得需要的价格，订单号和机票号，插入对应信息即可。

图 4.14 没有考虑并发异常等具体事项，仅仅是一个大致的流程图，在实际操作时，需要将它与 4.12 的一般增删改的流程进行结合。其中还需要注意的是，由于前面的多数步骤都是查询，没有进行插入操作，所以可以设置不同的隔离级别。

值得注意的是，在插入的时候需要使用 `Serializable` 的串行隔离级别，因为余票数量的信息是通过计算算出来的，在数据库本身中并没有涉及 `Check` 语句，(并且当前 MySQL 版本也不支持 `Check` 语句)，所以可能会出现幻读的情况，如果算出来还有一张余票，但是别人在你插入之前已经预定了这张机票，你无法感知，所以必须要用 `Serializable` 的隔离级别。如图 4.15。

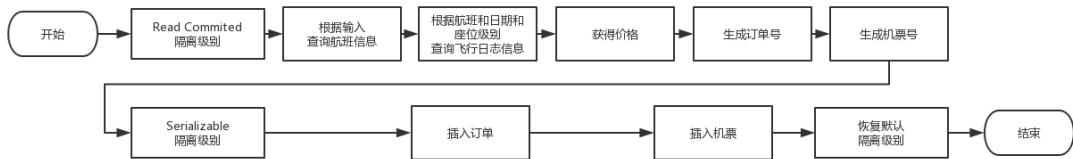


图 4.15 隔离级别控制图

接着为剩余函数进行简单阐述。

对于 `Login` 用户登录成员函数，只需要运用查找语句，查找出当前用户的账户和密码，并进行匹配，可能出现的问题是账户不存在，密码不匹配等。如果密码匹配，那么则创建一个新的用户操作对象，登陆成功，供当前用户使用。

对于 `Search_Tickets` 也是纯查找成员函数，只需要根据用户的输入，查找当前符合条件的机票即可，为了用户的更好体验，还需要查询剩余的票数。但一般剩余的票数都是不展示的，一般只会展示是否有余票，还有价格，供用户挑选并预定，对应的流程图如 4.16 所示。

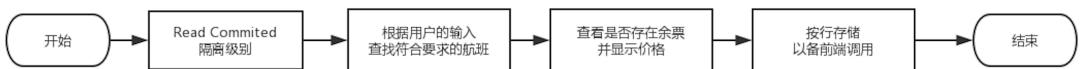


图 4.16 查询机票的一般流程

与 `Search_Tickets` 非常类似，`Lookup_History` 用于查找用户历史的订单，`get_Flying_time` 用于获得飞机起飞的具体实现，这些函数都是纯查找函数，用 `Read Committed` 的隔离级别，进行查询，并保存查询结果即可。再次不再说明具体实现。

`Add_Balance` 为充值余额函数，在 UI 中提供给用户多种选择，也就是说用户必须充值相应的额度，不能自定义具体的额度，比如说提供 1000, 2000, 5000, 10000 的充值额度。这样使得实现起来非常简单，首先得到用户现在的余额，再利用 `Decimal2P` 中的加法进行处理即可。需要说明的是，这种情况需要解决不可重复读的问题，所以隔离级别为 `Read Repeatable`。

为预定的票付款、取票、取消票都是比较复杂的操作，下面逐一说明。

图 4.17 为大致的为订单付款的流程,结合图 4.17 和图 4.12 一般增删改模式(事务模式)可以得到完整的流程图,由于过于复杂,此处只分开来说明,不进行合并。

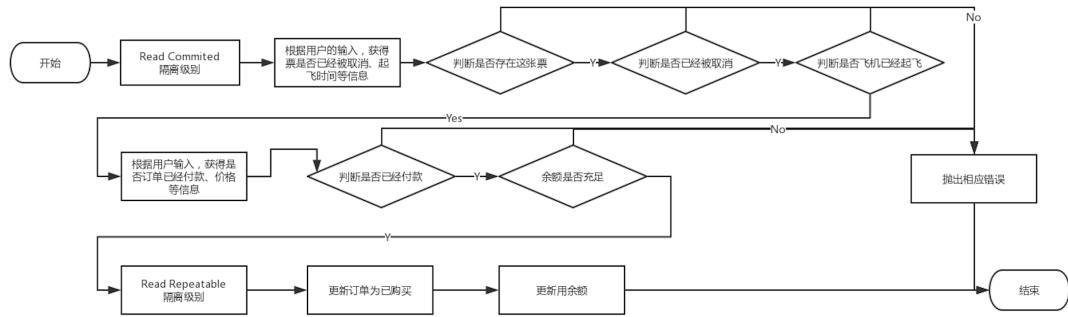


图 4.17 为订单付款的一般流程(结合图 4.12 的事务模式图后成为完整的流程图)

如图 4.17 所示,一开始只是查询语句,将语句设置成 Read Committed 的隔离级别,然后根据用户的输入,获得票是否已经被取消,起飞时间等等信息。这里需要判断这张票是否存在,判断是否已经取消,飞机是否已经起飞,在条件不满足的情况下,不能付款。然后得到订单的信息,判断是否已经付款,以及用户的余额是否充足。如果条件都满足,则需要设置为 Read Repeatable 的隔离级别,更新订单状态为已购买并且更新用户余额。

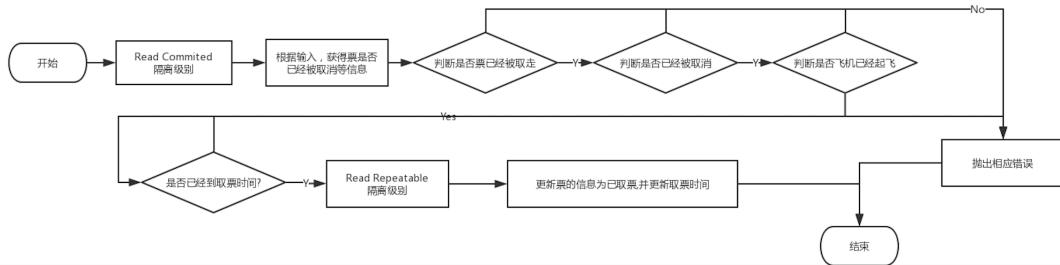


图 4.18 取票的一般流程(结合图 4.12 的事务模式图后成为完整的流程图)

如图 4.18,在取票时,需要考虑的是是否已经付款、票是否已经被取走、票是否已经被取消、飞机是否已经起飞,有没有到达取票的时间。如果这些查询的条件都满足,则更换为 Read Repe 的隔离集,更新表的信息为已取票,并且更新取票时间,否则抛出相应错误。

与付款和取票相比,取消票需要考虑的更多一些。首先需要获得票的信息,判断是否票已经被取了,是否飞机已经起飞了,是否票之前已经被取消了,然后需要根据购票时的座位等级更新对应的座位余座数量。最重要的一点时,需要判断用户是否已经付过钱,如果用户已经付了钱,那么用户需要得到退款,如果用户没有付过钱,则直接进行座位预订数量的减少即可。与订票相同的是,取消票也需要用到 Serializable 来防止幻读的问题。

由于大致流程图与 4.17 和 4.18 非常相似,在此不再针对取消票专门回执流程图。

4.5.3 后端管理员操作接口设计与实现

对于管理员操作接口，后端设计并实现了如表 4.17 的成员函数：

表 4.17 后端管理员操作类成员函数说明表

成员函数名称	接收参数	实现内容
<code>__construct</code>	无	创建管理员专用链接
<code>Add_Flight</code>	航班基本信息	管理员录入航班基本信息
<code>Add_Flying_Date</code>	航班编号和飞行日期	添加飞行计划
<code>List_Data</code>	无	日常上座率等统计
<code>Show_Revenue_By_Month</code>	无	按照月份进行收入统计
<code>Cancel_date</code>	航班编号和飞行日期	取消某天的飞行计划

管理员的操作没有用户的操作那么丰富，但涵盖了一些日常需要的操作，其中 `Add_Flight`, `Add_Flying_Date` 和 `Cancel_Date` 都是逻辑比较复杂的成员函数。

由于前端返回的信息可能不合法；或者操作数据库时可能执行发生了错误；亦或是服务器过于繁忙，不能返回错误。所以本文设计了如表 4.18 的管理员进行操作时可能引起的异常。

表 4.18 后端管理员操作类异常说明表

异常编号	异常名称	实现内容
0	<code>UNKNOWN</code>	不确定，不常用
1	<code>FIDNotNumeric</code>	格式不符合，不是数字
2	<code>PlaceNotValid</code>	不存在这个机场
3	<code>SeatsNotNumeric</code>	座位不是数字
4	<code>FIDAlreadyExist</code>	已经存在这个航班
5	<code>InsertFlightFailed</code>	插入航班失败，服务器忙等
6	<code>InvalidSeatsParam</code>	参数设置错误
7	<code>AddFlightDateFailed</code>	飞行日志添加失败，服务器忙等
8	<code>TimeLogicError</code>	时间的逻辑错误
9	<code>DiscountNotNumeric</code>	折扣格式错误
10	<code>NoSuchFlight</code>	没有这趟航班
11	<code>ServerBusy</code>	服务器忙
20	<code>CouldNotFindTicket</code>	无法找到这个票

构造函数做的工作是给管理员维持一个向 MySQL 的连接。此外，需要得到机场对应的号码和城市的表格，即 `Code_City` 表格的内容。由于逻辑简单，此处不详细说明。

首先解释 `List_Data` 和 `Show_Revenue_By_Month` 两个统计性的功能，因为它们只需要用到 `Select` 语句。

对于 `List_Data` 语句，只需要通过调用之前数据库创建的视图，得到每一趟航班在每一个飞行日的时间、航班编号、三种级别座椅被订购的数量，三种级别座椅的总数量，以及收入即可。将结果存储在一个数组中，可以由上层通过解析这个多维数组，得到数据并显示。由于 `List_Data` 这种操作实时性不强，可以直接把事务设置为 `Read Uncommitted` 的级别。

对于 Show_Revenue_By_Month，用到了 Hash 散列表的存储方法，解析由 List_Data 获得的数组数据，获得(Year+Month, Revenue)的组合，如果已经存在 Key = Year + Month，则 Revenue = Revenue + this_Revenue，否则插入一个新的键值对，伪代码如下：

```
$month = date("Y-m", strtotime($f_date));
if (array_key_exists($month, $month_record)) { // 已经存在这样的键值对
    $month_record[$month] += $revenue;           // 更新新的营收额
}
else { // 不存在这样的键值对
    $month_record[$month] = $revenue;           // 插入新的键值对
}
```

下面着重描述一下三个与航班和飞行日志有关的函数。

首先是 Add_Flight 函数添加一个航班信息：

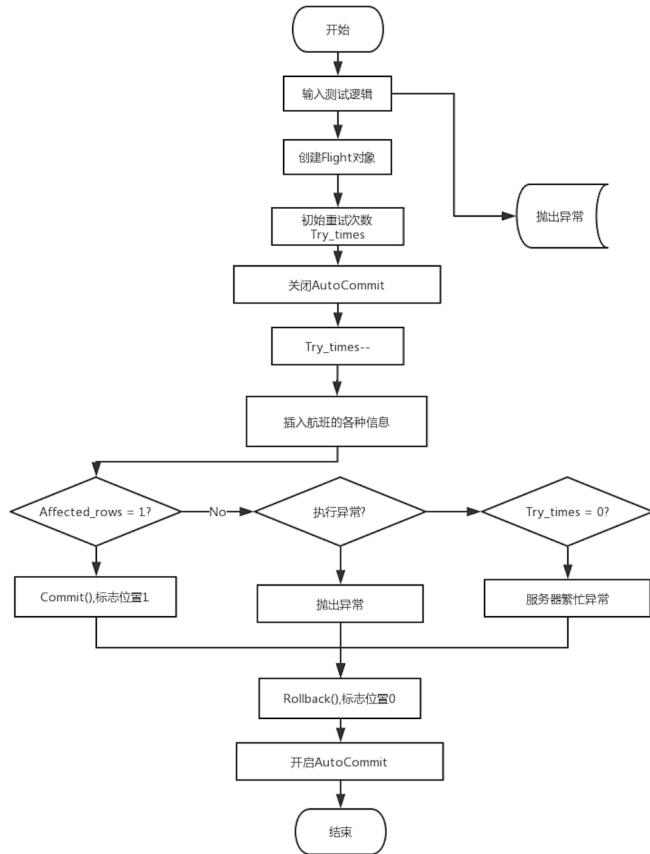


图 4.19 添加新航班的流程图

添加一个新航班首先需要进行输入逻辑的测试，包括测试航班的 ID 是不是纯数字，航班的出发地和目的地是否在给定的 Code_City 表中。(因为 Code_City 表早就已经在构造函数的时候进行了保存，所以是在内存里的，不需要再次进行查询，非常方便)，然后判断座位的数量是不是数字，主要都是一些格式上的错误判断。如果格式正确，那么初始化一个 Flight 对象，方便后续进行 toString 操作。如果上述操作都成功，那么可以正常插入航班信息，经过几次重试，若成功则 Commit，否则 Rollback，超过重试次数也 Rollback。

添加飞行日期的具体流程图如图 4.20 所示：

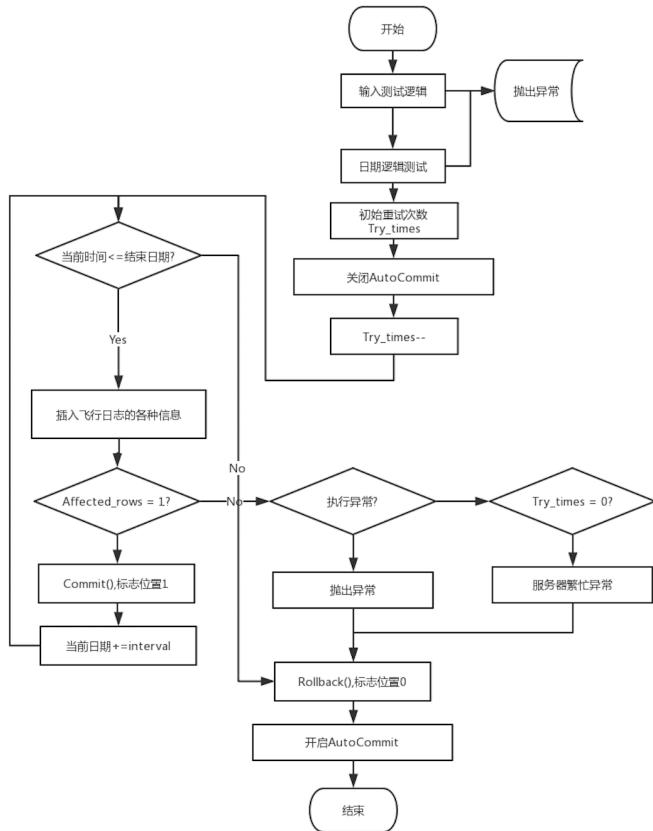


图 4.19 添加新飞行日期的流程图

如图 4.19，首先需要判断输入的正确性，包括是否航班不存在，是否日期是符合逻辑的(新添加的飞行日期是否在飞机的服役时间内)，输入是否具有格式不正确的情况等等。都判断成功以后进入循环，从当前时间等于开始时间，每次循环增加间隔时间，一直到结束时间，开始插入新的飞行日志信息，如果插入失败需要进行回滚并且需要重试。全部都成功了之后才 Commit。隔离集可以设计为 Read Committed。

删除飞行时间的逻辑较为复杂，这里只列出粗略的过程图。

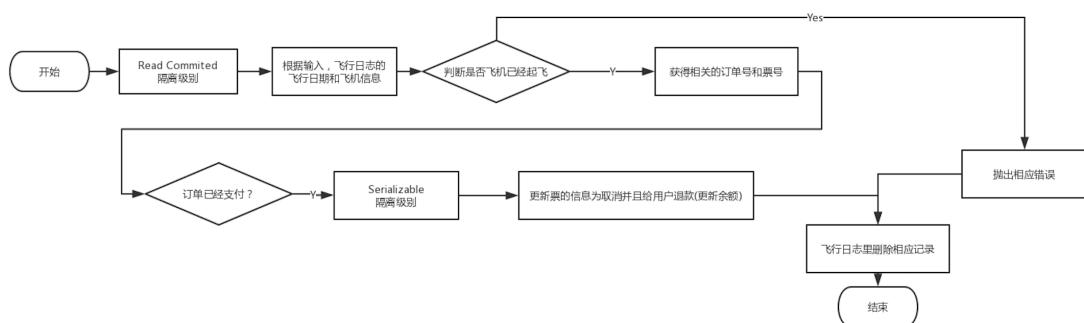


图 4.20 删除飞行日期的流程图

如图 4.20，与其他不同的是，需要考虑用户是否已经付款，如果已经付款，需要进行退款操作。其余的与上述过程相似，不再赘述。

4.5.4 用户前端逻辑设计与实现

对于用户前端的逻辑设计，总共设计了如表 4.19 的这些模块：

表 4.19 前端用户模块说明表

模块名称	模块功能
Login	登录界面
Signup	注册界面
User_main	用户主界面模块
Order_ticket	订票模块
Add_balance	充值模块
Lookup_history	历史订单模块

其中登录界面 **Login** 模块可以给用户提供输入用户名或者账户，然后可以点击登录进行登录，对于没有账户的用户，可以点击注册进行注册账户，注册完账户后可以用已注册的账户进行登录，注意账户是系统设定的，所以需要提醒用户记住账户。登录界面的示意图如图 4.21 所示：

The diagram shows a login interface with a red border. Inside, there is a title '欢迎登陆机票预订系统'. Below it are two input fields: '账户:' followed by a text input box, and '密码:' followed by another text input box. At the bottom are two buttons: '注册' (Registration) and '登录' (Login).

图 4.21 登录界面示意图

用户登录完成以后进入了 **User_main** 的用户主界面模块，在主界面，用户可以通过输入出发地和目的地，并点击查票搜索满足条件的航班，此外用户还可以选择充值或者查看历史订单。主函数的界面示意图如图 4.22 所示：

The diagram shows a main function interface with a red border. Inside, there is a title '机票预订功能块'. Below it are three dropdown menus: '出发城市:' (Departure City) set to '广州', '目的城市:' (Destination City) set to '广州', and '出发时间:' (Departure Date) with a placeholder '年 /月/日'. At the bottom are three buttons: '查票' (Search), '充值' (Recharge), and '历史' (History).

图 4.22 用户主函数界面示意图

查票之后就可以显示出查到的票的结果，用户可以通过选择完成订票的逻辑，也可以提供返回的逻辑，重新查询别的机票，通过重新输入出发城市、目的地城市和时间进行重新查询。

充值模块提供给用户多种选择，选择充值的金额，由于仅仅是模拟，此处没有更多的验证操作，真实环境下可以接入银行卡，网银支付等等。充值界面如图 4.23 所示：



充值界面示意图，显示了一个红色边框的对话框。对话框内有“欢迎登陆充值系统”字样，下方有“充值金额”选项，包括“1000.00”，“2000.00”，“5000.00”，“10000.00”以及一个“充值”按钮。

图 4.22 充值界面示意图

最后，历史订单的界面展示了用户订票的历史，并提供给用户付款、退票、取票的功能。需要用户选择对哪一张票或订单进行操作，示意图如图 4.23。



历史订单界面示意图，显示了一个对话框。对话框内有“付款/取票/退票模块”字样，下方有“票/单号”输入框，以及“选 项: ○付款 ○取票 ○退票”三个单选按钮。下方是一个表格，包含“用户编号”、“用户名称”、“用户电话”、“用户余额”四列，表格中有一条记录：60002, Tom Jones, 18013257890, 5491.65。表格上方有“返回”和“确定”两个按钮。

图 4.23 历史订单界面示意图

4.5.5 管理员前端逻辑设计与实现

对于管理员前端的逻辑设计，总共设计了如表 4.20 的这些模块：

表 4.20 前端管理员模块说明表

模块名称	模块功能
home	主界面，选择功能界面
show_revenue	营收额统计界面
add_flight	添加航班模块
add_flying_time	添加飞行计划模块
statistic	统计模块
Cancel_date	取消飞行计划模块

管理员的逻辑比较简洁清晰，主界面 Home 可以选择生下来的 5 个功能，show_revenue 可以按照月份显示营收额的统计，add_flight 可以添加航班，add_flying_time 可以添加飞行计划日志，cancel_date 对应着取消飞行计划，而 statistic 模块主要可以显示更多信息。模块间可以彼此跳转。

4.6 系统优化

系统优化本文主要从以下四个角度进行考虑。

- 1、优化查询语句；
- 2、重试次数；
- 3、不同的隔离级别；
- 4、存储与计算同步在数据库中进行。

对于第一点，本文遵守了书上提出的优化查询语句的五大规则：选择语句尽早执行；将选择、投影运算同时执行；把选择运算与其前面的笛卡尔积运算结合起来一起计算；把投影运算与其前后的双目运算结合起来一起运算；把公共子表达式的运算结果存放在外存，作中间结果，使用时读入主存。此外，本文中还侧重的点是调整多个查询语句的顺序，让查询语句尽可能的靠近，增删改语句尽可能的靠近，因为可能会切换它们的隔离级别，尽量让隔离级别少切换。

对于第二点，为了保证在并发情况下，函数执行的成功率尽可能高，所以使用了重试次数，一个函数的一些 SQL 语句可能会重复执行多次直到成功，但是也不能执行过多次，因为这样会持久占用带宽，导致整个系统的并发能力下降，所以本文使用了 3-5 次的重试次数，对于频繁使用的语句用了 3 次重试次数，不频繁的 5 次。

对于第三点，本文在不会导致出现错误的情况下尽可能的使用了最低符合要求的隔离级别。比如说在管理员统计信息时，由于实时性不强，所以用了 Read Uncommitted 的隔离级别，在纯查询语句处，尽可能的使用了 Read Committed 的隔离级别，在一般的增删改操作处使用了默认的 Read Repeatable 的隔离级别。在可能出现幻读问题导致错误的比如订票处，采用了 Serializable 的隔离级别。

对于第四点主要是利用了触发器，使得压力不完全在后台，还分担了一部分计算任务给了数据库服务器。

4.6 系统安全性保证

对于系统的安全性保证，本文主要从以下两个角度进行考虑。

- 1、防止基本的 SQL 注入；

SQL 注入主要在于用户输入一些奇怪的内容，导致查找出现问题。比如说用户账户名为”6 or 1 = 1”，如果直接传送到数据库中，那么 or 1=1 就会变成数据库中的一个有用的语句，那么查询结果都是成功的，会把所有账户和密码都返回。为了防止这样问题的出现，本文中用了很多异常处理，输入逻辑判断，将 SQL 注入的可能性降低。

- 2、防止 XSS 攻击。

`$_SERVER["PHP_SELF"]` 是超级全局变量，返回当前正在执行脚本的文件名，与 `document root` 相关，`$_SERVER["PHP_SELF"]` 会发送表单数据到当前页面，而不是跳转到不同的页面。

但是，当黑客使用跨网站脚本的 HTTP 链接来攻击时，
\$_SERVER["PHP_SELF"]服务器变量也会被植入脚本。原因就是跨网站脚本是附
在执行文件的路径后面的，因此\$_SERVER["PHP_SELF"]的字符串就会包含
HTTP 链接后面的 JavaScript 程序代码。

XSS 又叫 CSS (Cross-Site Script), 跨站脚本攻击。恶意攻击者往 Web 页面里
插入恶意 html 代码，当用户浏览该页之时，嵌入其中 Web 里面的 html 代码会被
执行，从而达到恶意用户的特殊目的

首先我们对用户所有提交的数据都通过 PHP 的 htmlspecialchars() 函数处理。
这样 JavaScript 脚本就不会执行。

4.7 系统测试

由于集成测试中已经包含了对数据库进行单元测试，以及数据库-后台进行集
成测试，所以此次的系统测试直接通过对整个系统进行测试。

首先对用户的功能进行测试，注册一个新的用户。



The image shows a registration form titled "欢迎登陆机票预订系统". It contains three text input fields: "姓 名:" with value "Tester1", "密 码:" with value "123456", and "电 话:" with value "18051806789". Below the inputs is a blue "OK" button.

图 4.24 注册信息填写图

如图 4.24，填写了注册信息以后点击 OK，注意：此处填写的密码是 123456，
得到了如图 4.25 的反馈提示，新注册的账号是 60003。

127.0.0.1 显示
你的账号是60003,请一定记住这个账号!



图 4.25 注册成功提示图

可以先在数据库中进行验证，如图 4.26 显示，验证结果符合预期，注册成功。

```
mysql> select * from User_t;
+-----+-----+-----+-----+-----+
| U_ID | U_PASSWORD | U_NAME   | U_TELEPHONE | U_BALANCE |
+-----+-----+-----+-----+-----+
| 60001 | tonybrown911 | Tony Brown | 13900000011 | 5900.00 |
| 60002 | 123456       | Tom Jones | 18013257890 | 5491.65 |
| 60003 | 123456       | Tester1  | 18051806789 | 5000.00 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

图 4.26 MySQL 验证注册结果图

然后进入前端进行测试登录：

The screenshot shows a login interface for a flight booking system. The input fields are highlighted with a red border. The account 'Tester1' and password '*****' are entered. A message on the right says 'Sorry, this Account do not exist' and a blue '确定' (Confirm) button is visible.

图 4.27 登录测试 1

如图 4.27，输入一个不正确的账户，提示了账户不存在。

The screenshot shows a login interface for a flight booking system. The input fields are highlighted with a red border. The account '60003' and password '*****' are entered. A message on the right says '密码错误!' (Incorrect password) and a blue '确定' (Confirm) button is visible.

图 4.28 登录测试 2

如图 4.28，输入一个错误的密码，提示密码不正确。

The screenshot shows a login interface for a flight booking system. The input fields are highlighted with a red border. The account '60003 or 1=' and password '*****' are entered. A message on the right says '账户必须是全数字!' (The account must be all digits) and a blue '确定' (Confirm) button is visible.

图 4.29 登录测试 3

如图 4.29，试图进行 SQL 注入，提示账户必须全是数字。
输入正确的密码，登录成功，跳转到界面如图 4.30.

The screenshot shows a search interface for flight bookings. It includes fields for '出发城市' (Departure City) set to '北京', '目的城市' (Destination City) set to '上海', and '出发时间' (Departure Time) set to '2019/06/27'. Below the fields are buttons for '查票' (Check Ticket), '充值' (Recharge), and '历史' (History).

图 4.30 登录成功界面

查询某天从北京到上海的机票，得到如图 4.31 的界面，可以看到，查询从上海到北京的机票可以查到北京到上海两个机场的机票：

The screenshot shows a table of flight search results. The columns are: 飞机编号 (Flight Number), 机型 (Model), 出发机场 (Departure Airport), 目的机场 (Arrival Airport), 出发日期 (Departure Date), 出发时间 (Departure Time), 降落时间 (Arrival Time), 经济舱价格 (Economy Class Price), 商务舱价格 (Business Class Price), 头等舱价格 (First Class Price), 经济舱余票 (Economy Class Remaining Seats), 商务舱余票 (Business Class Remaining Seats), and 头等舱余票 (First Class Remaining Seats). The table lists three flights: one from Beijing Capital International to Shanghai Pudong International, one from Beijing Capital International to Shanghai Hongqiao International, and one from Beijing Capital International to Shanghai Hongqiao International.

飞机编号	机型	出发机场	目的机场	出发日期	出发时间	降落时间	经济舱价格	商务舱价格	头等舱价格	经济舱余票	商务舱余票	头等舱余票
358	B777	北京首都国际机场	上海浦东国际机场	2019-06-27	14:55:00	16:57:00	1710.00	2250.45	3960.00	Y	Y	Y
351	A330	北京首都国际机场	上海虹桥国际机场	2019-06-27	12:35:00	14:37:00	1665.00	2241.45	3690.31	Y	Y	Y
433	A330	北京首都国际机场	上海虹桥国际机场	2019-06-27	16:30:00	18:35:00	1656.00	2392.00	3588.00	Y	Y	Y

图 4.31 机票查询界面

现在可以选择一个机票进行订购，这里对同一个飞机的机票多订购几次方便后续的测试。总共订购了同一价飞机的 5 张机票，显示结果都如图 4.32 显示。

127.0.0.1 显示

订票成功!

图 4.32 机票订购界面

可以通过查询历史记录得到订购的机票如图 4.33，通过对比可以得出，符合订购的顺序。

付款/取票/退票模块											
票/单号: <input type="text"/>											
选 项: <input type="radio"/> 付款 <input type="radio"/> 取票											
<input type="radio"/> 退票											
<input type="button" value="返回"/> <input type="button" value="确定"/>											
用户编号	用户名	用户电话	用户余额								
60003	Tester1	18051806789	5000.00								
订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	0	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00
90026	60003	2019-06-18 21:47:39	0	2250.45	120026	0		2019-06-27 14:55:00	358	C	2250.45
90027	60003	2019-06-18 21:47:55	0	1710.00	120027	0		2019-06-27 14:55:00	358	E	1710.00
90028	60003	2019-06-18 21:48:08	0	1710.00	120028	0		2019-06-27 14:55:00	358	E	1710.00
90029	60003	2019-06-18 21:48:19	0	2250.45	120029	0		2019-06-27 14:55:00	358	C	2250.45

图 4.33 机票订单历史界面

也可以通过数据库中查询得到相同的结果，由于结果一致，此处便不展示了。再对一个订单进行付款操作：

付款/取票/退票模块											
票/单号: 90025											
选 项: <input checked="" type="radio"/> 付款 <input type="radio"/> 取票											
<input type="radio"/> 退票											
<input type="button" value="返回"/> <input type="button" value="确定"/>											
用户编号	用户名	用户电话	用户余额								
60003	Tester1	18051806789	5000.00								
订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	1	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00

图 4.34 机票付款选择(左)与付款结果(右)

付款成功后，用户余额也相应地减少了，且已付款状态已经更新，如图 4.35 所示：

付款/取票/退票模块											
票/单号: <input type="text"/>											
选 项: <input type="radio"/> 付款 <input type="radio"/> 取票											
<input type="radio"/> 退票											
<input type="button" value="返回"/> <input type="button" value="确定"/>											
用户编号	用户名	用户电话	用户余额								
60003	Tester1	18051806789	1040.00								
订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	1	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00

图 4.35 付款成功后的状态

此时再对另一张机票进行购买操作，可以得到余额不足的提示，如图 4.36 所示。

127.0.0.1 显示

Not enough balance to pay for the order

确定

图 4.36 余额不足提示图

此时选择充值功能对账户进行充值进行测试，如图 4.37 所示。



图 4.37 充值操作与充值成功提示图

由图 4.38 所示，可以知道充值成功。

付款/取票/退票模块

票/单号:

选 项: 付款 取票
 退票

用户编号	用户名称	用户电话	用户余额
60003	Tester1	18051806789	11040.00

图 4.38 充值操作结果显示图

此时对刚才已经买过的一张票进行再次购买，可以得到提示已经购买过了。如图 4.39 所示：

127.0.0.1 显示

It is already paid

确定

图 4.39 对已经购买的订单再次购买提示图

再购买一些机票，得到的结果如图 4.40 所示。

返回 确定

用户编号	用户名称	用户电话	用户余额
60003	Tester1	18051806789	7079.55

订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	1	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00
90026	60003	2019-06-18 21:47:39	1	2250.45	120026	0		2019-06-27 14:55:00	358	C	2250.45
90027	60003	2019-06-18 21:47:55	1	1710.00	120027	0		2019-06-27 14:55:00	358	E	1710.00
90028	60003	2019-06-18 21:48:08	0	1710.00	120028	0		2019-06-27 14:55:00	358	E	1710.00
90029	60003	2019-06-18 21:48:19	0	2250.45	120029	0		2019-06-27 14:55:00	358	C	2250.45

图 4.40 继续购买 2 张机票结果图

下面对取票进行测试，首先取 120028 这张票，由于取票时间还没有到，所以不能取票，如图 4.41 所示。

127.0.0.1 显示

To early to do this

确定

图 4.41 过早取票结果图

下面预订一张一天以内出发的机票，并且进行付款，付款后进行取票，结果取票成功如图 4.42 所示。

127.0.0.1 显示

取票成功

确定

图 4.42 取票成功结果图

再看历史记录，取票成功，并且记录了取票时间，如图 4.43 所示。

订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	1	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00
90026	60003	2019-06-18 21:47:39	1	2250.45	120026	0		2019-06-27 14:55:00	358	C	2250.45
90027	60003	2019-06-18 21:47:55	1	1710.00	120027	0		2019-06-27 14:55:00	358	E	1710.00
90028	60003	2019-06-18 21:48:08	0	1710.00	120028	0		2019-06-27 14:55:00	358	E	1710.00
90029	60003	2019-06-18 21:48:19	0	2250.45	120029	0		2019-06-27 14:55:00	358	C	2250.45
90030	60003	2019-06-19 00:06:09	1	1584.00	120030	0	2019-06-19 00:08:25	2019-06-19 08:32:00	432	E	1584.00

图 4.43 取票成功历史记录结果图

对于用户，最后再测试取消票的功能。如果取消一张已经取了的票，会报错，如图 4.44 所示。

127.0.0.1 显示

The ticket is already got

确定

图 4.44 取消票失败历史记录结果图

如果取消一张没有购买的票，不会退款。如图 4.45 所示

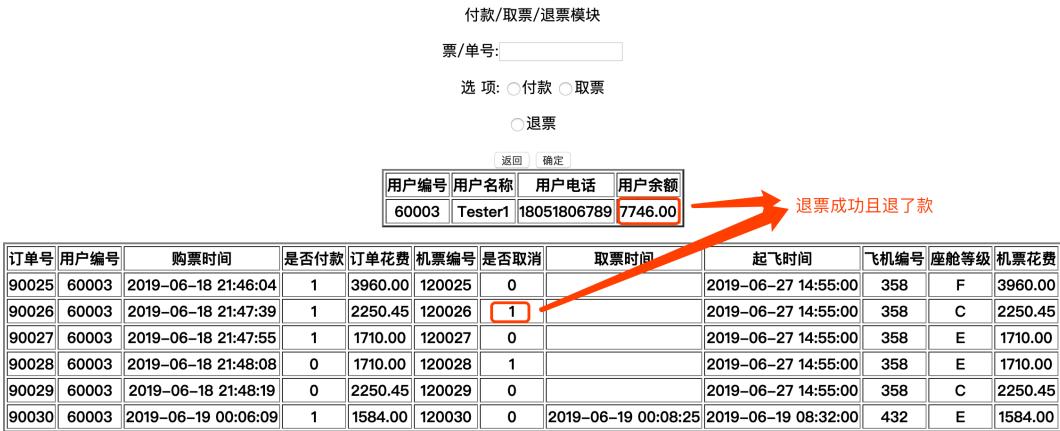
用户编号	用户名	用户电话	用户余额
60003	Tester1	18051806789	5495.55

退票成功,但没有退款

订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	1	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00
90026	60003	2019-06-18 21:47:39	1	2250.45	120026	0		2019-06-27 14:55:00	358	C	2250.45
90027	60003	2019-06-18 21:47:55	1	1710.00	120027	0		2019-06-27 14:55:00	358	E	1710.00
90028	60003	2019-06-18 21:48:08	0	1710.00	120028	1		2019-06-27 14:55:00	358	E	1710.00
90029	60003	2019-06-18 21:48:19	0	2250.45	120029	0		2019-06-27 14:55:00	358	C	2250.45
90030	60003	2019-06-19 00:06:09	1	1584.00	120030	0	2019-06-19 00:08:25	2019-06-19 08:32:00	432	E	1584.00

图 4.45 退票成功历史记录结果图 1

若退了一张已经付款的票，会一并退款，如图 4.46 所示



订单号	用户编号	购票时间	是否付款	订单花费	机票编号	是否取消	取票时间	起飞时间	飞机编号	座舱等级	机票花费
90025	60003	2019-06-18 21:46:04	1	3960.00	120025	0		2019-06-27 14:55:00	358	F	3960.00
90026	60003	2019-06-18 21:47:39	1	2250.45	120026	1		2019-06-27 14:55:00	358	C	2250.45
90027	60003	2019-06-18 21:47:55	1	1710.00	120027	0		2019-06-27 14:55:00	358	E	1710.00
90028	60003	2019-06-18 21:48:08	0	1710.00	120028	1		2019-06-27 14:55:00	358	E	1710.00
90029	60003	2019-06-18 21:48:19	0	2250.45	120029	0		2019-06-27 14:55:00	358	C	2250.45
90030	60003	2019-06-19 00:06:09	1	1584.00	120030	0	2019-06-19 00:08:25	2019-06-19 08:32:00	432	E	1584.00

图 4.46 退票成功历史记录结果图 2

下面测试管理员的功能。

首先测试一下统计上座率和收益的功能，如图所示，可以根据多个维度进行排序，如图 4.47 显示的是根据飞机编号降序排序得到的表格。

日期	飞机编号	经济舱上座率	商务舱上座率	经济舱上座率	收益
2019-06-09	433	0	0	0	0.00
2019-06-08	433	0	0	0	0.00
2019-06-07	433	0	0	0	0.00
2019-06-13	433	0	0.05	0	2392.00
2019-06-05	433	0	0.05	0	2392.00
2019-06-06	433	0	0	0	0.00
2019-06-04	433	0	0	0	0.00

图 4.47 统计功能测试结果图 1

如图 4.48 显示的是根据收益降序得到的表格。

日期	飞机编号	经济舱上座率	商务舱上座率	经济舱上座率	收益
2019-06-27	358	0.03	0	0.07	7920.45
2019-06-20	351	0.07	0	1.3	7020.32
2019-06-19	350	0	0	0	3690.32
2019-07-11	351	0	0	1.3	3690.32
2019-06-05	433	0	0.05	0	2392.00
2019-06-13	433	0	0.05	0	2392.00
2019-06-18	351	0	0	0	2241.45
2019-06-19	432	0.01	0	0	1584.00

图 4.48 统计功能测试结果图 2

如图 4.49 展示的是按照月份统计的收益总额(按月份排序)。

月份	收益总额
2019-05	0
2019-06	13256.66
2019-07	3690.32
2019-08	0

图 4.49 按月统计营业额功能测试结果图 1

如图 4.50 展示的是按照月份统计的收益总额(按收益总额降序)。

月份	收益总额
2019-06	13256.66
2019-07	3690.32
2019-08	0
2019-05	0

返回

图 4.50 按月统计营业额功能测试结果图 2

下面测试添加一架航班，航班填写信息如图 4.51 所示。

欢迎进入添加飞机模块

飞机编号:	599
飞机类型:	A320
出发时间:	上午 09:39
飞行时长:	120
出发城市:	SYX
目的城市:	WUH
开始服务时间:	2019/06/01
结束服务时间:	2019/06/30
头等舱数量:	10
商务舱数量:	30
经济舱数量:	50
<input type="button" value="添加"/>	

返回

图 4.51 航班信息填写图

点击添加以后，可以再数据库中查询到这个航班，如图 4.52 所示。

F_ID	F_TYPE	DEPART_TIME	DURATION	DEPART_PLACE	ARRIVE_PLACE	BEGIN_SERVICE	END_SERVICE	FSEAT_NUMBER	ESEAT_NUMBER	CSEAT_NUMBER	
350	A330	16:35:00		121	SHA	PEK	2019-05-01 12:00:00	2019-10-01 23:00:00	10	30	260
351	A330	12:35:00		122	PEK	SHA	2019-05-01 12:00:00	2019-10-01 23:00:00	10	30	260
358	B777	14:55:00		122	PEK	PVG	2019-05-01 12:00:00	2019-08-01 23:00:00	15	32	280
359	B777	19:55:00		121	PVG	PEK	2019-05-01 12:00:00	2019-08-01 23:00:00	15	32	280
401	A320	16:00:00		42	NKG	SHA	2019-06-01 00:00:00	2019-06-30 00:00:00	10	30	20
402	A330	12:35:00		122	PEK	SHA	2019-05-01 12:00:00	2019-10-01 23:00:00	10	30	260
404	A320	21:20:00		50	CAN	SZX	2019-06-08 00:00:00	2019-06-30 00:00:00	10	50	20
432	A330	08:32:00		120	SHA	PEK	2019-06-01 00:00:00	2019-09-19 00:00:00	15	100	30
433	A330	16:30:00		125	PEK	SHA	2019-06-01 00:00:00	2019-09-11 00:00:00	12	100	20
599	A320	09:39:00		120	SYX	WUH	2019-06-01 00:00:00	2019-06-30 00:00:00	10	50	30

10 rows in set (0.00 sec)

图 4.52 添加航班结果图

再测试添加飞行日志的功能，此处 599 航班从 6 月 20 号到 6 月 28 号每隔 3 天发一般的飞机。填写信息如图 4.53。

欢迎进入添加飞机模块

飞机编号:	599
开始飞行日期:	2019/06/20
结束飞行日期:	2019/06/28
间隔天数:	3
经济舱折扣:	90
商务舱折扣:	90
头等舱折扣:	90
经济舱金额:	1000
商务舱金额:	2000
头等舱金额:	3000
<input type="button" value="添加"/>	

返回

图 4.53 飞行计划填写图

添加结果如图 4.54 所示，易知添加成功。

f_date	f_FID	f_ediscount	f_cdiscount	f_fdiscount	e_price	c_price	f_price	e_taken	c_taken	f_taken	revenue
2019-06-20	599	90	90	90	1000.00	2000.00	3000.00	0	0	0	0.00
2019-06-23	599	90	90	90	1000.00	2000.00	3000.00	0	0	0	0.00
2019-06-26	599	90	90	90	1000.00	2000.00	3000.00	0	0	0	0.00

3 rows in set (0.00 sec)

图 4.54 飞行计划添加结果图

最后测试一下取消飞行计划的功能，取消 599 号航班 6 月 23 号的飞行计划。填写内容如图 4.55。



图 4.55 取消飞行计划填写图

根据图 4.56 的查询结果，可知功能实现成功。

```
mysql> select * from Flying_date where f_FID = 599;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| f_date | f_FID | f_ediscount | f_cdiscount | f_fdiscount | e_price | c_price | f_price | e_taken | c_taken | f_taken | revenue |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2019-06-20 | 599 | 90 | 90 | 90 | 1000.00 | 2000.00 | 3000.00 | 0 | 0 | 0 | 0.00 |
| 2019-06-23 | 599 | 90 | 90 | 90 | 1000.00 | 2000.00 | 3000.00 | 0 | 0 | 0 | 0.00 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

图 4.56 取消飞行计划结果图

4.8 系统设计与实现总结

最终，代码的文件目录如下。

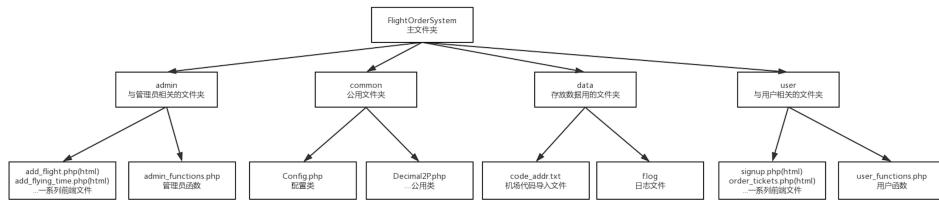


图 4.57 文件目录结构图

如图 4.57，主文件夹下包含了 admin, common, data 和 user 四个从文件夹，分别存放于管理员相关的文件，公用的文件，数据文件和与用户相关的文件。其中 admin 和 user 都包含了一系列前端的 HTML+PHP+JS 的前端文件，也包含了后台的处理函数 admin_functions.php 和 user_functions.php。公共文件夹里包含了 4.5.1 中实现的公共类，data 则放入一些基础导入的数据和导出的日志文件。

首先，完成了 PHP7 环境的配置，Apache 服务器的配置，MySQL 基本环境的配置，使得所有的实验环境都可以正常使用。

第二，根据 E-R 图，完成了 MySQL 中模式的创建，用了两个不同的用户，分别具有不同的权限，管理员和客户分别通过不同的用户对数据库进行连接。此外，还在数据库中创建了多个基本表，完成了用户完整性定义。并设定了一些视图、触发器、事件等方便上层进行调用，也增加了数据表中的完整性。在数据库中就进行了单元测试。

第三，完成了后端的设计，后端中注意到了正确性的问题，在正确性的前提下完成提升效率，包括优化 SQL 语句，设置不同隔离级别，设置重试次数等方法。此外还保证了一定的安全性。在完成了后端的时候，进行了后端-数据库的集成测试。

第四，根据 UI 的逻辑设计，对前端进行了实现，最后进行了整体的集成测试。

5 课程总结

此次课程实践主要分为三次。

1.在基本 SQL 功能学习与实践的部分，我学会了各种 SQL 语句的基本操作，包括数据定义、数据操作、导入、增删改差等等；体验到了三个完整性的重要性，也感受到了如何优化数据库查询语句。基本上熟练掌握了 MySQL 相关的查询语句等，也可以在复杂限制条件下尽可能优化自己的查询代码。

第一个部分中需要改进的部分主要是需要自己仔细把题意读懂，还有很重要的一点是在之后的闲暇时间里可以多研究研究如何优化 SQL 查询。

2.在 SQL 进阶功能学习与实践的部分，我深入体会了事务、触发器、存储函数、存储过程等进阶特性给数据库带来的便利性。体会了不同事物隔离级别之间的差距，不同的隔离级别会带来怎样的问题，效率如何。

第二个部分中需要改进的部分主要是应该深入了解数据库是如何实现事务、触发器、存储函数等。因为基本已经了解了功能，就更应该去看 MySQL 的源码，最重要的就是如何实现事务。

3.在综合实践过程中，我利用了软件工程的分析方法，从需求分析开始，到总体设计，各部分的设计以及详细设计。利用了 B/S 架构，全面地学习了 PHP 语言，在过程中也学会了如何配置 MySQL 的基本配置项，以及如何配置 Apache 的 Web 服务器。在综合实践中，用多种不同的隔离级别尽可能保证并发下的效率，用重试次数来提高操作的成功率，同时也尽可能减少单个用户占用过多的服务器带宽。此外，我还运用到了很多异常处理来保证程序的正确性，也尽可能的保证了程序防止 SQL 注入、JavaScript 代码注入与 XSS 攻击。也学会了 HTTP 中如何充分运用 POST 和 GET，如何利用 Session 或 Cookie。

当然也存在一些问题，比如底层有些逻辑还可以再进行优化，有些代码放置的位置还不合理。SQL 语句还没有进行二次优化，最重要的是前端需要进行优化。本次实验有一个很拖后腿的点就是前端，我在实验过程中尽可能的想用 PHP+HTML 解决所有的问题，但是最后发现 JavaScript 是必须的，由于运用了 PHP，很多操作只能通过 POST 获得所有的参数，但是再 POST 之后，浏览器会自动清理掉一些从别的网页 POST 来的内容，这时需要用到 Session 和 Cookie 来进行保存。我尝试了很多方法，还是有一些东西保存不下来，只能保存部分的内容，这是一个必须要用 JavaScript 来解决的问题，因为 PHP 不能支持动态前端，最后由于时间的关系没有完成 JavaScript 的学习和应用，是一个遗憾，也是一个非常需要改进的地方。

附录

综合实践设计的数据库创建部分 SQL 代码如下：

```
create table User_t (
    U_ID int primary key,                      # 用户 ID
    U_PASSWORD char(30) not null,                # 用户密码
    U_NAME char(30) not null,                    # 用户名字
    U_TELEPHONE char(20) not null,              # 用户电话
    U_BALANCE decimal(8,2) not null default 5000# 用户余额, decimal
);

create table code_addr (
    AP_CODE char(3) primary key,                # 机场代码
    AP_NAME char(50),                          # 机场名称
    AP_CITY char(20)                           # 机场所在城市
);

create table Flight_t (
    F_ID int primary key,                      # 航班 ID
    F_TYPE char(10) not null,                  # 飞机类型
    DEPART_TIME time not null,                # 出发时间 假设每天都在同一时间出发
    DURATION int not null,                   # 飞行时长
    DEPART_PLACE char(3) not null,            # 出发地
    ARRIVE_PLACE char(3) not null,            # 到达地
    BEGIN_SERVICE datetime not null,          # 开始使用的时间
    END_SERVICE datetime not null,            # 结束使用的时间(检修等)
    FSEAT_NUMBER int not null,                # 头等舱座位数量
    ESEAT_NUMBER int not null,                # 经济舱座位数量
    CSEAT_NUMBER int not null,                # 经济舱座位数量
);

create table Flying_date (
    f_date date,                                # 日期
    f_FID int,                                  # 航班 ID
    f_ediscount int not null default 0,          # 经济舱折扣
    f_cdiscount int not null default 0,
    f_fdiscount int not null default 0,
    e_price decimal(8,2) not null,                # 经济舱价格
    c_price decimal(8,2) not null,
    f_price decimal(8,2) not null,
    e_taken int not null default 0,              # 经济舱已经被订的数量
    c_taken int not null default 0,
    f_taken int not null default 0,
    revenue decimal(11, 2) not null default 0.00, # 营收
    foreign key(f_FID) references Flight_t(F_ID),
);
```

```

primary key(f_date, f_FID)
);

create table Order_t (
    O_ID int primary key,                      # 订单号
    O_UID int,                                 # 所属用户
    O_TIME datetime not null,                  # 订的时间
    O_PAID boolean not null default FALSE,   # 是否付款
    O_COST decimal(8,2) not null,              # 订单价格
    foreign key(O_UID) references User_t(U_ID)
);

create table Ticket_t (
    T_ID int primary key,                      # 票 ID
    T_OID int,                                # 所属的订单 ID
    T_CANCELED boolean not null default FALSE, # 是否被取消
    T_GOTTIME datetime default null,          # 取票时间, 空为没取票
    T_TOKEOFFTIME datetime not null,          # 飞机起飞时间
    T_FID int,                                # 航班 ID
    T_SEATCLASS enum('F', 'C', 'E'),           # 座舱级别
    T_PRICE decimal(8,2) not null,              # 票价格
    foreign key(T_OID) references Order_t(O_ID),
    foreign key(T_FID) references Flight_t(F_ID)
);

delimiter $$

drop trigger if exists trg1 $$

create trigger trg1 before update
on Flying_date for each row
begin
    if new.e_taken = old.e_taken + 1 then
        set new.revenue = old.revenue + old.f_ediscount / 100 * old.e_price;
    elseif new.e_taken = old.e_taken - 1 then
        set new.revenue = old.revenue - old.f_ediscount / 100 * old.e_price;
    elseif new.c_taken = old.c_taken + 1 then
        set new.revenue = old.revenue + old.f_cdiscount / 100 * old.c_price;
    elseif new.c_taken = old.c_taken - 1 then
        set new.revenue = old.revenue - old.f_cdiscount / 100 * old.c_price;
    elseif new.f_taken = old.f_taken + 1 then
        set new.revenue = old.revenue + old.f_fdiscount / 100 * old.f_price;
    elseif new.f_taken = old.f_taken - 1 then
        set new.revenue = old.revenue - old.f_fdiscount / 100 * old.f_price;
    end if;
end$$

delimiter ;

```

```

# update Flying_date set e_taken = e_taken - 1 where f_date = '2019-08-31' and f_FID = 351;

drop view if exists v1;
create view v1 as
select Flying_date.f_date, Flying_date.f_FID, e_taken, c_taken, f_taken, revenue,
FSEAT_NUMBER, ESEAT_NUMBER, CSEAT_NUMBER
from Flying_date join Flight_t on Flying_date.f_FID = Flight_t.F_ID;
select * from v1;

Drop EVENT IF EXISTS temp_event;
delimiter $$

CREATE EVENT IF NOT EXISTS temp_event
ON SCHEDULE EVERY 1 DAY STARTS DATE_ADD(DATE_ADD(CURDATE(), INTERVAL 1 DAY), INTERVAL 1 HOUR)
ON COMPLETION PRESERVE ENABLE
DO update Flying_date set e_taken = e_taken - 1 where (f_FID, f_date) in (select
T_FID,date(T_TOKEOFFTIME) from Order_t join Ticket_t on ticket_t.T_OID = Order_t.O_ID where
O_PAID = false and T_CANCELED = false and T_SEATCLASS = 'C');
$$
delimiter ;

Drop EVENT IF EXISTS temp_event_1;
delimiter $$

CREATE EVENT IF NOT EXISTS temp_event_1
ON SCHEDULE EVERY 1 DAY STARTS DATE_ADD(DATE_ADD(CURDATE(), INTERVAL 1 DAY), INTERVAL 1 HOUR)
ON COMPLETION PRESERVE ENABLE
DO update Flying_date set f_taken = f_taken - 1 where (f_FID, f_date) in (select
T_FID,date(T_TOKEOFFTIME) from Order_t join Ticket_t on ticket_t.T_OID = Order_t.O_ID where
O_PAID = false and T_CANCELED = false and T_SEATCLASS = 'F');
$$
delimiter ;

Drop EVENT IF EXISTS temp_event_2;
delimiter $$

CREATE EVENT IF NOT EXISTS temp_event_2
ON SCHEDULE EVERY 1 DAY STARTS DATE_ADD(DATE_ADD(CURDATE(), INTERVAL 1 DAY), INTERVAL 1 HOUR)
ON COMPLETION PRESERVE ENABLE
DO update Flying_date set e_taken = e_taken - 1 where (f_FID, f_date) in (select
T_FID,date(T_TOKEOFFTIME) from Order_t join Ticket_t on ticket_t.T_OID = Order_t.O_ID where
O_PAID = false and T_CANCELED = false and T_SEATCLASS = 'E');
$$
delimiter ;

Drop EVENT IF EXISTS temp_event_3;

```

```

delimiter $$

CREATE EVENT IF NOT EXISTS temp_event_3
ON SCHEDULE EVERY 1 DAY STARTS DATE_ADD(DATE_ADD(CURDATE(), INTERVAL 1 DAY), INTERVAL 1 HOUR)
ON COMPLETION PRESERVE ENABLE
DO update Ticket_t set T_CANCELED = 1 where (T_ID, T_FID) in (select T_ID, T_FID from Order_t
join Ticket_t on ticket_t.T_OID = Order_t.O_ID where O_PAID = false and T_CANCELED = false);
$$
delimiter ;

load data infile "/Users/yonginxu/Codes/code_addr.txt" into table code_addr fields
terminated by ',' optionally enclosed by "" lines terminated by '\n';

create user 'customer'@'localhost' IDENTIFIED BY '123456';
grant all privileges on ticket.User_t to 'customer'@'localhost';
grant all privileges on ticket.Order_t to 'customer'@'localhost';
grant all privileges on ticket.Ticket_t to 'customer'@'localhost';
grant select on ticket.* to 'customer'@'localhost';
grant update on ticket.Flying_date to 'customer'@'localhost';

update Flying_date set e_taken;

```

机票预订系统后端主要代码：（前端代码过多，且多数设计到 CSS 样式和 HTML 组件此处不展示了）

配置类 Config.php 记录了数据库、数据库用户的一些基本信息，都是 CONST 定义的一些常量。

```

<?php
namespace config;

final class BJ_time {
    public static function get_current_datetime() {
        return date("Y-m-d H:i:s", time() + 8 * 3600) ;
    }

    public static function get_current_time() {
        return date("H:i:s", time() + 8 * 3600) ;
    }

    public static function get_current_date() {
        return date("Y-m-d", time() + 8 * 3600) ;
    }

    public static function get_limited_datetime() {
        return date("Y-m-d H:i:s", time() + 11 * 3600) ;
    }
}

```

```

}

final class DB_info {
    public const SERVER_ADDRESS = "127.0.0.1";
    public const DATABASE_ADMIN_NAME = "seller";
    public const DATABASE_ADMIN_PSW = "123456";
    public const DATABASE_NAME = "ticket";
    public const DATABASE_COSTUMER_NAME = "customer";
    public const DATABASE_COSTUMER_PSW = "123456";
}

final class Code_CITY {
    // Code to Address and City
    public const NAME = "code_addr";
    public const CODE = "AP_CODE";           // primary key, @datatype: char(3)
    public const AP_NAME = "AP_NAME";        // @datatype: char(50)
    public const CITY = "AP_CITY";          // @datatype: char(20)
}

final class User_table {
    // User table
    public const NAME = "User_t";
    public const ID = "U_ID";                // primary key, @datatype: int
    public const PASSWORD = "U_PASSWORD";    // @datatype: char(30)
    public const UNAME = "U_NAME";           // @datatype: char(30)
    public const TELEPHONE = "U_TELEPHONE"; // @datatype: char(20)
    public const BALANCE = "U_BALANCE";      // @datatype: decimal(8,2)
}

final class Order_table {
    // Order table
    public const NAME = "Order_t";
    public const ID = "O_ID";                // primary key, @datatype: char(30)
    public const UID = "O_UID";              // foreign key from User table
                                            // @datatype: int
    public const TIME = "O_TIME";            // @datatype: datetime
    public const PAID = "O_PAID";            // @datatype: boolean
    public const COST = "O_COST";            // @datatype: decimal(8,2)
}

final class Flight_table {
    // Flight table
    public const NAME = "Flight_t";
    public const ID = "F_ID";                // primary key, @datatype: int
    public const TYPE = "F_TYPE";             // @datatype: char(10)
    public const DEPART_TIME = "DEPART_TIME"; // @datatype: time
}

```

```

public const DURATION = "DURATION";           // @datatype: int
public const DEPART_PLACE = "DEPART_PLACE"; // @datatype: char(3)
public const ARRIVE_PLACE = "ARRIVE_PLACE"; // @datatype: char(3)
public const BEGIN_SERVICE = "BEGIN_SERVICE"; // @datatype: datetime
public const END_SERVICE = "END_SERVICE";    // @datatype: datetime
public const FSEAT_NUMBER = "FSEAT_NUMBER"; // @datatype: int
public const CSEAT_NUMBER = "CSEAT_NUMBER"; // @datatype: int
public const ESEAT_NUMBER = "CSEAT_NUMBER"; // @datatype: int
}

final class Ticket_table {
    // Ticket table
    public const NAME = "Ticket_t";
    public const ID = "T_ID";                      // primary key @datatype: int
    public const OID = "T_OID";                    // foreign key from Order table
                                                // @datatype: int
    public const CANCELED = "T_CANCELED";          // @datatype: boolean
    public const GOT_TIME = "T_GOTTIME";           // @datatype: datetime
    public const TOOKOFF_TIME = "T_TOKEOFFTIME"; // @datatype: datetime
    public const FID = "T_FID";                    // foreign key from Flight table
                                                // @datatype: int
    public const SEATCLAS = "T_SEATCLASS";         // @datatype: enum('F', 'C', 'E')
    public const PRICE = "T_PRICE";                // @datatype: decimal(8,2)
}
}

final class Flying_date_table {
    // Flying date table
    public const NAME = "Flying_date";
    public const FDATE = "f_date";                 // @datatype: date, part of priamry key
    public const FID = "f_FID";                   // @datatype: int, part of priamry key
    public const EDISCOUNT = "f_ediscount";        // @datatype: int
    public const CDISCOUNT = "f_cdiscount";        // @datatype: int
    public const FDISCOUNT = "f_fdiscount";        // @datatype: int
    public const EPRICE = "e_price";               // @datatype: decimal(8,2)
    public const CPRICE = "c_price";               // @datatype: decimal(8,2)
    public const FPRICE = "f_price";               // @datatype: decimal(8,2)
    public const ETAKEN = "e_taken";               // @datatype: int , default 0
    public const CTAKEN = "c_taken";               // @datatype: int , default 0
    public const FTAKEN = "f_taken";               // @datatype: int , default 0
    public const REVENUE = "revenue";              // @datatype: decimal(11, 2) , default 0.00
}
}

final class Views {
    public const DATA_SEL = "v1";
}
?>

```

`DB_Connector.php` 提供了连接到数据库的一些接口，也提供了一些公用的查询语句接口。

```
<?php

include_once 'config.php';

use config\Code_CITY;
use config\DB_info as INFO;

class DBException extends Exception {
    function __construct($message = "", $code = 0, Throwable $previous = null) {
        parent::__construct($message, $code, $previous);
    }
}

class DBConnector {
    private $db_host;
    private $db_user;
    private $db_password;
    private $db_name;
    public $link;

    /*
     * construct connection to MySQL based on parameters
     * pay ATTENTION that these parameters (including the link itself) are PRIVATE variables
     */
    function __construct(bool $admin = true) {
        $this->db_host = INFO::SERVER_ADDRESS;
        $this->db_user = $admin ? INFO::DATABASE_ADMIN_NAME : INFO::DATABASE_COSTUMER_NAME;
        $this->db_password = $admin ? INFO::DATABASE_ADMIN_PSW :
INFO::DATABASE_COSTUMER_PSW;
        $this->db_name = INFO::DATABASE_NAME;

        $this->link = mysqli_connect($this->db_host, $this->db_user, $this->db_password,
$this->db_name);

        try {
            if (!$this->link) {
                echo "Error: Unable to connect to MySQL." . PHP_EOL;
                echo "Debugging errno: " . mysqli_connect_errno() . PHP_EOL;
                echo "Debugging error: " . mysqli_connect_error() . PHP_EOL;
                throw new DBException("Connect to MySQL error", 0, null);
            }
        } catch (DBException $e) {
            echo "Error code ".$e->getCode()." . ".$e->getMessage();
        }
    }
}
```

```

        }

    }

    /**
     * display whole table
     * @param string $table_name: the table name
     * @return array whole table stored in array
     */
    function get_full_table(string $table_name) {
        try {
            $query = "select * " .
                " from " . $table_name . ";";
            $result = $this->link->query($query);
            $number_column = $result->field_count;
            $ret = array();

            // fetch columns and rows
            while ($row = $result->fetch_row()) {
                $temp = array();
                for ($i = 0; $i < $number_column; $i++) {
                    $temp[] = $row[$i];
                }
                $ret[] = $temp;
            }

            $result->free();
            return $ret;
        }
        catch (mysqli_sql_exception $ex) {
            echo $ex->getMessage();
            throw $ex;
        }
    }

    /**
     * get all airports' code
     * @return array the array contains all airports' code in the database
     */
    function get_all_airports_code() {
        try {
            $query = "select " . config\Code_CITY::CODE .
                " from " . config\Code_CITY::NAME . ";";
            $result = $this->link->query($query);
            $ret = array();
            while(list($code) = $result->fetch_row()) {
                $ret[] = $code;
            }
        }
    }
}

```

```

        }
        $result->free();
        return $ret;
    }

    catch (mysqli_sql_exception $ex) {
        echo $ex->getMessage();
        throw $ex;
    }
}

/**
 * @param string $code: the code corresponding to the airport
 * @example: input --> PVG
 *           output --> 上海
 */
function get_city_from_code(string $code, bool $full_code_table = false) {
    try {
        $query = $full_code_table ? "select ".config\Code_CITY::CITY." from
".config\Code_CITY::NAME.";" :
            "select ". config\Code_CITY::CITY .
            " from ". config\Code_CITY::NAME .
            " where ". config\Code_CITY::CODE . " = \\" . $code . "\\";";
        $result = $this->link->query($query);

        if (!$full_code_table) {
            if(list($city) = $result->fetch_row()) {
                $result->free();
                return $city;
            }
            else {
                $result->free();
                return null;
            }
        }
        else {
            $ret = array();
            while(list($city) = $result->fetch_row()) {
                $ret[] = $city;
            }
            $result->free();
            return $ret;
        }
    }

    catch (mysqli_sql_exception $ex) {
        echo $ex->getMessage();
        throw $ex;
    }
}

```

```

        }

    }

    /**
     * @param string $city: a city's name
     * @example: input --> 上海
     *           output --> [PVG, SHA]
     */
    function get_code_from_city(string $city) {
        try {
            $query = "select " . config\Code_CITY::CODE .
                " from " . config\Code_CITY::NAME .
                " where " . config\Code_CITY::CITY . " = \'" . $city . "\';";
            $result = $this->link->query($query);
            $codes = array();
            while(list($code) = $result->fetch_row()) {
                $codes[] = $code;
            }
            $result->free();
            return $codes;
        }
        catch (mysqli_sql_exception $ex) {
            echo $ex->getMessage();
            throw $ex;
        }
    }

    /**
     * @param string $city: a city's name
     * @return array 2D array
     * @example: input --> 上海
     *           output --> [[PVG, 上海浦东国际机场], [SHA, 上海虹桥国际机场]]
     */
    function get_airport_and_code_from_city(string $city) {
        try {
            $query = "select " . config\Code_CITY::CODE . ", " . config\Code_CITY::AP_NAME .
                " from " . config\Code_CITY::NAME .
                " where " . config\Code_CITY::CITY . " = \'" . $city . "\';";
            $result = $this->link->query($query);
            $ret = array();
            while(list($code, $airport) = $result->fetch_row()) {
                $row = array();
                $row[] = $code;
                $row[] = $airport;
                $ret[] = $row;
            }
        }
    }
}

```

```

        $result->free();
        return $ret;
    }

    catch (mysqli_sql_exception $ex) {
        echo $ex->getMessage();
        throw $ex;
    }
}

/*
 * showTables is a function that tests the correctness of a query
 * it uses "show tables" in MySQL
*/
final function showTables() {
    $test_query = "show tables";
    $result = $this->link->query($test_query);
    echo "Tables in ". $this->db_name. " are shown below: <br />";
    while(list($table_name) = $result->fetch_row())
        printf("%s <br />", $table_name);
    $result->free();
}

/**
 * print the whole table, used for debugging
 * @param array $arr the table
*/
final function printWholeTable(array $arr) {
    for ($i = 0; $i < count($arr); $i++) {
        $row = $arr[$i];
        for ($j = 0; $j < count($row); $j++) {
            echo $row[$j] . " ";
        }
        echo "<br />";
    }
}

/*
 * destructor simply close the connection between PHP and MySQL
*/
function __destruct() {
    mysqli_close($this->link);
}

```

Decimal2P.php 提供了处理 2 位顶点小数的类

```
<?php
class decimal2P {
    /* this class only dealt with money that has 2 point precision
     *example: 180.99
    */
    private const PRECISION = 2;
    private $money;
    /**
     * decimal2P constructor.
     * @param $money the input string
     * PLEASE PAY ATTENTION to the VALIDATION CHECK
    */
    public function __construct(string $money) {
        $int_part = 0;
        $frac_part = 0;
        $flag_int = true;
        for ($i = 0; $i < strlen($money); $i++) {
            if ('0' <= $money[$i] && $money[$i] <= '9') {
                if ($flag_int) {
                    $int_part = $int_part * 10 + ($money[$i] - '0');
                }
                else {
                    $frac_part = $frac_part * 10 + ($money[$i] - '0');
                }
            }
            else if ($money[$i] == '.') {
                if ($i + 3 != strlen($money)) {
                    $this->money = null;
                    return;
                }
                $flag_int = false;
            }
            else {
                $this->money = null;
                return;
            }
        }
        $this->money = $int_part * 100 + $frac_part;
    }

    public function compare(decimal2P $b) {
        return ($this->money - $b->getMoney()) >= 0;
    }

    public function compare_str(string $b) {
```

```

        $bm = new decimal2P($b);
        return $this->compare($bm);
    }

    public function plus(decimal2P $b) {
        $this->money += $b->getMoney();
    }

    public function minus(decimal2P $b) {
        $this->money -= $b->getMoney();
    }

    public function multiply(decimal2P $b) {
        $this->money *= (int)($b->getMoney() / 100);
    }

    public function multiply_discount(int $b) {
        $this->money *= ($b / 100);
    }

    public function getMoney() {
        return $this->money;
    }

    /**
     * @return null if the format is wrong / the str of the money
     */
    public function showMoney() {
        if ($this->money == null)
            return null;
        $int_part = (int)($this->money / 100);
        $frac_part = $this->money % 100;
        return ($frac_part == 0) ? $int_part . ".00" : $int_part . "." . $frac_part;
    }

    /**
     * @return string|null Same as showMoney()
     */
    public function __toString() {
        if ($this->money == null)
            return null;
        $int_part = (int)($this->money / 100);
        $frac_part = $this->money % 100;
        return ($frac_part == 0) ? $int_part . ".00" : $int_part . "." . $frac_part;
    }
}

```

Flight.php 记录了航班信息类

```
<?php

class Flight {
    private $fid;
    private $f_type;
    private $depart_time;
    private $duration;
    private $depart_place;
    private $arrive_place;
    private $begin_service_date;
    private $end_service_date;
    private $f_number;
    private $e_number;
    private $c_number;

    public function __construct($fid, $f_type, $depart_time,
                                $duration, $depart_place, $arrive_place,
                                $begin_service_date, $end_service_date,
                                $f_number, $e_number, $c_number) {
        $this->fid = $fid;
        $this->f_type = $f_type;
        $this->depart_time = $depart_time;
        $this->duration = $duration;
        $this->depart_place = $depart_place;
        $this->arrive_place = $arrive_place;
        $this->begin_service_date = $begin_service_date;
        $this->end_service_date = $end_service_date;
        $this->f_number = $f_number;
        $this->e_number = $e_number;
        $this->c_number = $c_number;
    }

    public function __toString() {
        return $this->fid . "," . $this->f_type . "," . $this->depart_time . "," .
        $this->duration . "," .
        $this->depart_place . "," . $this->arrive_place . "," .
        $this->begin_service_date . "," . $this->end_service_date . "," .
        $this->f_number . "," . $this->e_number . "," . $this->c_number;
    }
}
```

User_functions.h 记录了用户定义的一些功能接口。

```
<?php

use config\Flight_table;
use config\Flying_date_table;
use config\Order_table;
use config\Ticket_table;
use config\User_table;

include_once '../common/decimal2P.php';

/**
 * Class user_exception_codes shows the exception codes for user_exception
 */
class user_exception_codes {
    public const InvalidInput = 0;
    public const NameTooLong = 1;
    public const PswTooLong = 2;
    public const TelTooLong = 3;
    public const InsertAcconutFailed = 4;
    public const AccountNotExist = 5;
    public const IDInvalidFormat = 6;
    public const SrcPlaceNotExist = 7;
    public const DstPlaceNotExist = 8;
    public const NoTargetFlight = 9;
    public const ServerBusy = 10;
    public const TooLatetodo = 11;
    public const AlreadyCanceled = 12;
    public const AlreadyPaid = 13;
    public const AlreadyGot = 14;
    public const NotEnoughBalance = 15;
    public const CouldNotFindOrder = 16;
    public const HaventPaid = 17;
    public const TooEarly = 18;
    public const SeatsSoldOut = 19;
    public const CouldNotFindTicket = 20;
}

class user_exception extends Exception {
    public $code;

    public function __construct($code = 0) {
        $this->code = $code;
        parent::__construct("", $code, null);
    }

    public function __toString() {
        switch ($this->code) {
            case user_exception_codes::InvalidInput:
                return "Invalid input";
            case user_exception_codes::NameTooLong:
                return "Name too long.";
            case user_exception_codes::PswTooLong:
                return "Password too long.";
            case user_exception_codes::TelTooLong:
                return "Telephone too long.";
            case user_exception_codes::InsertAcconutFailed:
                return "Insert into user account table failed";
            case user_exception_codes::AccountNotExist:
                return "Sorry, this Account do not exist";
            case user_exception_codes::IDInvalidFormat:
                return "Invalid Format! ID can only be numbers";
            case user_exception_codes::SrcPlaceNotExist:
                return "no flight coming from the search city";
            case user_exception_codes::DstPlaceNotExist:
                return "no flight coming to the search city";
            case user_exception_codes::NoTargetFlight:
                return "No flight satisfy all the conditions";
            case user_exception_codes::ServerBusy:
                return "Server is busy or input is invalid";
            case user_exception_codes::TooLatetodo:
                return "It is too late";
            case user_exception_codes::AlreadyCanceled:
                return "It is already canceled";
            case user_exception_codes::AlreadyPaid:
                return "It is already paid";
            case user_exception_codes::AlreadyGot:
                return "The ticket is already got";
            case user_exception_codes::NotEnoughBalance:
                return "Not enough balance to pay for the order";
        }
    }
}
```

```

        case user_exception_codes::CouldNotFindOrder:
            return "Sorry, could not find the order";
        case user_exception_codes::HaventPaid:
            return "Sorry, you haven't paid for the ticket";
        case user_exception_codes::TooEarly:
            return "To early to do this";
        case user_exception_codes::SeatsSoldOut:
            return "Sorry, this type of tickets are sold out";
        case user_exception_codes::CouldNotFindTicket:
            return "Sorry, couldn't find this ticket";
        default:
            return "Some user exception occurred.";
    }
}

class flight_User {
    /**
     * User basic info
     */
    public $UID;
    public $UName;
    public $UTelephone;
    private $UBalance;

    /**
     * flight_User constructor. Construct basic info
     * @param int $uid User's ID
     * @param string $uname User's Name
     * @param string $utel User's telephone number (string)
     * @param string $ubalance User's balance, defalut $5000.00
     */
    public function __construct(int $uid, string $uname, string $utel, string $ubalance)
    {
        $this->UID = $uid;
        $this->UName = $uname;
        $this->UTelephone = $utel;
        $this->UBalance = $ubalance;
    }

    /**
     * @return string return balance
     */
    final public function getUBalance(): string {
        return $this->UBalance;
    }

    /**
     * @param string $money
     * @example input 500, Ubalance = 5000
     *          after this function Ubalance = 5500
     */
    final public function incBalance(string $money): void {
        $nowbalance = new decimal2P($this->UBalance);
        $bm = new decimal2P($money);
        $nowbalance->plus($bm);
        $this->UBalance = $nowbalance->showMoney();
    }

    /**
     * @param string $money
     * @example input 500, Ubalance = 5000
     *          after this function Ubalance = 4500
     */
    final public function decBalance(string $money): void {
        $nowbalance = new decimal2P($this->UBalance);
        $bm = new decimal2P($money);
        $nowbalance->minus($bm);
        $this->UBalance = $nowbalance->showMoney();
    }

    /**
     * @param string $UBalance
     * Ubalance can only be revised through a class which inherit
     * class flight_User
     */
    final protected function setUBalance(string $UBalance): void {
        $this->UBalance = $UBalance;
    }
}

```

```

final class User_functions {
    /**
     * Constants
     */
    const RETRY_TIMES = 3;

    /**
     * create an account for user, and insert it into User_table
     * @param mysqli $link: mysqli connection
     * @param string $name: user input name
     * @param string $psw: user input password
     * @param string $tel: user input telephone
     * @throws user_exception some exception specified by Code
     */
    public static function create_account(mysqli &$link, string $name, string $psw, string
$tel) {
        if (strlen($name) >= 30) {
            throw new user_exception(user_exception_codes::NameTooLong);
        }
        else if (strlen($psw) >= 30) {
            throw new user_exception(user_exception_codes::PswTooLong);
        }
        else if (strlen($tel) >= 20) {
            throw new user_exception(user_exception_codes::TelTooLong);
        }
        else {
            try {
                $retry_times = self::RETRY_TIMES;
                $succeeded = false;
                do {
                    $link->autocommit(false);
                    $find_max_id_query = "select max(" . config\User_table::ID . ")" .
                        " from " . config\User_table::NAME;
                    $result = $link->query($find_max_id_query);
                    list($maxid) = $result->fetch_row();
                    $id = ($maxid == null) ? 60001 : $maxid + 1;
                    $insert_account = "insert into " . config\User_table::NAME .
                        " values(" . "$id,'$psw','$name','$tel'" . ",5000.00);";
                    $link->query($insert_account, MYSQLI_STORE_RESULT);
                    if ($link->affected_rows != 0) {
                        $link->commit(); // commit this transaction
                        $succeeded = true;
                        $link->autocommit(true);
                        break;
                    }
                    else {
                        $link->rollback();
                        $link->autocommit(true);
                    }
                }while($retry_times--);
                if ($succeeded == false) {
                    // Already tried TRY_TIMES times but still failed, this is mainly because
                    // the server is busy at the time being
                    throw new user_exception(user_exception_codes::InsertAccountFailed);
                }
                return $succeeded ? $id : -1;
            }
            catch (mysqli_sql_exception $ex) {
                echo $ex->getMessage();
                throw $ex;
            }
            catch (user_exception $ex) {
                throw $ex;
            }
            finally {
                // whatever happened, set mode to autocommit
                $link->autocommit(true);
            }
        }
    }

    /**
     * this function is for user login
     * @param mysqli $link      mysqli connection
     * @param string $uid       user input id
     * @param string $upsw      user input password
     * @return flight_User|null return a new user class object, return null if psw is
not correct
     * @throws user_exception some exception specified by Code
     */
}

```

```

public static function login_account(mysqli &$link, string $uid, string $upsw) {
    try {
        if (!is_numeric($uid)) {
            throw new user_exception(user_exception_codes::IDInvalidFormat);
        }
        $query = "select ".config\User_table::PASSWORD . "," . config\User_table::UNAME .
        " , "
        . config\User_table::TELEPHONE . "," . config\User_table::BALANCE .
        " from ". config\User_table::NAME .
        " where ". config\User_table::ID . " = " . $uid . ";";
        $result = $link->query($query);
        list($expect_psw, $uname, $telephone, $ubalance) = $result->fetch_row();
        $result->free();
        if ($expect_psw == null) {
            throw new user_exception(user_exception_codes::AccountNotExist);
        }
        else {
            if (!$expect_psw == $upsw) {
                return new flight_User($uid, $uname, $telephone, $ubalance);
            }
            else {
                return null;
            }
        }
    } catch (mysqli_sql_exception $ex) {
        echo $ex->getMessage();
        throw $ex;
    }
    catch (user_exception $ex) {
        throw $ex;
    }
}

/**
 * this function is for user to search Tickets
 * @param string $target_date
 * @param string $from_place
 * @param string $to_place
 * @throws user_exception
 */
public static function search_tickets(string $target_date,
                                      string $from_place, string $to_place) {
    try {
        $conn = new DBConnector(false);
        $src_arr = $conn->get_airport_and_code_from_city($from_place);
        $dst_arr = $conn->get_airport_and_code_from_city($to_place);

        if (count($src_arr) == 0) {
            throw new user_exception(user_exception_codes::SrcPlaceNotExist);
        }
        else if(count($dst_arr) == 0) {
            throw new user_exception(user_exception_codes::DstPlaceNotExist);
        }
        else if ($target_date == null) {
            throw new user_exception(user_exception_codes::InvalidInput);
        }
        else if (strtotime(config\BJ_time::get_current_date()) >
strtotime($target_date)) {
            throw new user_exception(user_exception_codes::TooLatetoDo);
        }

        $ret = array();
        for ($i = 0; $i < count($src_arr); $i++) {
            for ($j = 0; $j < count($dst_arr); $j++) {
                $flight_query = "select ". config\Flight_table::ID . "," .
config\Flight_table::TYPE . ","
                           . config\Flight_table::DEPART_TIME . "," .
config\Flight_table::DURATION . ","
                           . config\Flight_table::FSEAT_NUMBER . "," .
config\Flight_table::CSEAT_NUMBER . ","
                           . config\Flight_table::ESEAT_NUMBER .
                           " from ". config\Flight_table::NAME .
                           " where ". config\Flight_table::DEPART_PLACE . " = " . "!" .
$src_arr[$i][0] . "!" . " and " . config\Flight_table::ARRIVE_PLACE . " = " . "!" .
$dst_arr[$j][0] . "!";
                $result = $conn->link->query($flight_query);

```

```

        while (list($fid, $ftype, $fdptime, $fduration, $ffn, $fcn, $fen) =
$result->fetch_row()) {
            $spec_query = "select " . config\Flying_date_table::EDISCOUNT . "," .
config\Flying_date_table::FDISCOUNT . "," .
config\Flying_date_table::CPRICE . "," .
config\Flying_date_table::EPRICE . "," .
config\Flying_date_table::ETAKEN . "," .
config\Flying_date_table::CTAKEN . "," .
config\Flying_date_table::FTAKEN .
" from ".config\Flying_date_table::NAME .
" where ".config\Flying_date_table::FID . "=" . $fid .
" and " . config\Flying_date_table::FDATE . "=" . "" .
$target_date . ";";

            $tar = $conn->link->query($spec_query);
            if (list($edis, $cdis, $fdis, $ep, $cp, $fp, $et, $ct, $ft) =
$tar->fetch_row()) {
                $e_final_price = new decimal2P((string)$ep);
                $e_final_price->multiply_discount($edis);
                $c_final_price = new decimal2P($cp);
                $c_final_price->multiply_discount($cdis);
                $f_final_price = new decimal2P($fp);
                $f_final_price->multiply_discount($fdis);
                $yleft = (int)$et < (int)$fen ? "Y" : "N";
                $yleft = (int)$ct < (int)$fcn ? "Y" : "N";
                $yleft = (int)$ft < (int)$ffn ? "Y" : "N";
                $fartime = date("H:i:s", strtotime("+$fduration
min", strtotime($fdptime)));
                $src_airport = $src_arr[$i][1];
                $dst_airport = $dst_arr[$j][1];
                $temp = array($fid, $ftype, $src_airport, $dst_airport,
$target_date, $fdptime, $fartime,
                    $e_final_price, $c_final_price, $f_final_price, $yleft, $yleft,
$yleft);
                $ret[] = $temp;
                $tar->free();
            }
        }
        $result->free();
    }
    if (count($ret) == 0) {
        throw new user_exception(user_exception_codes::NoTargetFlight);
    }
    return $ret;
}
catch (mysqli_sql_exception $ex) {
    throw $ex;
}
catch (user_exception $ex) {
    throw $ex;
}
catch (Exception $ex) {
    throw $ex;
}
}

/**
 * order a ticket for the specific user
 * @param mysqli $link
 * @param flight_User $usr
 * @param string $prc : price of all tickets (in this one order)
 * @param $fid
 * @param $seat_class
 * @param string $offtime : tookoff time of the flight
 * @throws user_exception
 */
public static function order_tickets(mysqli &$link, $uid, $fid, $seat_class, string
$offtime) {
    try {
        $try_times = self::RETRY_TIMES;
        $succeeded = false;

        do {
            $target_date = date("Y-m-d", strtotime($offtime));
            $serializable = "set session transaction isolation level serializable;";
            $link->query($serializable);
            $link->autocommit(false);

```

```

        $spec_query = "select " . config\Flying_date_table::EDISCOUNT . "," .
            config\Flying_date_table::CDISCOUNT . "," .
        config\Flying_date_table::FDISCOUNT . "," .
            config\Flying_date_table::EPRICE . "," .
        config\Flying_date_table::CPRICE . "," .
            config\Flying_date_table::FPRICE . "," .
        config\Flying_date_table::ETAKEN . "," .
            config\Flying_date_table::CTAKEN . "," .
        config\Flying_date_table::FTAKEN .
            " from " . config\Flying_date_table::NAME .
            " where " . config\Flying_date_table::FID . "=" . $fid .
            " and " . config\Flying_date_table::FDATE . "=" . "" . $target_date .
        "';" ;

        $final_price = null;

        $flight_query = "select " .
            config\Flight_table::FSEAT_NUMBER . "," .
        config\Flight_table::CSEAT_NUMBER . "," .
            config\Flight_table::ESEAT_NUMBER .
            " from " . config\Flight_table::NAME .
            " where " . config\Flight_table::ID . "=" . $fid;
        $flight_result = $link->query($flight_query);
        if ((list($ffn, $fcn, $fen) = $flight_result->fetch_row()) == null) {
            continue;
        }
        $tar = $link->query($spec_query);
        echo $spec_query;
        if (list($edis, $cdis, $fdis, $ep, $cp, $fp, $et, $ct, $ft) = $tar->fetch_row())
    {
        if ($seat_class == 'E') {
            if ((int)$et == (int)$fen) {
                throw new user_exception(user_exception_codes::SeatsSoldOut);
            }
            $e_final_price = new decimal2P((string)$ep);
            $e_final_price->multiply_discount($edis);
            $final_price = $e_final_price;
            echo $e_final_price;
        }
        else if ($seat_class == 'C') {
            if ((int)$ct == (int)$fcn) {
                throw new user_exception(user_exception_codes::SeatsSoldOut);
            }
            $c_final_price = new decimal2P($cp);
            $c_final_price->multiply_discount($cdis);
            $final_price = $c_final_price;
            echo $c_final_price;
        }
        else {
            if ((int)$ft == (int)$ffn) {
                throw new user_exception(user_exception_codes::SeatsSoldOut);
            }
            $f_final_price = new decimal2P($fp);
            $f_final_price->multiply_discount($fdis);
            $final_price = $f_final_price;
            echo $f_final_price;
        }
    }
    else {
        continue;
    }

    $oid_query = "select max(" . config\Order_table::ID . ")" .
        " from " . config\Order_table::NAME . ";";

    $oid_result = $link->query($oid_query);
    $cur_time = config\BJ_time::get_current_datetime();

    // if you order the ticket within 3 hours before the flight, you are not allowed
    // to order this ticket
    if (strtotime(config\BJ_time::get_limited_datetime()) >
        strtotime($offtime)) {
        throw new user_exception(user_exception_codes::TooLatetoDo);
    }

    list($oid) = $oid_result->fetch_row();
    $oid = ($oid == null) ? 90001 : $oid + 1;
    $insert_order = "insert into " . config\Order_table::NAME . " values(" .
        "$oid, $uid, '$cur_time', false, '$final_price');";
    echo "$insert_order";
}

```

```

        $link->query($insert_order, MYSQLI_STORE_RESULT);
        if ($link->affected_rows > 0) {
            $tid_query = "select max(`". config\Ticket_table::ID . `") .
                " from `". config\Ticket_table::NAME . `";;

            $tid_result = $link->query($tid_query);
            list($tid) = $tid_result->fetch_row();
            $tid = ($tid == null) ? 120001 : $tid + 1;
            $insert_ticket = "insert into `". config\Ticket_table::NAME . ` values(` .
                "$tid, $oid, false, null, '$offtime', $fid, '$seat_class',
                '$final_price');";

            // echo $insert_ticket;
            $link->query($insert_ticket, MYSQLI_STORE_RESULT);

            if ($link->affected_rows > 0) {
                $update_seats = "";
                $off_date = date("Y-m-d", strtotime($offtime));
                if ($seat_class == 'E') {
                    $update_seats = "update Flying_date set e_taken = e_taken + 1 where
                    f_date = '2019-08-31' and f_FID = 351;";
                    $update_seats = "update ".config\Flying_date_table::NAME." set
                    ".config\Flying_date_table::ETAKEN." = ".config\Flying_date_table::ETAKEN." + 1".
                    " where f_date = '$off_date' and f_FID = '$fid';";
                }
                else if ($seat_class == 'C') {
                    $update_seats = "update ".config\Flying_date_table::NAME." set
                    ".config\Flying_date_table::CTAKEN." = ".config\Flying_date_table::CTAKEN." + 1".
                    " where f_date = '$off_date' and f_FID = '$fid';";
                }
                else {
                    $update_seats = "update ".config\Flying_date_table::NAME." set
                    ".config\Flying_date_table::FTAKEN." = ".config\Flying_date_table::FTAKEN." + 1".
                    " where f_date = '$off_date' and f_FID = '$fid';";
                }
            }
            // echo $update_seats. "<br />";
            $link->query($update_seats, MYSQLI_STORE_RESULT);
            if($link->affected_rows > 0) {
                $succeeded = true;
                $link->commit();
                break;
            }
            else {
                $link->rollback();
            }
        }
        else {
            $link->rollback();
        }
    }
    else {
        $link->rollback();
    }
} while($try_times--);

if ($try_times == 0 && $succeeded == false) {
    throw new user_exception(user_exception_codes::ServerBusy);
}
}

catch (mysqli_sql_exception $ex) {
    throw $ex;
}
catch (user_exception $ex) {
    throw $ex;
}
catch (Exception $ex) {
    throw $ex;
}
finally {
    $link->autocommit(true);
    $serializable = "set session transaction isolation repeatable read;";
    $link->query($serializable);
}
}

/**
 * buy a ticket for the specific user
 * @param mysqli $link
 * @param flight_User $usr
 * @param $oid
 * @throws user_exception
 */

```

```

/*
public static function pay_for_orders(mysqli &$link, flight_User &$usr, $oid) {
    try {
        $succeeded = false;
        $select_order = "select ".config\Ticket_table::CANCELED . "," .
            config\Ticket_table::TOOKOFF_TIME .
            " from ". config\Ticket_table::NAME .
            " where ". config\Ticket_table::OID . "=" . $oid . ";";
        echo $select_order;
        $result_rows = $link->query($select_order);

        $satisfied = true;
        $hasthis = false;
        while (list($canceled, $offtime) = $result_rows->fetch_row()) {
            $hasthis = true;
            if ((bool)$canceled == true) {
                throw new user_exception(user_exception_codes::AlreadyCanceled);
            }
            else if (strtotime(config\BJ_time::get_current_datetime()) >
                strtotime($offtime)) {
                $satisfied = false;
            }
        }

        if (!$hasthis) {
            throw new user_exception(user_exception_codes::CouldNotFindOrder);
        }
        if (!$satisfied) {
            throw new user_exception(user_exception_codes::TooLatetoDo);
        }
        else {
            $select_order = "select ".config\Order_table::PAID . "," .
                config\Order_table::COST . " from ". config\Order_table::NAME .
                " where ". config\Order_table::ID . " = " . $oid . ";";
            $result_rows = $link->query($select_order);

            if (list($paid, $cost) = $result_rows->fetch_row()) {
                if ((bool)$paid == true) {
                    throw new user_exception(user_exception_codes::AlreadyPaid);
                }
                $cost2p = new decimal2P($cost);
                $ublance = new decimal2P($usr->getUBalance());
                if((bool)($ublance->compare($cost2p)) == false) {
                    throw new user_exception(user_exception_codes::NotEnoughBalance);
                }
                else {
                    $link->autocommit(false);
                    $update_query = "update ". config\Order_table::NAME .
                        " set ". config\Order_table::PAID . " = true" .
                        " where ". config\Order_table::ID . " = " . $oid . ";";

                    $link->query($update_query, MYSQLI_STORE_RESULT);

                    if ($link->affected_rows > 0) {
                        $usr->decBalance($cost);
                        $update_query = "update ". config\User_table::NAME .
                            " set ". config\User_table::BALANCE . " =
                            " . $usr->getUBalance() .
                            " where ". config\User_table::ID . " = " . $usr->UID . ";";
                        $link->query($update_query, MYSQLI_STORE_RESULT);

                        if ($link->affected_rows > 0) {
                            $link->commit();
                            $succeeded = true;
                        }
                        else {
                            $usr->incBalance($cost);
                            $link->rollback();
                            throw new user_exception(user_exception_codes::ServerBusy);
                        }
                    }
                    else {
                        $link->rollback();
                        throw new user_exception(user_exception_codes::ServerBusy);
                    }
                }
            }
            else {
                throw new user_exception(user_exception_codes::CouldNotFindOrder);
            }
        }
    }
}

```

```

        }
        catch (user_exception $ex) {
            throw $ex;
        }
        catch (mysqli_sql_exception $ex) {
            throw $ex;
        }
        catch (Exception $ex) {
            throw $ex;
        }
        finally {
            $link->autocommit(true);
        }
    }

    /**
     * add balance to account
     * @param mysqli $link
     * @param flight_User $usr
     * @param string $money      the money add to the account
     * @throws user_exception
     * Generate a new decimal2P object for money
     */
    public static function add_balance(mysqli &$link, flight_User &$usr, string $money) {
        // in UI, it needs user to choose, 100.00, 500.00, 1000.00, 5000.00
        try {
            $link->autocommit(false);
            $add_money = new decimal2P($money);
            $query = "update ". config\User_table::NAME .
                " set ". config\User_table::BALANCE ."=" .
config\User_table::BALANCE ."+" . $add_money .
                " where " . config\User_table::ID . "=" . $usr->UID . ";";
            $try_times = self::RETRY_TIMES;
            $succeed = false;

            do {
                $link->query($query, MYSQLI_STORE_RESULT);
                if ($link->affected_rows > 0) {
                    $succeed = true;
                    $link->commit();
                    break;
                }
                else {
                    $link->rollback();
                }
            }while($try_times--);

            if ($try_times == 0 && $succeed == false) {
                throw new user_exception(user_exception_codes::ServerBusy);
            }
            else {
                $usr->incBalance($money);
            }
        }
        catch (mysqli_sql_exception $ex) {
            throw $ex;
        }
        catch (user_exception $ex) {
            throw $ex;
        }
        catch (Exception $ex) {
            throw $ex;
        }
        finally {
            $link->autocommit(true);
        }
    }

    /**
     * Take ticket based on a specific tid
     * @param mysqli $link
     * @param flight_User $usr
     * @param $tid
     * @throws user_exception
     */
    public static function take_ticket(mysqli &$link, flight_User &$usr, $tid) {
        try {
            $try_times = self::RETRY_TIMES;
            $succeeded = false;

```

```

        do {
            $tic_search = "select " .
                config\Ticket_table::OID . "," . config\Ticket_table::CANCELED . "," .
                config\Ticket_table::GOT_TIME . ",";
            config\Ticket_table::TOOKOFF_TIME .
                " from ".config\Ticket_table::NAME . " where " . config\Ticket_table::ID .
                " = " . $tid . ";";
        //
            echo $tic_search;
            $tic_result = $link->query($tic_search);
        //
            if (list($oid, $canceled, $gott, $offt) = $tic_result->fetch_row()) {
                echo "$oid, $canceled, $gott, $offt <br />";
                if ((bool)$canceled == true) {
                    throw new user_exception(user_exception_codes::AlreadyCanceled);
                }
                else if ($gott != null) {
                    throw new user_exception(user_exception_codes::AlreadyGot);
                }
                else {
                    $cur_t_plus_24h = strtotime("+24 hour",
                        strtotime(config\BJ_time::get_current_datetime()));
                    if (strtotime($offt) > $cur_t_plus_24h) {
                        throw new user_exception(user_exception_codes::TooEarly);
                    }
                    else {
                        $order_search = "select ".config\Order_table::PAID .
                            " from ".config\Order_table::NAME." where ".
                            ".config\Order_table::ID." = ".$oid.";";
                    //
                        echo "$order_search <br />";
                        $result = $link->query($order_search);
                    //
                        list($paid) = $result->fetch_row();
                        echo "$paid";
                        if ((bool)$paid == true) {
                            // update ticket table
                            $link->autocommit(false);
                            $ctime = config\BJ_time::get_current_datetime();
                            $update_query = "update ".config\Ticket_table::NAME .
                                " set ".config\Ticket_table::GOT_TIME." = '$ctime' where " .
                                config\Ticket_table::ID." = ".$tid.";";
                            $link->query($update_query, MYSQLI_STORE_RESULT);
                            if ($link->affected_rows == 1) {
                                $succeeded = true;
                                $link->commit();
                                $link->autocommit(true);
                                break;
                            }
                            else {
                                $link->rollback();
                            }
                        }
                        else {
                            throw new user_exception(user_exception_codes::HaventPaid);
                        }
                    }
                }
            }
        //
            else {
                throw new user_exception(user_exception_codes::CouldNotFindTicket);
            }
        }
    while($try_times--);
    if ($try_times == 0 && !$succeeded) {
        $link->rollback();
        throw new user_exception(user_exception_codes::ServerBusy);
    }
}
catch (mysqli_sql_exception $ex) {
    throw $ex;
}
catch (user_exception $ex) {
    echo $ex;
    throw $ex;
}
finally {
    $link->autocommit(true);
}
}

```

```

/**
 * CANCEL TICKETS
 * @param mysqli $link
 * @param flight_User $usr
 * @param $tid
 * @throws user_exception
 */
public static function cancel_ticket(mysqli &$link, flight_User &$usr, $tid) {
    try {
        $try_times = self::RETRY_TIMES;
        $succeeded = false;

        do {
            $tic_search = "select
".config\Ticket_table::OID.", ".config\Ticket_table::CANCELED.", ".config\Ticket_table::
GOT_TIME.", "
                .config\Ticket_table::TOOKOFF_TIME.", ".config\Ticket_table::FID.", ".c
onfig\Ticket_table::SEATCLAS.", ".config\Ticket_table::PRICE.
                " from ".config\Ticket_table::NAME." where ".config\Ticket_table::ID." =
$tid;";
        // echo $tic_search;
        $tic_result = $link->query($tic_search);

        if (list($oid, $canceled, $gtime, $offtime, $fid, $seat_class, $price) =
$tic_result->fetch_row()) {
            // echo "$oid, $canceled, $gtime, $offtime, $fid, $seat_class, $price <br
/>";
            $curtime = config\BJ_time::get_current_datetime();
            if ($gtime != null) {
                throw new user_exception(user_exception_codes::AlreadyGot);
            }
            if(strtotime($curtime) > strtotime($offtime)) {
                throw new user_exception(user_exception_codes::TooLatetoDo);
            }
            if ((bool)$canceled == true) {
                throw new user_exception(user_exception_codes::AlreadyCanceled);
            }
            else {
                $query_paid = "select ".config\Order_table::PAID." from ".
                    config\Order_table::NAME." where ".config\Order_table::ID." =
$oid;";
                $paid_result = $link->query($query_paid);

                if (list($alpaid) = $paid_result->fetch_row()) {
                    $link->autocommit(false);
                    $off_date = date("Y-m-d", strtotime( $offtime));
                    $update_seats = "";
                    if ($seat_class == 'E') {
                        $update_seats = "update Flying_date set e_taken = e_taken - 1 where
f_date = '2019-08-31' and f_FID = 351;" ;
                        $update_seats = "update ".config\Flying_date_table::NAME." set
".config\Flying_date_table::ETAKEN." = ".config\Flying_date_table::ETAKEN." - 1".
                            " where f_date = '$off_date' and f_FID = '$fid';";
                    }
                    else if ($seat_class == 'C') {
                        $update_seats = "update ".config\Flying_date_table::NAME." set
".config\Flying_date_table::CTAKEN." = ".config\Flying_date_table::CTAKEN." - 1".
                            " where f_date = '$off_date' and f_FID = '$fid';";
                    }
                    else {
                        $update_seats = "update ".config\Flying_date_table::NAME." set
".config\Flying_date_table::FTAKEN." = ".config\Flying_date_table::FTAKEN." - 1".
                            " where f_date = '$off_date' and f_FID = '$fid';";
                    }
                    $link->query($update_seats, MYSQLI_STORE_RESULT);
                    if ($link->affected_rows > 0) {
                        $reg_cancel = "update ".config\Ticket_table::NAME.
                            " set ".config\Ticket_table::CANCELED." = true where
".config\Ticket_table::ID." = $tid;";
                        $link->query($reg_cancel, MYSQLI_STORE_RESULT);
                        echo $reg_cancel . "<br />";
                        if ($link->affected_rows > 0) {
                            if ((bool)$alpaid == true) {
                                $refund = "update ".config\User_table::NAME.
                                    " set ".config\User_table::BALANCE." =
".config\User_table::BALANCE.
$usr->UID;";
                            }
                            echo $refund . "<br />";
                            $link->query($refund, MYSQLI_STORE_RESULT);
                        }
                    }
                }
            }
        }
    }
}

```

```

                if ($link->affected_rows > 0) {
                    $link->commit();
                    break;
                }
            }
        }
    }
}
else {
    $link->commit();
    break;
}
}
}
}
}
else {
    throw new user_exception(user_exception_codes::CouldNotFindTicket);
}
}
else {
    throw new user_exception(user_exception_codes::CouldNotFindTicket);
}
}

while($try_times--);
if ($try_times == 0 && !$succeeded) {
    $link->rollback();
    throw new user_exception(user_exception_codes::ServerBusy);
}
catch (mysqli_sql_exception $ex) {
    throw $ex;
}
catch (user_exception $ex) {
    throw $ex;
}
finally {
    $link->autocommit(true);
}
}

/**
 * look up the history of ticket and order, using join ticket and order to show
 * @param mysqli $link
 * @param flight_User $usr
 * @return array
 * @throws user_exception
 */
public static function lookup_history(mysqli &$link, flight_User &$usr) {
    try {
        $query = "select ".
config\Order_table::ID.", ".config\Order_table::UID.", ".config\Order_table::TIME.", ".
config\Order_table::PAID.", ".config\Order_table::COST.", ".
config\Ticket_table::ID.", ".config\Ticket_table::CANCELED.", ".
config\Ticket_table::GOT_TIME.", ".config\Ticket_table::TOOKOFF_TIME.", ".
config\Ticket_table::FID.", ".config\Ticket_table::SEATCLAS.", ".config\Ticket_table::PR
ICE." from ".
            config\Order_table::NAME." join ".config\Ticket_table::NAME." on ".
            config\Order_table::NAME.".config\Order_table::ID." =
".config\Ticket_table::NAME.".config\Ticket_table::OID.
            " and ".config\Order_table::UID." = ".$usr->UID.";";

        $result = $link->query($query);
        $ret = array();
        while ($temp = $result->fetch_row()) {$ret[] = $temp;}
        //        while(list($oid, $uid, $otime, $opaid, $ocost, $tid, $tcanceled, $tgottime,
        //        $tofftime,
        //        $fid, $seatclass, $tprice) = $result->fetch_row()) {
        //
        //}
        $result->free();
        //        var_dump($ret);
        return $ret;
    }
    catch (mysqli_sql_exception $ex) {
        throw $ex;
    }
    catch (user_exception $ex) {
        throw $ex;
    }
    catch (Exception $ex) {

```

```

        throw $ex;
    }
    finally {
        $link->autocommit(true);
    }
}

public static function get_flying_time(mysqli &$link, $fid) {
    try {
        $query = "select ".config\Flight_table::DEPART_TIME." from
".config\Flight_table::NAME.
        " where ".config\Flight_table::ID." = $fid;";
        $ret = $link->query($query);
        list($offtime) = $ret->fetch_row();
        return $offtime;
    }
    catch (Exception $ex) {
        throw $ex;
    }
}
}

```

Admin_functions.php 定义并实现了一些管理员操作的接口。

```

<?php

include_once '../common/decimal2P.php';
include_once '../common/config.php';
include_once '../common/DBConnector.php';
include_once '../common/Flight.php';

error_reporting(E_ALL | E_STRICT);

class admin_exception_codes {
    public const UNKNOWN = 0;
    public const FIDNotNumeric = 1;
    public const PlaceNotValid = 2;
    public const SeatsNotNumeric = 3;
    public const FIDAlreadyExist = 4;
    public const InsertFlightFailed = 5;
    public const InvalidSeatsParam = 6;
    public const AddFlightDateFailed = 6;
    public const TimeLogicError = 7;
    public const DiscountNotNumeric = 8;
    public const NoSuchFlight = 9;
}

class admin_exception extends Exception {
    public $code;

    function __construct($code = 0) {
        parent::__construct("", $code, null);
    }

    function __toString() {
        switch ($this->code) {
            case admin_exception_codes::UNKNOWN:
                return "Sorry, unknown exception";
            case admin_exception_codes::FIDNotNumeric:
                return "Format exception! FID is not numeric!";
            case admin_exception_codes::PlaceNotValid:
                return "Place is not valid!";
            case admin_exception_codes::SeatsNotNumeric:
                return "Seats not numeric!";
            case admin_exception_codes::FIDAlreadyExist:
                return "This FID already exists";
            case admin_exception_codes::InsertFlightFailed:
                return "Server is busy, or some input error occurred.";
            case admin_exception_codes::InvalidSeatsParam:
                return "Incorrect Format";
            case admin_exception_codes::AddFlightDateFailed:
                return "Failed to add flight date.";
            case admin_exception_codes::TimeLogicError:
                return "Time conflicts";
            case admin_exception_codes::DiscountNotNumeric:
                return "Discount is not numeric";
            case admin_exception_codes::NoSuchFlight:
                return "No such flight";
            default:
                return "Some admin exception occurred.";
        }
    }
}

```

```

        }
    }

class admin_functions {
    public $airports;
    private $conn;
    private const RETRY_TIMES = 5;

    /**
     * admin_functions constructor.
     * Be careful that the constructor should create a DBConnector
     * and create airports codes' array
     */
    function __construct() {
        $this->conn = new DBConnector(true);
        $this->airports = $this->conn->get_all_airports_code();
    }

    /**
     * PAY ATTENTION to ERROR HANDLE and INPUT CHECK
     * @params basic information about flight
     * this should always create seats for flight
     * @throws admin_exception
     */
    function add_flight($fid, $f_type, $depart_time,
                        $duration, $depart_place, $arrive_place,
                        $begin_service_date, $end_service_date,
                        $fnum, $enum, $cnum) {
        $new_flight = new Flight($fid, $f_type, $depart_time,
                                $duration, $depart_place, $arrive_place,
                                $begin_service_date, $end_service_date, $fnum, $enum, $cnum);

        /* input parameters tests */
        try {
            if (!is_numeric($fid)) {
                throw new admin_exception(admin_exception_codes::FIDNotNumeric);
            }
            else if (!in_array($depart_place, $this->airports) || !in_array($arrive_place, $this->airports)) {
                throw new admin_exception(admin_exception_codes::PlaceNotValid);
            }
            else if (!is_numeric($fnum) || !is_numeric($enum) || !is_numeric($cnum)) {
                throw new admin_exception(admin_exception_codes::SeatsNotNumeric);
            }
            else {
                $search_fid = "select " . config\Flight_table::ID .
                    " from " . config\Flight_table::NAME .
                    " where " . config\Flight_table::ID . " = " . $fid;
                $result = $this->conn->link->query($search_fid);
                if ($result->fetch_row() != null) {
                    throw new admin_exception(admin_exception_codes::FIDAlreadyExist);
                }
                $retry_times = self::RETRY_TIMES;
                $succeeded = false;

                do {
                    $this->conn->link->autocommit(false);
                    $insert_flight_query = "insert into " . config\Flight_table::NAME . " "
                    values(" .
                        $new_flight. ")");
                    $this->conn->link->query($insert_flight_query, MYSQLI_STORE_RESULT);
                    $insert_flight_result = $this->conn->link->affected_rows;

                    if ($insert_flight_result > 0) {
                        $this->conn->link->commit();
                        $succeeded = true;
                        break;
                    }
                    else {
                        $this->conn->link->rollback();
                    }
                }while(--$retry_times);
                if ($retry_times == 0 && $succeeded == false) {
                    throw new admin_exception(admin_exception_codes::InsertFlightFailed);
                }
            }
        }
        catch (mysqli_sql_exception $ex) {

```

```

//           echo $ex . "<br />";
//           throw $ex;
}
catch (admin_exception $ex) {
    throw $ex;
}
finally {
    // Ensure that autocommit should be reopen
    $this->conn->link->autocommit(true);
}

/**
 * @param $fid          : corresponding to the flight
 * @param string $begin_date: day begin
 * @param string $end_date : day end
 * @param int $interval_days: number of days between flights
 * @param int $edis       : economic seats' discount
 * @param int $cdis       : commercial seats' discount
 * @param int $fdis       : first-class seats' discount
 * @throws admin_exception
*/
function add_flying_date($fid, string $begin_date,
                        string $end_date, int $interval_days, int $edis, int $cdis, int
$fdis,
                        string $E_seat_price, string $C_seat_price, string $F_seat_price)
{
    try {
        $eprice = new decimal2P($E_seat_price);
        $cprice = new decimal2P($C_seat_price);
        $fprice = new decimal2P($F_seat_price);
        $search_fid = "select ". config\Flight_table::ID . "," .
config\Flight_table::BEGIN_SERVICE
            . "," . config\Flight_table::END_SERVICE .
            " from ". config\Flight_table::NAME .
            " where ". config\Flight_table::ID . " = " . $fid . ";";

        $result = $this->conn->link->query($search_fid);
        list($rfid, $rbtime, $retime) = $result->fetch_row();
        echo $search_fid . "<br />";
        if ($rfid == null) {
            throw new admin_exception(admin_exception_codes::FIDAlreadyExist);
        }
        else if(strtotime($begin_date) < strtotime($rbtime) || strtotime($end_date) >
strtotime($retime)) {
            throw new admin_exception(admin_exception_codes::TimeLogicError);
        }
        else if(!is_numeric($edis) || !is_numeric($cdis) || !is_numeric($fdis)) {
            throw new admin_exception(admin_exception_codes::DiscountNotNumeric);
        }
        else if ((($eprice->showMoney() == null) || ($cprice->showMoney() == null) ||
($fprice->showMoney() == null))) {
            throw new admin_exception(admin_exception_codes::InvalidSeatsParam);
        }
        $retry_times = self::RETRY_TIMES;
        $succeeded = false;
        do {
            $cnt = 0;
            for ($t = date($begin_date); $t <= date($end_date); ) {
                $insert_flying_date = "insert into ". config\Flying_date_table::NAME .
" values(" .
                    "" . date("Y-m-d", strtotime($t)). "," . $fid . "," . $edis . "," . $cdis .
"," . $fdis .
                    "" . $eprice . "," . $cprice . "," . $fprice . ",0,0,0,0.00);";
//                echo $insert_flying_date . "<br />";
                $this->conn->link->query($insert_flying_date, MYSQLI_STORE_RESULT);
                $cnt += $this->conn->link->affected_rows;
                $t = date("Y-m-d", strtotime("+$interval_days day", strtotime($t)));
            }
            if ($cnt > 0) {
                $succeeded = true;
                break;
            }
        }while($retry_times--);
        if ($retry_times == 0 && $succeeded == false) {
            throw new admin_exception(admin_exception_codes::InsertFlightFailed);
        }
    }
    catch (mysqli_sql_exception $ex) {
        throw $ex;
    }
}

```

```

        catch (admin_exception $ex) {
            throw $ex;
        }
    }

    function list_data() {
        try {
            $query = "select * from ".config\Views::DATA_SEL . ";";
            $result = $this->conn->link->query($query);
            $ret = array();
            while (list($f_date, $fid, $etaken, $ctaken, $ftaken, $revenue, $fn, $en, $cn)
= $result->fetch_row()) {
                $e_taken_rate = round((int)$etaken * 1.0 / (int)$en, 2);
                $c_taken_rate = round((int)$ctaken * 1.0 / (int)$cn, 2);
                $f_taken_rate = round((int)$ftaken * 1.0 / (int)$fn, 2);
                $ret[] = array($f_date, $fid, $etaken, $ctaken, $ftaken, $revenue, $fn, $en,
$cn, $e_taken_rate, $c_taken_rate, $f_taken_rate);
            }
            $result->free();
            return $ret;
        }
        catch (mysqli_sql_exception $ex) {
            throw $ex;
        }
        catch (Exception $ex) {
            throw $ex;
        }
    }

    function show_revenue_by_month(array $data) {
        if ($data == array()) {
            return ;
        }
        $output = array();
        $month_record = array();
        for ($i = 0; $i < count($data); $i++) {
            list($f_date, $fid, $etaken, $ctaken, $ftaken, $revenue, $fn, $en, $cn,
$e_taken_rate, $c_taken_rate, $f_taken_rate)
= $data[$i];
            $month = date("Y-m", strtotime($f_date));
            if (array_key_exists($month, $month_record)) {
                $month_record[$month] += $revenue;
            }
            else {
                $month_record[$month] = 0;
            }
        }
        return $month_record;
    }
}

```