# DB-m12306

Database lab2: 火车订票系统模拟.



---


- 1. 综述
  - 1.1. 文件夹结构
  - 1.2. 关键代码及功能对应
- 2. 数据库逻辑
  - 2.1. 数据库应用整体实现
  - 2.2. ER图
  - 2.3. 关系模式
    - 2.3.1. ID_Station_City 车站城市对照表ISC_
    - 2.3.2. Train 列车T_
    - 2.3.3. Train_Table 列车时刻表TT_
    - 2.3.4. Empty_Seat 可用座位表ES_
    - 2.3.5. Passenger 乘客P_
    - 2.3.6. Orders 订单O_
    - 2.3.7. Station_Connection 车站联通表SC_
    - 2.3.8. City_Connection 城市联通表CC_
  - 2.4. 范式细化，分析
    - 2.4.1. 第一范式
    - 2.4.2. 第二范式
    - 2.4.3. 第三范式
      - 2.4.3.1. BCNF范式
- 3. 查询与刷新函数
  - 3.1. 特殊类型定义
  - 3.2. SQL查询语句模板
    - 3.2.1. 记录乘客信息
      - 3.2.1.1. 乘客注册时手机号重复检查
      - 3.2.1.2. 乘客注册时身份证重复检查
    - 3.2.2. 显示乘客相关信息
    - 3.2.3. 查询具体车次
      - 3.2.3.1. 查询某一车次从起始站到终点站的全部信息
    - 3.2.4. 查询两地之间的车次
      - 3.2.4.1. 查询两地之间的车次及余票信息(带换乘)
    - 3.2.5. 预订车次座位
      - 3.2.5.1. 直接生成新订单号
    - 3.2.6. 查询订单和删除订单
      - 3.2.6.1. 查询某个时间范围内的订单
    - 3.2.7. 管理员
      - 3.2.7.1. 总订单数
      - 3.2.7.2. 总票价
    - 3.2.8. 最热点车次排序，排名前10的车次
    - 3.2.9. 当前注册用户列表
    - 3.2.10. 查看每个用户的订单

    - 3.3. SQL刷新语句及调用地点
        - 3.3.1. 创建订单并扣除余票
        - 3.3.2. 取消订单与恢复余票
        - 3.3.3. 开放新一天的购票
- 4. 数据库系统实现
    - 4.1. 数据导入
    - 4.2. 余票查询
    - 4.3. LIMIT 语句的引入
    - 4.4. 其他查询语句
- 5. 前端实现
- 6. ACKNOWLEDGE


# 1. 综述

在此项目中, 我们完成了实验二要求的全部内容: 搭建一个类似于12306的应用, 支持用户登录, 管理员登陆, 订票, 余票查询等功能.

技术实现采用python脚本进行数据预处理, psql搭建数据库, php实现前端代码.

小组分工如下:

- 前端实现: 刘蕴哲
- 换乘查询, ER图及范式分析: 蔡昕
- 数据预处理, 数据库架构, 数据库查询语句, 报告撰写: 王华强

## 1.1. 文件夹结构

```
-data                   原始数据以及经过预处理的数据
-dest                   最终生成的数据库应用及sql源码
-dev                    开发过程中使用的脚本
-doc                    文档
-src                    开发过程中的全部源码
-src-database           数据库源码
-src-database-setup_db  预处理及建库源码
-src-database-model     数据库操作语句源码
-src-php                php源码
-web                    网站建设参考代码
-playground             测试目录
```

## 1.2. 关键代码及功能对应

```
create_database.py      数据预处理
declare.sql             建立数据库, 各表定义
seats.sql               余座表初始化(处理随时间推移时, 新的列车开放订票的情况)
user.sql                处理用户相关的逻辑
admin.sql               处理管理员相关的逻辑
transfer_withleft.sql   处理换乘以及余座计算
orders.sql              生成/取消订单
```

---

# 2. 数据库逻辑

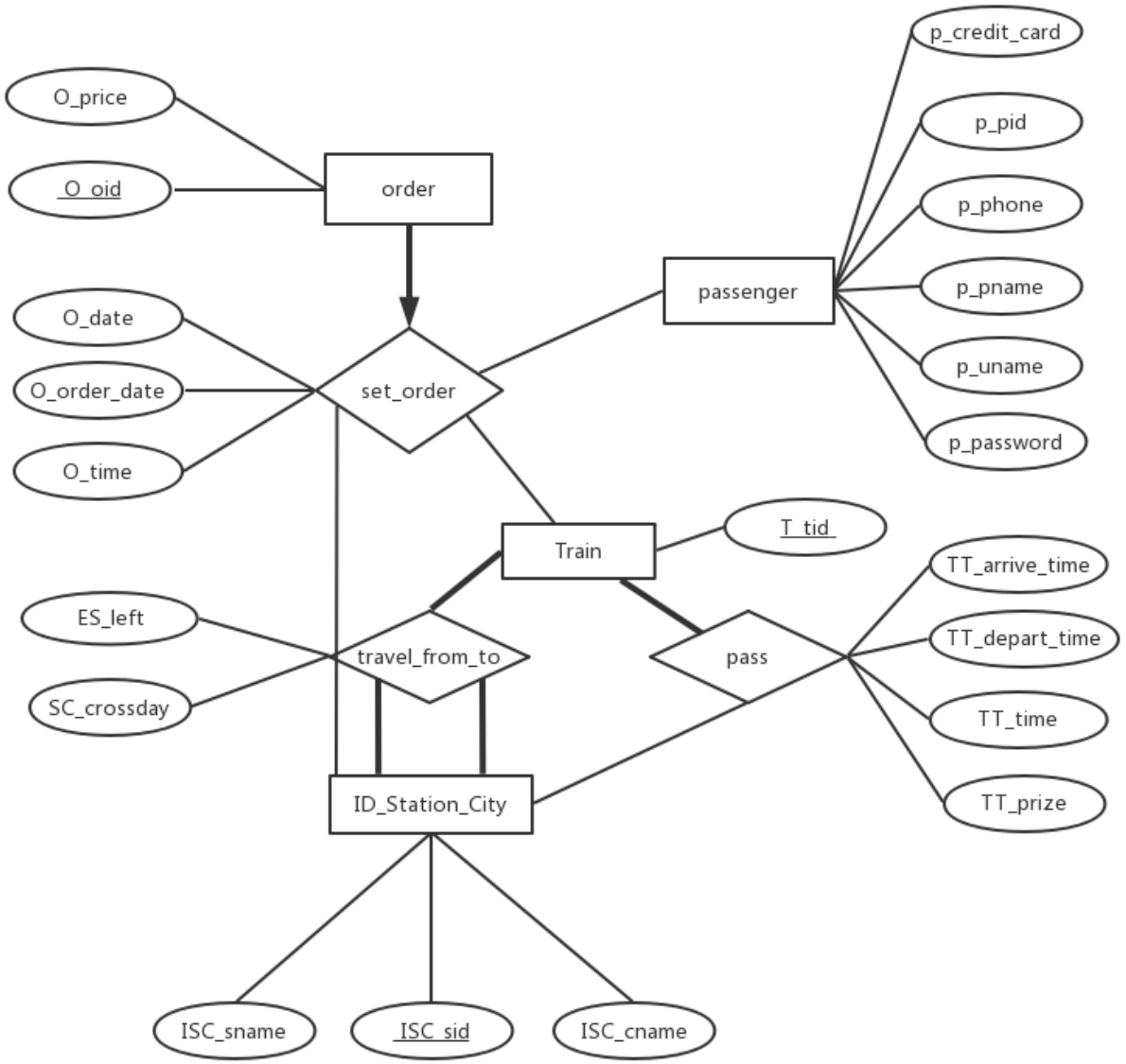
## 2.1. 数据库应用整体实现

数据库系统的实现首先依赖于数据的导入. 在本实验中数据的预处理使用python脚本解析正则表达式的形式, 将数据转化成规整的CSV文件. 在预处理过程中同时对数据中的缺失项做了处理, 同时根据换乘查询的需要, 计算生成了城市间的连通图表和车站之间的连通图表, 希望通过这些数据的预处理来提高查询速度, 降低查询难度.

基本的数据库查询语句略过不表. 查询语句的难点集中在换乘计算的消耗上. 通过在预处理中预先建表来降低计算复杂度, 以及引入`LIMIT`来降低计算量的方式, 我们将换乘查询的时间消耗控制在了可以接受的量级.

在前端通过php语句调用sql查询语句来访问数据库. 尽管从软件工程的角度上来讲, 应该使用MVC模型为佳. 但是考虑到小组成员都是第一次接触PHP, 因此没有采用这些面向对象的软件开发模型.

## 2.2. ER图




## 2.3. 关系模式

具体到代码级别的定义, 可以参见`declare.sql`

Table Layouts 如下:

### 2.3.1. ID_Station_City 车站城市对照表ISC_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| ISC_sid | INTEGER | not null, primary key | |
| ISC_sname | CHAR(20) | not null | |
| ISC_cname | CHAR(20) | not null | |

### 2.3.2. Train 列车T_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| T_tid | 车次号 | char(10) | identifier, not null |
| T_start_sid | 始发站id | int | foreign key references ID_Station_City(ISC_sid) |
| T_end_sid | 终点站id | int | foreign key references ID_Station_City(ISC_sid) |

### 2.3.3. Train_Table 列车时刻表TT_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| TT_tid | 车次号 | char(10) | identifier |
| TT_sid | 车站id | int | identifier |
| TT_depart_time | 发车时间 | date | |
| TT_arrive_time | 到达时间 | date | |
| TT_price_yz | 硬座票价 | decimal | |
| TT_price_rz | 软座票价 | decimal | |
| TT_price_yws | 硬卧上铺票价 | decimal | |
| TT_price_ywz | 硬卧中铺票价 | decimal | |
| TT_price_ywx | 硬卧下铺票价 | decimal | |
| TT_price_rws | 软卧上铺票价 | decimal | |
| TT_price_rwx | 软卧下铺票价 | decimal | |
| TT_count | 当前站是列车的第几站 | int | 用于换乘计算 |

### 2.3.4. Empty_Seat 可用座位表ES_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| ES_tid | 车次号 | char(10) | identifier |
| ES_current_sid | 当前车站id | int | identifier |

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| ES_date | 日期 | date | identifier |
| ES_left_yz | 硬座剩余座位数 | int | 初始化时置为5 |
| ES_left_rz | 软座剩余座位数 | int | 初始化时置为5 |
| ES_left_yws | 硬卧上铺剩余座位数 | int | 初始化时置为5 |
| ES_left_ywz | 硬卧中铺剩余座位数 | int | 初始化时置为5 |
| ES_left_ywx | 硬卧下铺剩余座位数 | int | 初始化时置为5 |
| ES_left_rws | 软卧上铺剩余座位数 | int | 初始化时置为5 |
| ES_left_rwx | 软卧下铺剩余座位数 | int | 初始化时置为5 |

注意: 终点站的剩余座位数无用

## 2.3.5. Passenger 乘客P_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| P_pid | 身份证号 | bigint | primary key, unique |
| P_phone | 手机号 | bigint | unique |
| P_pname | 姓名 | char(20) | |
| P_uname | 用户名 | char(30) | |
| P_credit_card | 信用卡 | int | |
| P_password | 密码char(30) | | |

## 2.3.6. Orders 订单O_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| O_oid | 订单号 | int | identifier |
| O_pid | 身份证号 | int | |
| O_order_date | 订单日期 | date | not null default CURRENT_DATE |
| O_tid | 车次序号 | char(10) | |
| O_start_sid | 始发站 | int | |
| O_arrive_sid | 到达站 | int | |
| O_price | 总票价 | decimal | |
| O_date1 | 第一列车的日期 | date | not null |
| O_time1 | 第一列车的时间 | time | not null |
| O_tid1 | 第一列车id | char(10) | not null |
| O_start_sid1 | 第一列车的始发站 | int | not null |

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| O_arrive_sid1 | 第一列车的终到站 | int | not null |
| O_date2 | 第二列车的日期 | date | |
| O_time2 | 第二列车的时间 | time | |
| O_tid2 | 第二列车id | char(10) | |
| O_start_sid2 | 第二列车的始发站 | int | |
| O_arrive_sid2 | 第二列车的终到站 | int | |
| O_valid | 订单有效 | int | default 1 |

附注: 在设计的过程中这个表支持同时包含两次列车(换乘), 但在实际设计中为了操作便利, 在一个订单中只包含一列车, 即一次换乘需要产生两个订单.

### 2.3.7. Station_Connection 车站联通表SC_

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| SC_depart_sid | 出发车站id | int | identifier, foreign key references ID_Station_City(ISC_sid) |
| SC_arrive_sid | 到达车站id | int | identifier, foreign key references ID_Station_City(ISC_sid) |
| SC_tid | 列车号 | int | identifier, foreign key references Train(T_tid |
| SC_crossday | 列车开行过程中跨日情况记录 | int | default 0 |

### 2.3.8. City_Connection 城市联通表CC_

注: 此表用在某个换乘查询算法中. 最终实现的数据库不需要此表. 为了演示多种不同的查询方法, 故保留.

| 列名 | 内容 | 数据种类 | 附注 |
| --- | --- | --- | --- |
| CC_depart_city | 出发城市 | char(20) | identifier |
| CC_arrive_city | 到达城市 | char(20) | identifier |
| CC_tid | 列车号 | int | identifier |
| CC_crossday | 列车开行过程中跨日情况记录 | int | default 0 |

---

## 2.4. 范式细化，分析

> ref: https://www.ibm.com/developerworks/cn/data/library/techarticles/dm-0605jiangt/index.html

本数据库实现满足第二范式的要求, 经少量改动可以满足第三范式的要求.

### 2.4.1. 第一范式

1. 因为这张表中字段都是单一属性的，不可再分；

2. 而且每一行的记录都是没有重复的；
3. 存在主属性，而且所有的属性都是依赖于主属性；
4. 所有的主属性都已经定义

> 事实上在当前所有的关系数据库管理系统（DBMS）中，都已经在建表的时候强制满足第一范式。

## 2.4.2. 第二范式

ref: https://zhuanlan.zhihu.com/p/20028672

> 根据2NF的定义，判断的依据实际上就是看数据表中是否存在非主属性对于码的部分函数依赖

消除部分依赖: 本数据库系统中通过将Users实体, City-ID-Station关系单独建表来防止部分依赖.

列出各个表中的主键, 主属性, 非主属性如下, 这里只列出主属性不唯一的情况:

Train_Table:

- TT_tid 主属性
- TT_sid 主属性
- TT_depart_time
- TT_arrive_time
- TT_price_yz
- TT_price_rz
- TT_price_yws
- TT_price_ywz
- TT_price_ywx
- TT_price_rws
- TT_price_rwx
- TT_count

非主属性由TT_tid与TT_sid联合确定, 不存在部分函数依赖关系.

Empty_Seat:

- ES_tid
- ES_current_sid
- ES_date
- ES_left_yz
- ES_left_rz
- ES_left_yws
- ES_left_ywz
- ES_left_ywx
- ES_left_rws
- ES_left_rwx

非主属性由ES_yid与ES_current_sid联合确定, 不存在部分函数依赖关系.

Station_Connection:

- SC_depart_sid 主属性
- SC_arrive_sid 主属性
- SC_tid
- SC_crossday

站之间的联通情况需要由出发站和到达站确定, 因此不存在部分函数依赖.

综上, 不存在非主属性对主键的部分函数依赖, 数据库满足第二范式.

## 2.4.3. 第三范式

观察是否存在非主属性对于键的传递函数依赖:

ID_Station_City:

存在传递函数依赖: ISC_sid->ISC_sname->ISC_cname

因为已知站名可以推出对应的城市名.

如果要使得数据库满足第三范式, 需要将ID_Station_City表拆分为:

table1:

- ISC_sid(主键)
- ISC_sname

table2:

- ISC_sname(主键)
- ISC_cname

Train:

不存在传递函数依赖

Train_Table:

不存在传递函数依赖

Empty_Seat:

不存在传递函数依赖

Passenger:

- P_pid 主属性
- P_phone 可以作为主键: 主属性
- P_pname
- P_uname
- P_credit_card
- P_password

余下部分不存在传递函数依赖

Orders:

为了支持单订单多列车, 存在少量数据冗余, 但是在一个订单中只有单列车的情况下基本可以不考虑.

Station_Connection:

不存在传递函数依赖

### 2.4.3.1. BCNF范式

存在着主属性对于候选键的部分函数依赖与传递函数依赖. 因此不符合BCNF范式.

比如Passenger表中的P_pid与P_phone属性.

---

# 3. 查询与刷新函数

## 3.1. 特殊类型定义

座位枚举类型:

```
CREATE TYPE seat_type AS ENUM ('yz', 'rz', 'yws','ywz','ywx','rws','rwz','rwx');
```

## 3.2. SQL查询语句模板

### 3.2.1. 记录乘客信息

#### 3.2.1.1. 乘客注册时手机号重复检查

```
SELECT count(*)
FROM Passenger
WHERE P_phone=$phone;
```

#### 3.2.1.2. 乘客注册时身份证重复检查

```
SELECT count(*)
FROM Passenger
WHERE P_pid=$pid;
```

### 3.2.2. 显示乘客相关信息

```
SELECT
    P_pid,
    P_phone,
    P_pname,
    P_uname,
    P_credit_card
FROM Passenger
WHERE P_pid=[pid];
```

参数:

1. [PID] 用户id

### 3.2.3. 查询具体车次

#### 3.2.3.1. 查询某一车次从起始站到终点站的全部信息

```
-- 一列车的所有余票
SELECT
    RES.TT_count,
    RES.TT_tid,
    ISC_sname,
    RES.TT_sid,
    RES.TT_arrive_time,
    RES.TT_depart_time,
    RES.TT_price_yz,
    min(ES_left_yz) as left_yz
FROM Empty_Seat,
    Train_Table as TT1,
    Train_Table as TT2,
    Train_Table as TT3,
    ID_Station_City,
    Train,
    (
        SELECT
            TT_count, TT_tid, TT_sid, TT_arrive_time, TT_depart_time, TT_price_yz
        FROM Train_Table
        WHERE TT_tid=[TID]
        ORDER BY TT_count
    ) as RES
WHERE
    ES_tid=[TID] and
    ES_tid=TT1.TT_tid and
    ES_tid=TT2.TT_tid and
    ES_tid=TT3.TT_tid and
    T_tid=ES_tid and
    ISC_sid=RES.TT_sid and
    ES_date=[DATE] and
    (
        (
            TT1.TT_sid=T_start_sid and
            TT2.TT_sid=RES.TT_sid and
            TT3.TT_sid=ES_current_sid and
            (TT3.TT_count>=TT1.TT_count) and
            (TT3.TT_count<TT2.TT_count)
        )
        OR
        (
            TT1.TT_sid=T_start_sid and
            TT2.TT_sid=T_start_sid and
            TT3.TT_sid=T_start_sid and
            ES_current_sid=T_start_sid
        )
    )
GROUP BY
    RES.TT_count,
    RES.TT_tid,
    ISC_sname,
    RES.TT_sid,
    RES.TT_arrive_time,
    RES.TT_depart_time,
    RES.TT_price_yz
ORDER BY RES.TT_count
;
```

参数:

1. [TID] 列车id
2. [DATE] 要查询的日期

## 3.2.4. 查询两地之间的车次

#### 3.2.4.1. 查询两地之间的车次及余票信息(带换乘)

```
SELECT
    RES.depart_station,
    RES.transfer_station1,
    RES.transfer_station2,
    RES.arrive_station,
    RES.transfer_city,
    RES.tid1,
    RES.tid2,
    RES.price1,
    RES.price2,
    RES.price_total,
    min(ES1.ES_left_yz) as left_yz1,
    min(ES2.ES_left_yz) as left_yz2

FROM
    Empty_Seat as ES1,
    Empty_Seat as ES2,
    Train_Table as TT1,
    Train_Table as TT2,
    Train_Table as TT3,
    Train_Table as TT4,
    Train_Table as TT5,
    Train_Table as TT6,
    (


SELECT
    SC1.SC_depart_sid as depart_station,
    SC1.SC_arrive_sid as transfer_station1,
    SC2.SC_depart_sid as transfer_station2,
    SC2.SC_arrive_sid as arrive_station,
    SCI3.ISC_cname as transfer_city,
    SC1.SC_tid as tid1,
    SC2.SC_tid as tid2,
    TT2.TT_price_yz-TT1.TT_price_yz as price1,
    TT4.TT_price_yz-TT3.TT_price_yz as price2,
    TT4.TT_price_yz-TT3.TT_price_yz+TT2.TT_price_yz-TT1.TT_price_yz as price_total
FROM
    Station_Connection as SC1,
    Station_Connection as SC2,
    ID_Station_City as SCI1,--始发站
    ID_Station_City as SCI2,--终到站
    ID_Station_City as SCI3, --换乘下车站
    ID_Station_City as SCI4, --换乘上车站
    Train_Table as TT1,--始发站
    Train_Table as TT2,--终到站
    Train_Table as TT3,--换乘下车站
    Train_Table as TT4,--换乘上车站
    Train as T1,--第一次列车
    Train as T2 --第二次列车
WHERE
    --始发终到站
    SCI1.ISC_cname=[CITY1] and
    SCI2.ISC_cname=[CITY2] and
    SCI1.ISC_sid=SC1.SC_depart_sid and
    SCI2.ISC_sid=SC2.SC_arrive_sid
    and
    --基本换乘逻辑
    -- SC1.SC_depart_sid=443 and
    -- SC1.SC_arrive_sid=SC2.SC_depart_sid and
    SCI3.ISC_cname=SCI4.ISC_cname and
    SCI3.ISC_sid=SC1.SC_arrive_sid and
    SCI4.ISC_sid=SC2.SC_depart_sid and
    -- SC2.SC_arrive_sid=59 and
    SC1.SC_tid!=SC2.SC_tid
    -- and
    and
```

```
and
--禁止始发站/终到站同城市换乘
SCI1.ISC_cname!=SCI3.ISC_cname and
SCI2.ISC_cname!=SCI4.ISC_cname
and
--连接对应车次
TT1.TT_tid=SC1.SC_tid and
TT2.TT_tid=SC1.SC_tid and
TT3.TT_tid=SC2.SC_tid and
TT4.TT_tid=SC2.SC_tid and
TT1.TT_sid=SC1.SC_depart_sid and
TT2.TT_sid=SC1.SC_arrive_sid and
TT3.TT_sid=SC2.SC_depart_sid and
TT4.TT_sid=SC2.SC_arrive_sid
and
--换乘时间条件
-- TT3.TT_depart_time-TT2.TT_arrive_time
(
    (
        (
            SC1.SC_arrive_sid=SC2.SC_depart_sid --同站换乘
        )
        and
        (
            (
                (interval '60 min'<TT3.TT_depart_time-TT2.TT_arrive_time) and
                (interval '240 min'>TT3.TT_depart_time-TT2.TT_arrive_time)
            )
            or
            (
                (interval '60 min'<interval '24 hour'+TT3.TT_depart_time-TT2.TT_arrive_time) and
                (interval '240 min'>interval '24 hour'+TT3.TT_depart_time-TT2.TT_arrive_time)
            )
        )
    )
    OR
    (
        (
            SC1.SC_arrive_sid!=SC2.SC_depart_sid --异站换乘
        )
        and
        (
            (
                (interval '120 min'<TT3.TT_depart_time-TT2.TT_arrive_time) and
                (interval '240 min'>TT3.TT_depart_time-TT2.TT_arrive_time)
            )
            or
            (
                (interval '120 min'<interval '24 hour'+TT3.TT_depart_time-TT2.TT_arrive_time) and
                (interval '240 min'>interval '24 hour'+TT3.TT_depart_time-TT2.TT_arrive_time)
            )
        )
    )
)
--不可上车/下车站判断
and
(
    T1.T_tid=SC1.SC_tid and
    T2.T_tid=SC2.SC_tid
)
and
(
    (TT1.TT_sid=T1.T_start_sid)
    or
    (
        TT1.TT_sid!=T1.T_start_sid and
        TT1.TT_price_yz!=0
```

```
        )
    )
    and
    (
        (TT3.TT_sid=T2.T_start_sid)
        or
        (
            TT3.TT_sid!=T2.T_start_sid and
            TT3.TT_price_yz!=0
        )
    )
    and
    TT2.TT_price_yz!=0
    and
    TT4.TT_price_yz!=0
    and
    TT1.TT_depart_time>[TIME]
ORDER BY
    price_total,
    case
        when ((TT4.TT_arrive_time-TT1.Tt_depart_time)>interval '0 min')
        then TT4.TT_arrive_time-TT1.Tt_depart_time
        else TT4.TT_arrive_time-TT1.Tt_depart_time + interval '24 hour'
        end,
    TT1.Tt_depart_time

LIMIT 20

    ) as RES
WHERE
    ES1.ES_tid=RES.tid1 and
    ES1.ES_tid=TT1.TT_tid and
    ES1.ES_tid=TT2.TT_tid and
    ES1.ES_tid=TT3.TT_tid and
    TT1.TT_sid=RES.depart_station and
    TT2.TT_sid=RES.transfer_station1 and
    TT3.TT_sid=ES1.ES_current_sid and
    (TT3.TT_count>=TT1.TT_count) and
    (TT3.TT_count<TT2.TT_count)
    and
    ES2.ES_tid=RES.tid2 and
    ES2.ES_tid=TT4.TT_tid and
    ES2.ES_tid=TT5.TT_tid and
    ES2.ES_tid=TT6.TT_tid and
    TT4.TT_sid=RES.transfer_station2 and
    TT5.TT_sid=RES.arrive_station and
    TT6.TT_sid=ES2.ES_current_sid and
    (TT6.TT_count>=TT4.TT_count) and
    (TT6.TT_count<TT5.TT_count)
GROUP BY
    RES.depart_station,
    RES.transfer_station1,
    RES.transfer_station2,
    RES.arrive_station,
    RES.transfer_city,
    RES.tid1,
    RES.tid2,
    RES.price1,
    RES.price2,
    RES.price_total
;
```

参数:

1. [CITY1] 上车城市

2. [CITY2] 下车城市
3. [TIME] 查找在此时间之后的列车

## 3.2.5. 预订车次座位

### 3.2.5.1. 直接生成新订单号

```
SELECT COUNT(*)+1
FROM ORDERS;
```

## 3.2.6. 查询订单和删除订单

### 3.2.6.1. 查询某个时间范围内的订单

```
SELECT
    O_oid, O_order_date,
    scid.ISC_sname as depart_station,
    scia.ISC_sname as arrive_station,
    O_price,
    O_valid
FROM ID_Station_City as scid, ID_Station_City as scia, Orders
WHERE O_start_sid=scid.ISC_sid
    and O_arrive_sid=scia.ISC_sid
    and O_order_date<=%date_upperbound
    and O_order_date>=%date_lowerbound;
```

## 3.2.7. 管理员

### 3.2.7.1. 总订单数

```
select count(*) from Orders;
```

### 3.2.7.2. 总票价

```
select sum(O_price) from Orders;
```

## 3.2.8. 最热点车次排序，排名前10的车次

```
select O_tid1, count(O_tid1)+count(O_tid2) as sum_cnt
from Orders
group by O_tid1
order by sum_cnt desc;
```

## 3.2.9. 当前注册用户列表

```
SELECT
    P_pid,
    P_phone,
    P_pname,
    P_uname,
    P_credit_card
FROM Passenger;
```

## 3.2.10. 查看每个用户的订单

```
SELECT
    O_oid,
    O_pid,
    O_order_date,
    O_start_sid,
    O_arrive_sid,
    O_price,
    O_date1,
    O_time1,
    O_tid1,
    O_start_sid1,
    O_arrive_sid1,
    O_date2,
    O_time2,
    O_tid2,
    O_start_sid2,
    O_arrive_sid2,
    O_valid
FROM Orders
WHERE O_pid=[PID];
```

参数:

1. [PID] 要查询的用户的ID(身份证号)

# 3.3. SQL刷新语句及调用地点

## 3.3.1. 创建订单并扣除余票

```
--插入订单
INSERT INTO Orders
VALUES(......);

--减少余票
UPDATE Empty_Seat
SET ES_left_yz=ES_left_yz-1
FROM Train_Table as TT1, Train_Table as TT2, Train_Table as TT3
WHERE
    ES_tid=$tid and
    ES_tid=TT1.TT_tid and
    ES_tid=TT2.TT_tid and
    ES_tid=TT3.TT_tid and
    TT1.TT_sid=[depart_sid] and
    TT2.TT_sid=[arrive_sid] and
    TT3.TT_sid=ES_current_sid and
    ES_date=[date] and
    (TT3.TT_count>=TT1.TT_count) and
    (TT3.TT_count<TT2.TT_count);
```

参数:

1. [depart_sid] 出发站id
2. [arrive_sid] 到达站id
3. [date] 日期

## 3.3.2. 取消订单与恢复余票

```
UPDATE Orders
SET O_valid=0
WHERE O_oid=[O_ID];

SELECT O_type1 FROM ORDERS WHERE O_oid=[OID];

UPDATE Empty_Seat
SET ES_left_yz=ES_left_yz+1
FROM Train_Table as TT1, Train_Table as TT2, Train_Table as TT3, Orders
WHERE
    ES_tid=O_tid1 and
    ES_tid=TT1.TT_tid and
    ES_tid=TT2.TT_tid and
    ES_tid=TT3.TT_tid and
    TT1.TT_sid=$O_start_sid1 and
    TT2.TT_sid=$O_arrive_sid1 and
    TT3.TT_sid=ES_current_sid and
    ES_date=[DATE] and
    (TT3.TT_count>=TT1.TT_count) and
    (TT3.TT_count<TT2.TT_count)and
    O_oid=[OID]

;
```

参数:

1. [OID] 订单id
2. [DATE] 日期

## 3.3.3. 开放新一天的购票

```
INSERT INTO Empty_Seat(
    ES_tid,
    ES_current_sid,
    -- ES_next_sid
    ES_date
    -- ES_left_yz,
    -- ES_left_rz,
    -- ES_left_yws,
    -- ES_left_ywz,
    -- ES_left_ywx,
    -- ES_left_rws,
    -- ES_left_rwx
    )
SELECT
    TT_tid,
    TT_sid,
    current_date+[new_day]
FROM Train_Table;
```

参数:

1. [new_day] 开放距今几天的购票

例如:

开放今天的购票: `[new_day]=0`

# 4. 数据库系统实现

## 4.1. 数据导入

在本实验中数据的预处理使用python脚本解析正则表达式的形式, 将数据转化成规整的CSV文件. 在预处理过程中同时对数据中的缺失项做了处理, 同时根据换乘查询的需要, 计算生成了城市间的连通图表和车站之间的连通图表, 希望通过这些数据的预处理来提高查询速度, 降低查询难度.

## 4.2. 余票查询

通过引入以下的框架, 可以使得一个查询语句支持两站之间的余票查询, 借此可以快速使某个查询支持余票查询.

```
SELECT
    RES.XX
    min(ES_left_yz) as left_yz
FROM Empty_Seat,
    Train_Table as TT1,
    Train_Table as TT2,
    Train_Table as TT3,
    (
        XX,
        ......
    ) as RES
WHERE
    ES_tid=RES.PT_tid and
    ES_tid=TT1.TT_tid and
    ES_tid=TT2.TT_tid and
    ES_tid=TT3.TT_tid and
    TT1.TT_sid=RES.PT_depart_sid_f and
    TT2.TT_sid=RES.PT_arrive_sid_f and
    TT3.TT_sid=ES_current_sid and
    (TT3.TT_count>=TT1.TT_count) and
    (TT3.TT_count<TT2.TT_count)
GROUP BY
    RES.XX
;
```

## 4.3. LIMIT 语句的引入

通过引入LIMIT语句, 可以限制结果集的大小, 从而大幅提高某些查询语句的速度.

## 4.4. 其他查询语句

其他查询语句已在上文详述.

---

# 5. 前端实现

前端通过php语句调用sql查询语句来访问数据库. 尽管从软件工程的角度上来讲, 应该使用MVC模型为佳. 但是考虑到小组成员都是第一次接触PHP, 因此没有采用这些面向对象的软件开发模型.

~~前端大哥辛苦子~~

# 6. ACKNOWLEDGE

[1][网络技术应用: MHW-50382]
[2][stackoverflow]