# 数据库实验二 设计文档

冯吕　丁诗哲

November 30, 2017

## 一、需求分析（lab2 说明）

### 1. 记录列车信息

需要记录车站: 始发站, 中间经停站, 终点站，最长 20 字符的站名，每站的发车时间和到达时间，不同座位的票价等信息。

### 2. 记录列车座位情况

每次列车、每类座位、每天有 5 个。

### 3. 记录乘客信息

每位乘客在使用前需要注册, 登记姓名、身份证、手机号、信用卡、用户名等信息。并给出后续链接。

### 4. 支持查询具体车次

根据车次序号、日期显示具体车次的静态信息以及动态信息。并给出预订功能链接。

### 5. 查询两地之间的车次

输入出发和到达城市、日期和时间，显示两地之间的直达列车和余票信息（按价格升序前十）

两地之间换乘一次的列车组合和余票信息（按价格升序前十）。换乘地必须是同一城市，换乘地是同一车站, 那么 1 小时 <= 换乘经停时间 <= 4 小时，如果换乘地是同城的不同车站, 那么 2 小时 <= 换乘经停时间 <= 4 小时。发车时间 >= 给定的出发时间。[**换乘未实现**]

### 6. 查询返程信息

和 5 相同，只是返程。

### 7. 预订车次座位

每个车次显示车次、出发日期、出发时间、出发车站、到达日期、到达时间、到达车站、座位类型、本次车票价。点击确认生成，取消返回。

### 8. 查询订单和删除订单

给定出发日期范围, 显示历史订单列表及相关信息。

### 9. 管理员登录

略。

## 二、概念设计

由需求 1，我们可以设列车 (Train) 可作为一个实体集，而列车相关的一系列信息比如始发站, 终点站，站名，可作为其属性。考虑到一趟列车可以有若干经停站，即经停站的信息不能是原子的，我们便不能将它作为列车的属性对待。我们可以设车站 (Station) 为一个实体集，其属性有车站名。再为列车和车站建立一个联系——“经过 (PassBy)”，即列车“经过”车站。那么列车经过某站的信息例如该站的站名、该站在该趟列车时刻表上的站次、到站离站时间、票价等信息，可作为“经过”这个联系的属性。

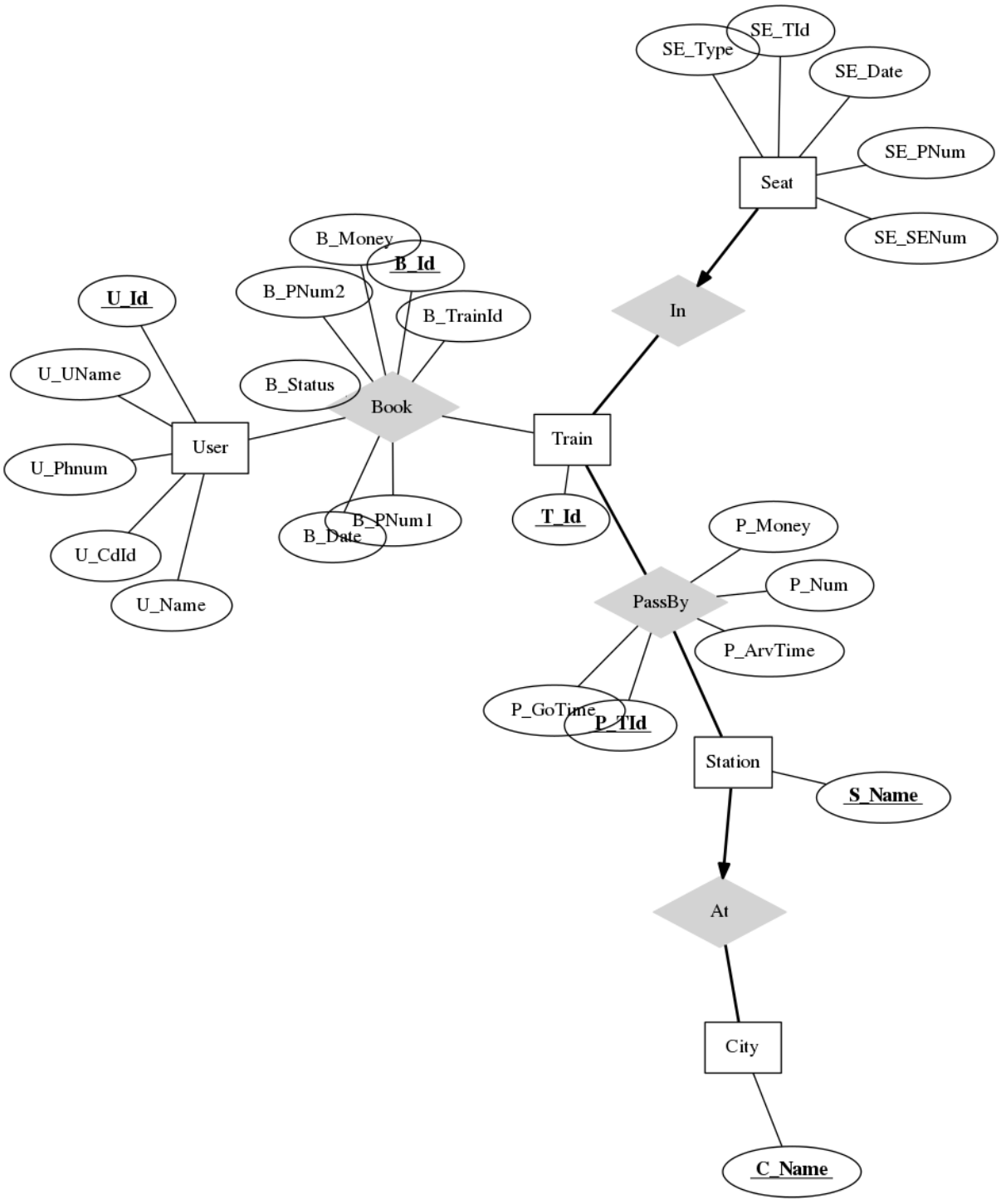
另外，在需求 5 中，我们会查询两地之间的列车。考虑到这一点，我们将城市 (City) 作为一个实体集，城市名作为其一个属性。城市和车站两个实体集之间构成一个关系“位于（At)”，即车站“位于”城市。

由需求 2，我们设列车座位 (Seat) 为一个实体集，其属性有日期、座位数量。座位和列车构成关系“属于（In)”，即座位“属于”列车。

由需求 3，我们设用户 (User) 为一个实体集，其属性有用户身份证号，用户姓名，用户名，用户手机号码，用户银行卡号等。对于用户而言，他订购车票的操作将它同列车联系起来，我们称这个联系为“预订（Book)”。这个“预订”需要有订单号、预订车次、日期、用户 ID 以及从何站上下车（用站次）以及订单是否被取消等信息作为属性。

需要说明的是，我们在这里并没有像实际情况中，把用户和座位直接建立联系。这是因为用户和座位之间的联系在这次实验中是不必要的。根据需求分析，本次试验中，列车所拥有的座位不必表示一个个实际座位的抽象，只需要记录当前列车在某个区间段有多少空闲作为即可。用户订到一张票并不必要对应到一个具体座位。因此，我们只是在用户和列车之间建立了联系“预订（Booking）”。

现在，我们已经有了实体集 **User**, **Train**, **City**, **Seat**, **Station** 和联系 **Book**,**Passby**, **In**, **At**，除此之外还有每个实体集和联系集相应的属性。我们画出 ER 图，如下图所示：




Train-Tickets-Booking-System E-R Diagram

## 三、关系模式和范式细化

对于上面的 ER 图，我们可以选择直接建表。即每个实体集和联系集，我们都创建一个表。对于实体集，其表的键对应实体集的各个属性，主键则是图中粗体表示的属性。对于每一个联系集，其表的键除了自身属性外，再包含参与其联系的两个实体集的表的主键作为外键。但在其基础上，我们可以对关系模型做进一步优化。

对于实体集 Station 和 City 以及它们的联系集 At，我们可以把 At 归并到 Station 表中，将 CityName 作为 Station 的一个属性。

实际上，实体集 Train 的属性比如始发站、终点站和 PassBy 中的数据产生冗余，可以舍弃。此时 Train 只剩下属性 TrainId，再对它单独建表已经没有意义。可以将它归并到 PassBy 中，和 PassBy 中的车站（事实上是一次列车依次经过站的站序）共同作为 PassBy 的主键。这时，对于 Book 和 Seat 而言，TrainId 可作为它们的一个属性。

根据这次实验的设计需求，我们需要知道每辆车次的订票和余票情况。但考虑到统计七天之内所有车次每两个相邻站之间的余票情况，存储开销较大。因此我们放弃建立 Seat 表的做法，转而建立一个名为 TicketInfo 的表，用来存储和维护所有车次每两个相邻站之间车票被订购的情况。这种实现和前者的不同之处在于，我们初始化时表为空，某个日期某个车次某两个相邻站之间票被订购后，表 TicketInfo 表才被插入或更新。

根据上面的讨论，我们的建表方案如下所示：

首先建立两个类型，用以描述订单的状态以及座位的类型。

```
create type status_type as enum (
    'normal', 'cancelled', 'past'
);

create type seat_type as enum (
    'YZ', 'RZ', 'YW1', 'YW2', 'YW3', 'RW1', 'RW2'
);
```

表 Station：

```
create table Station(
    S_Name          varchar(20) primary key,
    S_City          varchar(20) not null
);
```

表 PassBy:

```
create table Passby(
    P_TrainId       varchar(6) not null,
    P_StationName   varchar(20) not null,
    P_StationNum    integer not null,
    P_ArriveTime    time,
    P_GoTime        time,
    P_MoneyYZ       float,
    P_MoneyRZ       float,
    P_MoneyYW1      float,
    P_MoneyYW2      float,
    P_MoneyYW3      float,
    P_MoneyRW1      float,
    P_MoneyRW2      float,
    primary key (P_TrainId, P_StationNum),
    foreign key (P_StationName) references Station(S_Name)
);
```

表 UserInfo:

```
create table UserInfo(
    User_Id         char(18) primary key,
    U_Name          varchar(20) not null,
    U_Phone         char(11) not null,
    U_UName         varchar(20) not null,
    U_CreditCardId  char(16) not null
);
```

表 Book:

```
create table Book
(
    B_Id            SERIAL primary key,
    B_UserId        char(18) not null,
    B_TrainId       varchar(6) not null,
    B_Date          date not null,
```

```
    B_StationNum1   integer not null,
    B_StationNum2   integer not null,
    B_SType         seat_type not null,
    B_Money         integer not null,
    B_Status        status_type not null,
    foreign key (B_TrainId, B_StationNum1) references Passby
        (P_TrainId, P_StationNum),
    foreign key (B_UserId) references UserInfo(User_Id)
);
```

表 TicketInfo：

```
create table TicketInfo
(
    T_TrainId       varchar(6) not null,
    T_PStationNum   integer not null,
    T_Type          seat_type not null,
    T_Date          date not null,
    T_SeatNum       integer not null,
    primary key (T_TrainId, T_PStationNum, T_Type, T_Date),
    foreign key (T_TrainId, T_PStationNum) references Passby
        (P_TrainId, P_StationNum)
);
```

本次试验比较简单，就上述实现而言，我们组没有发现可以模式细化的必要。

## 三、SQL 查询语句的模板

下面给出本次试验比较复杂的查询语句的模板，下面的查询语句基本都需要前端配合才能完成相应的工作。

### 1. 给定某车次以及某两站站序，查询余票

由于 TicketInfo 表是存储的是某天一个车次相邻两站票被订购的次数。未被订就不存在记录，所以下面查询方式需要配合前端操作才能得到余票还有多少。注意'K474'、1、13、'2017-12-01'

是参数。

```
with T1(T1_Type, T1_SeatNum) as
(select T_Type, T_SeatNum
    from TicketInfo
    where T_TrainId = 'K474'
        and T_PStationNum >= 1
        and T_PStationNum < 13
        and T_Date = ),

select T1_Type, MAX(T1_SeatNum)
from T1
group by T1_Type;
```

### 2. 给定两地和时间，查询所有可以乘坐的列车（按硬座票价升序）

下面的查询方法没有考虑余票，因此需要前端进一步配合。‘天津’、‘苏州’和‘00:00’是可以配置的参数。

```
with S1(S1_TrainId, S1_StationNum) as
(select Passby.P_TrainId, Passby.P_StationNum
    from Passby, Station
    where Passby.P_StationName = Station.S_Name
        and Station.S_City = '天津'),

S2(S2_TrainId,S2_StationNum) as
(select Passby.P_TrainId, Passby.P_StationNum
    from Passby, Station
    where Passby.P_StationName = Station.S_Name
        and Station.S_City = '苏州'),

T1(T1_TrainId) as
(   select S1.S1_TrainId
    from S1, S2
    where S1.S1_TrainId = S2.S2_TrainId
```

```
17          and S1.S1_StationNum < S2.S2_StationNum
18      ),
19
20  T3(Tp_trainid, Tp_stationname, Tp_stationnum, Tp_arrivetime,
        Tp_gotime, Tp_moneyyz, Tp_moneyrz, Tp_moneyyw1,
       Tp_moneyyw2, Tp_moneyyw3, Tp_moneyrw1, Tp_moneyrw2,
       Tt1_trainid, Ts_name, s_city)  as
21      (select *
22      from Passby, T1, Station
23      where Passby.P_TrainId = T1.T1_TrainId
24          and Passby.P_StationName = Station.S_Name
25          and (Station.S_City = '天津'
26              or Station.S_City = '苏州')),
27
28  T4(T4_id,  yz, rz, yw1, yw2, yw3, rw1, rw2) as
29  (
30  select Tp_trainid, Max(Tp_moneyyz)-Min(Tp_moneyyz), Max(
        Tp_moneyrz)-Min(Tp_moneyrz), Max(Tp_moneyyw1)-Min(
        Tp_moneyyw1), Max(Tp_moneyyw2)-Min(Tp_moneyyw2), Max(
        Tp_moneyyw3)-Min(Tp_moneyyw3), Max(Tp_moneyrw1)-Min(
        Tp_moneyrw1), Max(Tp_moneyrw2)-Min(Tp_moneyrw2)
31  from T3
32  group by Tp_trainid
33  ),
34
35  T5(T5_id,T5_StationName, T5_GoTime, T5_StationNum) as
36  (
37  select T1.T1_TrainId, Passby.P_StationName, Passby.P_GoTime,
        Passby.P_StationNum
38  from T1, Passby, Station
39  where P_TrainId = T1_TrainId
40      and P_StationName = Station.S_Name
41      and Station.S_City = '天津'
```

```
42  ),
43
44  T6(T6_id, T6_StationName, T6_ArriveTime, T6_StationNum) as
45  (
46  select T1.T1_TrainId, Passby.P_StationName, Passby.
        P_ArriveTime, Passby.P_StationNum
47  from T1, Passby, Station
48  where P_TrainId = T1_TrainId
49      and P_StationName = Station.S_Name
50      and Station.S_City = '苏州'
51  )
52
53  -- order by yz;
54  select DISTINCT T4.T4_id,T5.T5_StationName, T5.T5_GoTime, T6
        .T6_StationName,
55  T6.T6_ArriveTime,  T4.yz, T4.rz, T4.yw1, T4.yw2, T4.yw3, T4.
        rw1, T4.rw2, T5.T5_StationNum, T6.T6_StationNum
56  from T4, T5, T6
57  where T4.T4_id = T5.T5_id
58      and T6.T6_id = T5.T5_id
59      and T5_GoTime > '00:00'
60  order by T4.yz
61  ;
```

### 3. 订票退订时，发生的操作

```
1  -- BOOK
2  insert into
3      Book(B_UserId,B_TrainId,B_Date,B_StationNum1,
           B_StationNum2,B_SType,B_Money,B_Status)
4  values(111, 'K474', '2017-12-01', 1, 3, 'YZ', 50, 'normal');
5  -- CANCAL
6  update Book  set B_Status = 'cancelled'
```

```
7  where B_Id = 1
```

同时，订票和退订时，对表 TicketInfo 也有操作，订票时：

```
1  select T_SeatNum
2  from TicketInfo
3  where T_TrainId = 'K474'
4      and T_PStationNum = 1
5      and T_Type = 'YZ'
6      and T_Date = '2017-12-01';
7  -- 前端检查返回结果，如果为空
8  insert into
9      TicketInfo(T_TrainId,T_PStationNum,T_Type,T_Date,
           T_SeatNum)
10 values ('K474', 1, 'YZ', '2017-12-01', 1),
11         ('K474', 1, 'YZ', '2017-12-01', 2);
12 -- 如果不空 [NEW_NUM] <= [NUM] + 1
13 update TicketInfo
14 set T_SeatNum = [NEW_NUM]
15 where T_TrainId = 'K474'
16     and T_PStationNum = 3
17     and T_Type = 'YZ'
18     and T_Date = '2017-12-01';
```

退订时：

```
1  select T_SeatNum
2  from TicketInfo
3  where T_TrainId = 'K474'
4      and T_PStationNum = 1
5      and T_Type = 'YZ'
6      and T_Date = '2017-12-01';
7  -- 前端检查返回结果，如果是 1
8  delete from TicketInfo
9  where T_TrainId = 'K474'
10     and T_PStationNum = 1
```

```
    and T_Type = 'YZ'
    and T_Date = '2017-12-01';
-- 否则 [NEW_NUM] <= [NUM] - 1
update TicketInfo
set T_SeatNum = [NEW_NUM]
where T_TrainId = 'K474'
    and T_PStationNum = 1
    and T_Type = 'YZ'
    and T_Date = '2017-12-01';
```

**注：**以上 sql 语句除了 createtable 之外，均嵌入在前端的 php 文件中，和前端配合完成某个工作。