Lab2-12306设计文档

高云聪 张丽玮 胡筱涵

需求分析

仿照12306设计和实现一个火车订票系统。

总体要求：

- 存储数据在数据库中
- 实现9个具体功能
- 提供基本的Web界面
- 不要求事务处理

具体需求：

1. 记录列车信息

根据CSV表中所给信息，记录包括：

- 车站（始发站，中间经停站，终点站）的站名（最长20字符）
- 每站的发车时间和到达时间
- 票价（硬座，软座，硬卧，软卧）等信息

,	站名,	到时,	发时,	停留（分）,	历时（分）,	里程（km）,	硬座/软座,	硬卧（上/中/下）,	软卧（上/下）
1,	宝鸡,	-,	16:00,	-,	-,	-,	-,	-,	-
2,	蔡家坡,	16:30,	16:33,	03分,	30,	0,	9/-,	55/58/63,	81.5/92
3,	杨陵,	17:01,	17:04,	03分,	61,	0,	12.5/-,	58.5/61/66,	88.5/100
4,	咸阳,	17:43,	17:53,	10分,	103,	0,	20.5/-,	66.5/70/76,	101.5/115
5,	西安,	18:17,	18:37,	20分,	137,	0,	25.5/-,	71.5/75/81,	109.5/124
6,	渭南,	19:16,	19:18,	02分,	196,	0,	32.5/-,	78.5/82/89,	120.5/136
7,	潼关,	20:21,	20:23,	02分,	261,	0,	40.5/-,	86.5/90/98,	134.5/152
8,	灵宝,	21:23,	21:25,	02分,	323,	0,	48.5/-,	94.5/99/107,	149.5/169
9,	三门峡西,	21:49,	21:51,	02分,	349,	0,	54.5/-,	102.5/107/116,	158.5/179
10,	洛阳,	23:50,	00:14,	24分,	470,	0,	69/-,	130/135/147,	205/231
11,	郑州,	01:53,	02:14,	21分,	-847,	0,	82/-,	152/158/172,	240/271
12,	开封,	03:00,	03:02,	02分,	-780,	0,	90/-,	166/173/188,	262/296
13,	商山,	05:26,	05:44,	18分,	-634,	0,	109/-,	200/208/226,	321/362
14,	黄口,	06:05,	06:15,	10分,	-595,	0,	113.5/-,	207.5/216/235,	333.5/376
15,	徐州,	06:53,	07:00,	07分,	-547,	0,	118.5/-,	215.5/224/244,	343.5/387
16,	新沂,	08:13,	08:17,	04分,	-467,	0,	124.5/-,	227.5/237/257,	364.5/411
17,	连云港东,	09:52,	09:52,	00分,	-368,	0,	134.5/-,	243.5/253/275,	391.5/441

2. 记录列车座位情况

记录每次列车、每类座位的信息。

- 车次
- 对应的站点
- 座位的类型
- 对应的票价
- 对应的余票，重点考虑余票问题，余票应该满足每天的每个车次每个站点有票的座位类型（即在所给CSV表上有票价）默认余票数为5

3. 记录乘客信息

记录乘客在使用前注册。

- 登记的姓名（可重名）
- 身份证（唯一）
- 手机号（唯一）
- 信用卡（唯一）
- 用户名（唯一）

4. 查询具体车次

输入车次序号和日期，显示该车次所有信息。（默认出发站为起始站）

- 显示静态信息：始发站，中间经停站，终点站；每站的发车时间和到达时间；票价
- 也有动态信息：余票。（余票上有链接可以跳转预订功能网页）

5. 查询两地之间的车次

输入出发地城市名、到达地城市名、出发日期和出发时间,显示两地之间的直达列车和余票信息及两地之间换乘一次的列车组合和余票信息。（选取最优前十）

其中换乘地必须是同一城市，如果换乘地是同一车站， 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。

- 显示静态信息：出发站，到达站；每站的发车时间和到达时间；票价
- 也有动态信息：余票。（余票上有链接可以跳转预订功能网页）

6. 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

用户点击确认生成订单,点击取消返回登录首页。

7. 查询订单和删除订单

给定出发日期范围，显示订单列表。

提供链接，点击显示订单具体信息：

- 订单号
- 出发日期
- 出发到达站
- 总票价
- 订单状态（是否已经取消）。

提供链接，点击取消订单。

8. 管理员

Admin登录后显示不同的登录首页。

Admin可以看到下述信息：

- 总订单数
- 总票价
- 最热点车次排序
- 当前注册用户列表
- 每个用户的订单

概念设计

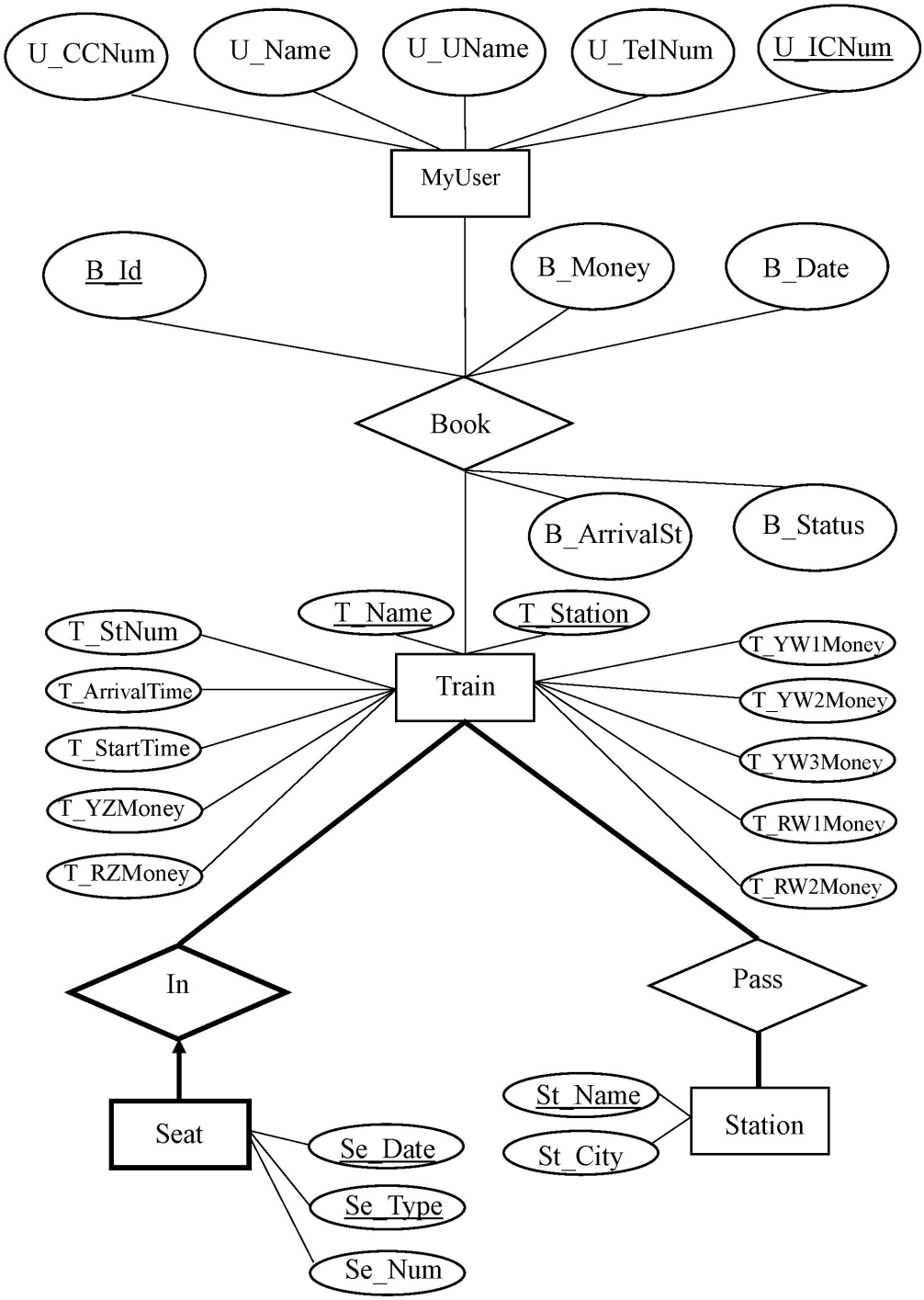
由需求1，我们将列车抽象为一个实体集“列车（Train）”。又由需求1，我们需记录始发站、中间经停站、终点站、发车时间、到达时间、票价等信息。又由于中间经停站、发车时间、到达时间与票价对于列车来说并不是原子的，因此不能作为列车的属性。我们建立属性为车站名的实体集“车站（Station）”，并建立关系集“经停（PassBy）”来描述列车与车站之间的经停关系，而此时中间经停站站名、发车时间、到达时间与票价便可作为此联系集的属性来描述列车经停某一车站时的信息。余下始发站、终点站与列车名便可作为实体集“列车Train”的信息。

由需求2，我们可将列车座位抽象为一个实体集“座位（Seat）”，其属性为日期、座位类型和剩余座位数量。而我们可以用联系“属于（In）”来描述座位与列车的从属关系。

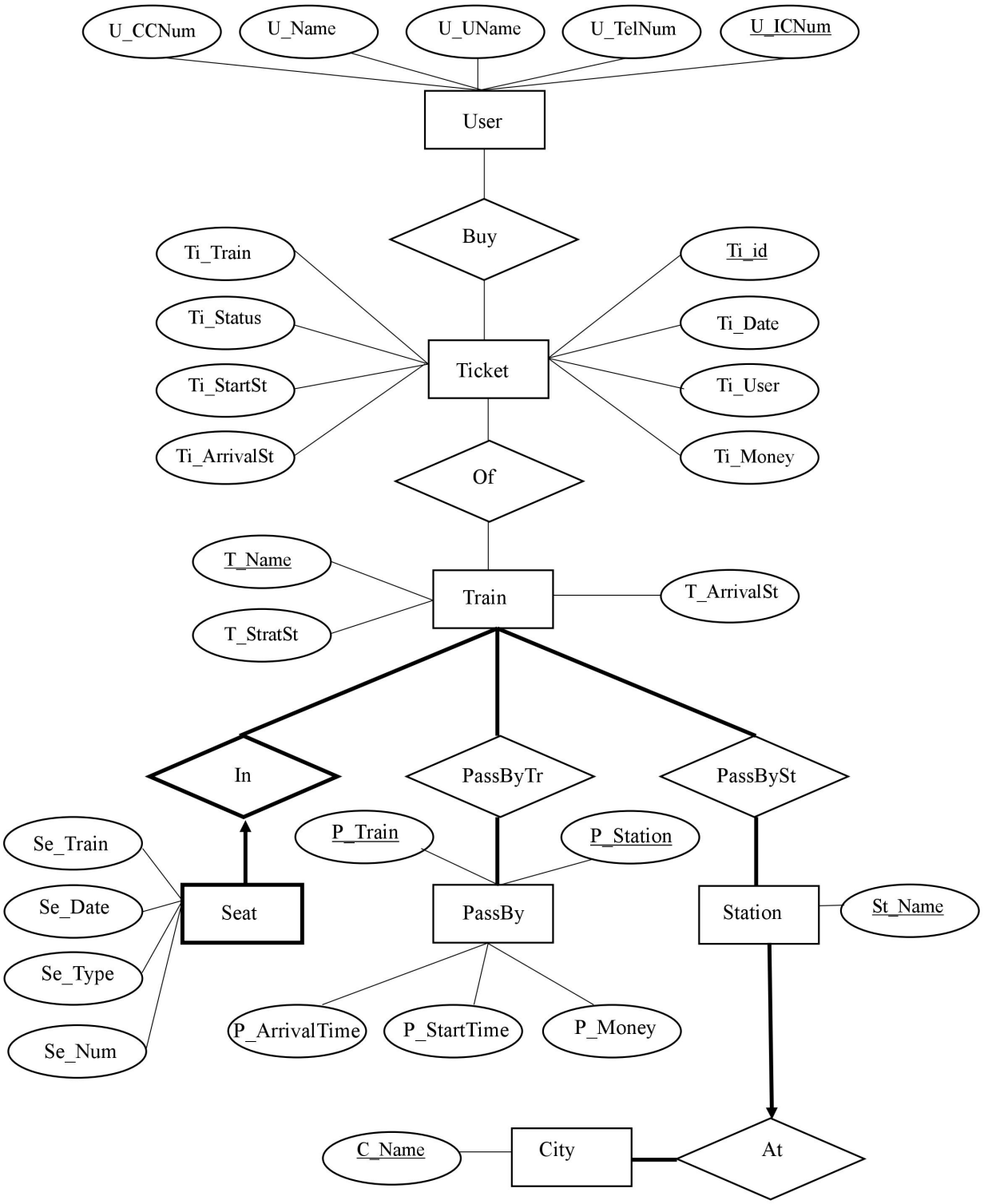
由需求3，我们需要记录乘客的注册信息，因此我们将乘客抽象为一个实体集“用户（User）”，其属性为姓名、身份证、手机号、信用卡和用户名。而由系统保存每位乘客所有的历史订单这一需求，我们可建立联系集“预定（Book）”，根据需求7，我们可将此联系集的属性确定为预定者用户名、订单号、日期、出发到达站、总票价和订单状态。

由需求5，我们建立一个实体集“城市（City）”，并将城市名作为其属性。为了实现查询两地间车次这一功能，我们将实体集“车站（Station）”与实体集“城市（City）”用联系“位于（Locate）”联系起来。

通过以上分析，我们可以初步画出E/R图如下：



我们将联系集的属性转化为实体集并增加新的联系集后重新绘制E/R图如下：


U_CCNum
U_Name
U_UName
U_TelNum
U_ICNum
User
Buy
Ti_Train
Ti_Status
Ti_StartSt
Ti_ArrivalSt
Ticket
Ti_id
Ti_Date
Ti_User
Ti_Money
Of
T_Name
T_StratSt
Train
T_ArrivalSt
In
PassByTr
PassBySt
Se_Train
Se_Date
Se_Type
Se_Num
Seat
P_Train
P_Station
PassBy
Station
St_Name
P_ArrivalTime
P_StartTime
P_Money
C_Name
City
At

逻辑设计

将E/R模型转换为关系模型。

首先定义座位类型和订单状态如下：

```
CREATE TYPE se_type as enum
(
    'YZ', 'RZ', 'YW1', 'YW2', 'YW3', 'RW1', 'RW2'
);
```

```
CREATE TYPE b_status as enum
(
    'cancelled','uncancelled'
);
```

由上述E/R模型，我们可以将实体集关系表如下：

```
CREATE TABLE Train
(
    T_Name          varchar(6) primary key,
    T_StartSt       varchar(20) not null,
    T_ArrivalSt     varchar(20) not null,
);
```

```
CREATE TABLE City
(
    C_Name          Varchar(20) primary key
);
```

```
CREATE TABLE User
(
    U_ICNum         char(18) primary key,
    U_TelNum        char(11) not null,
    U_UName         varchar(20) not null,
    U_Name          varchar(20) not null,
    U_CCNum         char(16) not null
);
```

```
CREATE TABLE Station
(
    St_Name         varchar(20) primary key
);
```

```
CREATE TABLE Seat
(
    Se_Train        varchar(6) not null,
    Se_Date         date not null,
    Se_Type         se_type not null,
    Se_Num          integer not null
);
```

```
CREATE TABLE Ticket
(
    Ti_Id            serial primary key,
    Ti_Date          date not null,
    Ti_User          char(18) not null,
    Ti_Money         float not null,
    Ti_Train         varchar(6) not null,
    Ti_Status        b_status not null,
    Ti_StartSt       varchar(20) not null,
    Ti_ArrivalSt     varchar(20) not null
);
```

```
CREATE TABLE PassBy
(
    P_Train         varchar(6) not null,
    P_Station       varchar(20) not null,
    P_ArrivalTime   time not null,
    P_StartTime     time not null,
    P_Money         float not null,
    primary key (P_Train,P_Station)
);
```

将E/R图中的联系集转化为关系表如下：

```
CREATE TABLE Buy
(
    B_User          char(18) not null,
    B_Ticket        serial not null,
    foreign key (B_User) references User(U_ICNum),
    foreign key (B_Ticket) references Ticket(Ti_Id)
);
```

```
CREATE TABLE Of
(
    O_Ticket        serial not null,
    O_Train         varchar(6) not null,
    foreign key (O_Ticket) references Ticket(Ti_Id),
    foreign key (O_Train) references Train(T_Name)
);
```

```
CREATE TABLE In
(
    I_Train         varchar(20) not null,
    foreign key (I_Train) references Train(T_Name)
);
```

```
CREATE TABLE PassByTr
(
    PT_Train        varchar(6) not null,
    PT_PBStation    varchar(20) not null,
    PT_PBTrain      varchar(6) not null,
    foreign key (PT_Train) references Train(T_Name),
    foreign key (PT_PBStation,PT_PBTrain) references PassBy(P_Station,P_Train)
);
```

```
CREATE TABLE PassBySt
(
    PS_PBStation    varchar(20) not null,
    PS_PBTrain      varchar(6) not null,
    PS_Station      varchar(20) not null,
    foreign key (PS_PBStation,PS_PBTrain) references PassBy(P_Station,P_Train),
    foreign key (PS_Station) references Station(St_Name)
);
```

```
CREATE TABLE At
(
    A_Station       varchar(20) not null,
    A_City          varchar(20) not null,
    foreign key (A_Station) references Station(St_Name),
    foreign key (A_City) references City(C_Name)
);
```

模式细化

初步简化

根据最初的ER图，我们可以对每个实体集和联系集直接进行建表。但其关系显然过于复杂，且数据冗余严重。

故而我们选择在此基础上，对关系模型进行优化。

为此，我们将原ER图划分为五个部分：

- User
- Buy, Ticket, Of
- Seat, In, Train
- Train, PassByTr, PassBy, PassBySt, Station
- Station, At, City

对于实体集Station和City及他们间的联系集At，我们可以把它们归并为同一实体集Station，含有城市名作为该实体集的属性。

表Station:

```
CREATE TABLE Station
(
    St_Name          varchar(20) primary key,
    St_City          varchar(20) not null
);
```

实体集Train中除主键车名外的属性，始发站和终点站都与PassBy中数据产生冗余，没有单独建立实体集的必要。

因而我们可以将PassBy与Train归并，把车名和站名作为联合主键。

同时将联系集PassBySt与Train归并，将上步所得实体集Staition中主键站名作为外键。

表Train：

```
CREATE TABLE Train
(
    T_Name          varchar(6) not null,
    T_Station       varchar(20) not null,
    T_StNum         integer not null,
    T_ArrivalTime   time,
    T_StartTime     time,
    T_YZMoney       float,
    T_RZMoney       float,
    T_YW1Money      float,
    T_YW2Money      float,
    T_YW3Money      float,
    T_RW1Money      float,
    T_RW2Money      float,
    primary key (T_Name,T_Station),
    foreign key (T_Station) references Station(St_Name)
);
```

把联系集Buy，实体集Ticket及联系集Of归并为联系集Book,则可将上步中归并后实体集Train的主键车名和站名作为其两个属性。

表Book:

```
CREATE TABLE Book
(
    B_Id            serial primary key,
    B_User          char(18) not null,
    B_Money         float not null,
    B_Date          date not null,
    B_Train         varchar(6) not null,
    B_StartSt       varchar(20) not null,
    B_ArrivalSt     varchar(20) not null,
    B_Status        b_status not null,
    foreign key (B_User) references MyUser(U_ICNum),
    foreign key (B_Train,B_StartSt) references Train(T_Name,T_Station)
);
```

把实体集Seat和联系集In进行归并，用以记录位置信息。将归并后实体集Train中的主键车名和站名作为外键。

表Seat:

```
CREATE TABLE Seat
(
    Se_Train         varchar(6) not null,
    Se_Station       varchar(20) not null,
    Se_Date          date not null,
    Se_Type          se_type not null,
    Se_Num           integer not null,
    primary key (Se_Train,Se_Station,Se_Date,Se_Type),
    foreign key (Se_Train,Se_Station) references Train(T_Name,T_Station)
);
```

保留实体集User。

表MyUser:

```
CREATE TABLE MyUser
(
    U_ICNum         char(18) primary key,
    U_TelNum        char(11) not null,
    U_UName         varchar(20) not null,
    U_Name          varchar(20) not null,
    U_CCNum         char(16) not null
);
```

经过上述优化，我们发现已经实现了较为简单的实现，无需再进行模式细化。

1NF

1NF范式：所有的属性都是原子类型。

在需求分析阶段，我们已经完成了此优化。

2NF

消除函数依赖中的部分依赖。

分析发现以上关系模式不存在部分依赖。

3NF

消除函数依赖中的非键传递依赖。

分析发现以上关系模式不存在非键传递依赖。

BCNF

消除所有的函数依赖。

分析发现关系表User存在U_UName → U_ICNum不满足BCNF范式，由于登录仅需用户名，因此我们默认每一用户名都是唯一的，在此情况下U_UName与U_ICNum之间便会存在一个一一对应关系，导致了数据冗余。

我们认为从需求出发，将注册信息存储于一个关系表中优于将原关系表进行分解，此冗余是有必要保存的。

SQL查询语句模板

1. 查询具体车次

输入车次序号和日期，查询列车的静态信息（始发站、中间经停站、终点站、每站的发车时间和到达时间、票价）和动态信息（余票）。

查询车次静态信息的SQL语句如下，其中'1096'是可配置的参数。

```
SELECT
T_Name,T_Station,T_ArrivalTime,T_StartTime,T_YZMoney,T_RZMoney,T_YW1Money,T_YW2Money,T_
YW3Money,T_RW1Money,T_RW2Money
FROM Train
WHERE T_Name = '1096';
```

> 日期信息只与余票信息相关，因此静态信息的查询只将车次号作为输入。

查询车次动态信息的SQL语句如下，注意'1096'、'2018-11-17'是可改变的参数。

```
SELECT Se_Station,Se_Type,Se_Num
FROM Seat
WHERE Se_Train = '1096'
    AND Se_Date = '2018-11-17'
GROUP BY Se_Station;
```

> 事实上只有在存在订购记录时才会建立票数信息的记录，因此需要配合前端才能完成此需求。

2. 查询两地之间的车次

输入出发地城市名、到达地城市名、出发日期和出发时间,查询两地之间的直达列车和余票信息及两地之间的换乘信息和余票信息。

查询两地之间的直达列车和余票信息的SQL语句如下，其中'北京'、'常州'、'2018-11-20'、'00:00'是可改变的参数。

```
-- 先搜过北京的列车
    WITH T1(T1_Name,T1_StNum) AS
    (
        SELECT Train.T_Name,Train.T_StNum
        FROM Train,Station
        WHERE Train.T_Station = Station.St_Name
            AND St_City = '北京'
    ),
-- 再搜过常州的列车
    T2(T2_Name,T2_StNum) AS
    (
        SELECT Train.T_Name,Train.T_StNum
        FROM Train,Station
        WHERE Train.T_Station = Station.St_Name
            AND St_City = '常州'
    ),
```

```
-- 搜北京->常州的列车
    T_Nonstop(TN_Name) AS
    (
        SELECT T1.T1_Name
        FROM T1,T2
        WHERE T1.T1_Name = T2.T2_Name
            AND T1.T1_StNum < T2.T2_StNum
    ),
-- 搜满足出发时间的列车
    T_Nonstop2(TN2_Name) AS
    (
        SELECT T_Nonstop.TN_Name
        FROM T_Nonstop,Train,Station
        WHERE Train.T_Name = T_Nonstop.TN_Name
            AND Train.T_Station = Station.St_Name
            AND Station.St_City = '北京'
            AND Train.T_StartTime >= '00:00'
    ),
-- 搜出所有票价（未做减法）
    T_Money(TM_Name,TM_Station,TM_YZ,TM_RZ,TM_YW1,TM_YW2,TM_YW3,TM_RW1,TM_RW2) AS
    (
        SELECT
Train.T_Name,Train.T_Station,Train.T_YZMoney,Train.T_RZMoney,Train.T_YW1Money,Train.T_Y
W2Money,Train.T_YW3Money,Train.T_RW1Money,Train.T_RW2Money
        FROM T_Nonstop2,Train,Station
        WHERE Train.T_Name = T_Nonstop2.TN2_Name
            AND (Station.St_City = '北京'
                 OR Station.St_City = '常州')
            AND Station.St_Name = Train.T_Station
    ),
-- 获得每次列车各种座位类型
    T_Type(TTP_Name,TTP_Type) AS
    (
        SELECT TN2_Name,CAST('YZ' AS se_type)
        FROM T_Nonstop2
        UNION
        SELECT TN2_Name,CAST('RZ' AS se_type)
        FROM T_Nonstop2
        UNION
        SELECT TN2_Name,CAST('YW1' AS se_type)
        FROM T_Nonstop2
        UNION
        SELECT TN2_Name,CAST('YW2' AS se_type)
        FROM T_Nonstop2
        UNION
        SELECT TN2_Name,CAST('YW3' AS se_type)
        FROM T_Nonstop2
        UNION
        SELECT TN2_Name,CAST('RW1' AS se_type)
        FROM T_Nonstop2
        UNION
        SELECT TN2_Name,CAST('RW2' AS se_type)
        FROM T_Nonstop2
```

```
    ),
-- 做减法并获得每次列车各种座位的票价
    T_MinMoney(TMM_Name,TMM_Type,TMM_Money) AS
    (
        SELECT T_Type.TTP_Name,T_Type.TTP_Type,
            (CASE T_Type.TTP_Type
            WHEN 'YZ' THEN MAX(T_Money.TM_YZ)-MIN(T_Money.TM_YZ)
            WHEN 'RZ' THEN MAX(T_Money.TM_RZ)-MIN(T_Money.TM_RZ)
            WHEN 'YW1' THEN MAX(T_Money.TM_YW1)-MIN(T_Money.TM_YW1)
            WHEN 'YW2' THEN MAX(T_Money.TM_YW2)-MIN(T_Money.TM_YW2)
            WHEN 'YW3' THEN MAX(T_Money.TM_YW3)-MIN(T_Money.TM_YW3)
            WHEN 'RW1' THEN MAX(T_Money.TM_RW1)-MIN(T_Money.TM_RW1)
            WHEN 'RW1' THEN MAX(T_Money.TM_RW2)-MIN(T_Money.TM_RW2)
            ELSE null
            END
            )TMM_Money
        FROM T_Money,T_Type
        GROUP BY T_Type.TTP_Name,T_Type.TTP_Type
    ),
-- 开始算余票
    TS_Num1(TS_train,TS_Station,TS_fromNum) AS
    (
                SELECT Train.T_Name,Train.T_Station,Train.T_StNum
                From Train,T_Nonstop2,Station
                WHERE Train.T_Name = T_Nonstop2.TN2_Name
                AND Train.T_Station = Station.St_Name
                AND Station.St_City = '北京'
    ),
    TS_Num2(TS_train,TS_Station,TS_toNum) AS
    (
                SELECT Train.T_Name,Train.T_Station,Train.T_StNum
                From Train,T_Nonstop2,Station
                WHERE Train.T_Name = T_Nonstop2.TN2_Name
                AND Train.T_Station = Station.St_Name
                AND Station.St_City = '常州'
    ),
-- 得到每次列车的座位类型及其余票
    TS_Seat(TS_Train,TS_Type,TS_Num) AS
    (
                SELECT T_Nonstop2.TN2_Name,Seat.Se_Type,MIN(Seat.Se_Num)
                FROM Train,Seat,TS_Num1,TS_Num2,T_Nonstop2
                WHERE Train.T_Name = T_Nonstop2.TN2_Name
                    AND Seat.Se_Train = Train.T_Name
                    AND Seat.Se_Date = '2018-11-20'
                    AND Seat.Se_Station = Train.T_Station
                    AND TS_Num1.TS_fromNum < Train.T_StNum
                    AND Train.T_StNum > TS_Num2.TS_toNum-1
                GROUP BY T_Nonstop2.TN2_Name,Seat.Se_Type
    ),
-- 得到每次列车**有余票**的座位类型、价格及其余票数
    TS_leftseat(TSL_Train,TSL_Type,TSL_Money,TSL_Num) AS
    (
```

```
                SELECT
TS_Seat.TS_Train,TS_Seat.TS_Type,T_MinMoney.TMM_Money,TS_Seat.TS_Num
                FROM TS_Seat,T_MinMoney
                WHERE TS_Seat.TS_Num <> 0
                    AND T_MinMoney.TMM_Name = TS_Seat.TS_Train
                    AND T_MinMoney.TMM_Type = TS_Seat.TS_Type
    ),
-- 获得每次列车的最低票价所对应的座位类型、价格及其余票数
    T_MinMoney2(TMM2_Name,TMM2_Type,TMM2_Money,TMM2_Num) AS
    (
        SELECT TSL_Train,TSL_Type,MIN(TSL_Money),TSL_Num
        FROM TS_leftseat
        GROUP BY TSL_Train,TSL_Type,TSL_Num
    ),
-- 获得出发信息
    T11(T11_Name,T11_St,T11_Time) AS
    (
        SELECT TS_Num1.TS_train,TS_Num1.TS_Station,Train.T_StartTime
        FROM TS_Num1,Train
        WHERE TS_Num1.TS_train = Train.T_Name
            AND TS_Num1.TS_Station = Train.T_Station
    ),
--获得到达信息
    T12(T12_Name,T12_St,T12_Time) AS
    (
        SELECT TS_Num2.TS_train,TS_Num2.TS_Station,Train.T_StartTime
        FROM TS_Num2,Train
        WHERE TS_Num2.TS_train = Train.T_Name
            AND TS_Num2.TS_Station = Train.T_Station
    ),
-- 计算历时
    TTime(TT_Name,AllTime) AS
    (
        SELECT  T11.T11_Name,
        (CASE
            WHEN DATE_PART('hour', T12.T12_Time::time - T11.T11_Time::time) * 60 +
DATE_PART('minute', T12.T12_Time::time - T11.T11_Time::time) > 0
                THEN DATE_PART('hour', T12.T12_Time::time - T11.T11_Time::time) * 60 +
DATE_PART('minute', T12.T12_Time::time - T11.T11_Time::time)
            ELSE DATE_PART('hour', '24:00'::time - T11.T11_Time::time) * 60 +
DATE_PART('minute', '24:00'::time - T11.T11_Time::time) +
                 DATE_PART('hour', T12.T12_Time::time - '00:00'::time) * 60 +
DATE_PART('minute', T12.T12_Time::time - '00:00'::time)
            END
            )AllTime
        FROM T11,T12
        WHERE T11.T11_Name = T12.T12_Name
    )
-- 按照先票价、再行程总时间、最后起始时间排序
    SELECT DISTINCT
TMM2_Name,T11_St,T11_Time,T12_St,T12_Time,TTime.AllTime,TMM2_Type,TMM2_Money,TMM2_Num
    FROM T_MinMoney2,T11,T12,TTime
    WHERE TMM2_Money <> 0
```

```
        AND T_MinMoney2.TMM2_Name = T11.T11_Name
        AND T_MinMoney2.TMM2_Name = T12.T12_Name
        AND T_MinMoney2.TMM2_Name = TTime.TT_Name
    ORDER BY T_MinMoney2.TMM2_Money,TTime.AllTime,T11.T11_Time;
```

3. 换乘一次的查询

输入出发地城市名、到达地城市名、出发日期和出发时间,查询两地之间换乘一次的列车和余票信息。

查询两地之间换乘一次的列车和余票信息的SQL语句如下，其中'北京'、'常州'、'2018-11-23'、'00:00'是可改变的参数。

```
-- 先搜过北京的列车
    WITH T0(T0_Name,T0_StNum) AS
    (
        SELECT Train.T_Name,Train.T_StNum
        FROM Train,Station
        WHERE Train.T_Station = Station.St_Name
            AND St_City = '北京'
    ),
-- 剔除过北京不符合出发时间要求的列车
    T1(T1_Name,T1_StNum) AS
    (
        SELECT T0.T0_Name,T0.T0_StNum
        FROM T0,Train,Station
        WHERE Train.T_Name = T0.T0_Name
            AND Train.T_Station = Station.St_Name
            AND Station.St_City = '北京'
            AND Train.T_StartTime > '00:00'
    ),
-- 再搜过常州的列车
    T2(T2_Name,T2_StNum) AS
    (
        SELECT Train.T_Name,Train.T_StNum
        FROM Train,Station
        WHERE Train.T_Station = Station.St_Name
            AND St_City = '常州'
    ),
-- 搜过北京和过常州的列车
    T3(T3_Name,T3_StNum) AS
    (
        SELECT T1.*
        FROM T1
        UNION
        SELECT T2.*
        FROM T2
    ),
-- 列出筛选后的过北京的列车、站名及站号
    S1(Name1,St1,StNum1,Time1) AS
    (
        SELECT T3.T3_Name,Train.T_Station,T1.T1_StNum,Train.T_StartTime
        FROM T1,T3,Train
        WHERE T1.T1_Name = T3.T3_Name
```

```
            AND Train.T_Name = T3.T3_Name
            AND Train.T_StNum = T1.T1_StNum
    ),
-- 列出筛选后的过常州的列车、站名及站号、时间
    S2(Name2,St2,StNum2,Time2) AS
    (
        SELECT T3.T3_Name,Train.T_Station,T2.T2_StNum,Train.T_ArrivalTime
        FROM T2,T3,Train
        WHERE T2.T2_Name = T3.T3_Name
            AND Train.T_Name = T3.T3_Name
            AND Train.T_StNum = T2.T2_StNum
    ),
-- 搜北京->常州存在换乘站的列车及其换乘站
    -- 北京->终点站的所有
    TEMP1(Name1,St0,St1,StNum1,City1,Time1) AS
    (
        SELECT
Train.T_Name,S1.St1,Train.T_Station,Train.T_StNum,Station.St_City,Train.T_StartTime
        FROM Train,S1,Station
        WHERE Train.T_Name = S1.Name1
            AND Train.T_StNum > S1.StNum1
            AND Station.St_Name = Train.T_Station
            AND Station.St_City != '北京'
    ),
    -- 始发站->常州的所有
    TEMP2(Name2,St0,St2,StNum2,City2,Time2) AS
    (
        SELECT
Train.T_Name,S2.St2,Train.T_Station,Train.T_StNum,Station.St_City,Train.T_ArrivalTime
        FROM Train,S2,Station
        WHERE Train.T_Name = S2.Name2
            AND Train.T_StNum < S2.StNum2
            AND Station.St_Name = Train.T_Station
            AND Train.T_StNum > 1
            AND Station.St_City != '常州'
    ),
    -- 所有同地换乘站及其换乘时间
    T4(Name1,St01,St1,Time1,Name2,St02,St2,Time2,AllTime) AS
    (
        SELECT
TEMP1.Name1,TEMP1.St0,TEMP1.St1,TEMP1.Time1,TEMP2.Name2,TEMP2.St0,TEMP2.St2,TEMP2.Time2
,
        (CASE
            WHEN DATE_PART('hour', TEMP2.Time2::time - TEMP1.Time1::time) * 60 +
DATE_PART('minute', TEMP2.Time2::time - TEMP1.Time1::time) > 0
                THEN DATE_PART('hour', TEMP2.Time2::time - TEMP1.Time1::time) * 60 +
DATE_PART('minute', TEMP2.Time2::time - TEMP1.Time1::time)
            ELSE DATE_PART('hour', '24:00'::time - TEMP1.Time1::time) * 60 +
DATE_PART('minute', '24:00'::time - TEMP1.Time1::time) +
                 DATE_PART('hour', TEMP2.Time2::time - '00:00'::time) * 60 +
DATE_PART('minute', TEMP2.Time2::time - '00:00'::time)
            END
            )AllTime
```

```
        FROM TEMP1,TEMP2
        WHERE TEMP1.City1 = TEMP2.City2
        AND TEMP1.Name1 != TEMP2.Name2
    ),
-- 计算第一段旅程历时
    TTime1(TT_Name,St1,St2,AllTime) AS
    (
        SELECT DISTINCT T4.Name1,T4.St01,T4.St1,
        (CASE
            WHEN DATE_PART('hour', T4.Time1::time - S1.Time1::time) * 60 +
DATE_PART('minute', T4.Time1::time - S1.Time1::time) > 0
                THEN DATE_PART('hour', T4.Time1::time - S1.Time1::time) * 60 +
DATE_PART('minute', T4.Time1::time - S1.Time1::time)
            ELSE DATE_PART('hour', '24:00'::time - S1.Time1::time) * 60 +
DATE_PART('minute', '24:00'::time - S1.Time1::time) +
                 DATE_PART('hour', T4.Time1::time - '00:00'::time) * 60 +
DATE_PART('minute', T4.Time1::time - '00:00'::time)
            END
            )AllTime
        FROM S1,T4
        WHERE S1.Name1 = T4.Name1
    ),
-- 计算第二段旅程历时
    TTime2(TT_Name,St1,St2,AllTime) AS
    (
        SELECT  T4.Name2,T4.St02,T4.St2,
        (CASE
            WHEN DATE_PART('hour', S2.Time2::time - T4.Time2::time) * 60 +
DATE_PART('minute', S2.Time2::time - T4.Time2::time) > 0
                THEN DATE_PART('hour', S2.Time2::time - T4.Time2::time) * 60 +
DATE_PART('minute', S2.Time2::time - T4.Time2::time)
            ELSE DATE_PART('hour', '24:00'::time - T4.Time2::time) * 60 +
DATE_PART('minute', '24:00'::time - T4.Time2::time) +
                 DATE_PART('hour', S2.Time2::time - '00:00'::time) * 60 +
DATE_PART('minute', S2.Time2::time - '00:00'::time)
            END
            )AllTime
        FROM S2,T4
        WHERE S2.Name2 = T4.Name2
    ),
-- 找出符合换乘时间要求的路线
    T5(Name1,St01,St1,Time1,Name2,St02,St2,Time2,AllTime) AS
    (
        SELECT Name1,St01,St1,Time1,Name2,St02,St2,Time2,AllTime
        FROM T4
        WHERE (St1 = St2 AND 60 <= AllTime AND AllTime <= 240)
            OR (St1 != St2 AND 120 <= AllTime AND AllTime <= 240)
        GROUP BY Name1,St01,St1,Time1,Name2,St02,St2,Time2,AllTime
    ),
-- 计算第一段旅程票价
    -- 搜出所有票价（未做减法）
    T_Money1(Name1,St1,St2,YZ,RZ,YW1,YW2,YW3,RW1,RW2) AS
    (
```

```
        SELECT
T5.Name1,T5.St01,T5.St1,Train.T_YZMoney,Train.T_RZMoney,Train.T_YW1Money,Train.T_YW2Mon
ey,Train.T_YW3Money,Train.T_RW1Money,Train.T_RW2Money
        FROM Train,T5
        WHERE Train.T_Name = T5.Name1
            AND (Train.T_Station = T5.St01 OR Train.T_Station = T5.St1)
    ),
    -- 获得每次列车各种座位类型,做减法并获得每次列车各种座位的票价
    T_Type1(Name1,St1,St2,Type1,Money1) AS
    (
        SELECT Name1,St1,St2,CAST('YZ' AS se_type),MAX(T_Money1.YZ)-MIN(T_Money1.YZ)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
        UNION
        SELECT Name1,St1,St2,CAST('RZ' AS se_type),MAX(T_Money1.RZ)-MIN(T_Money1.RZ)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
        UNION
        SELECT Name1,St1,St2,CAST('YW1' AS se_type),MAX(T_Money1.YW1)-MIN(T_Money1.YW1)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
        UNION
        SELECT Name1,St1,St2,CAST('YW2' AS se_type),MAX(T_Money1.YW2)-MIN(T_Money1.YW2)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
        UNION
        SELECT Name1,St1,St2,CAST('YW3' AS se_type),MAX(T_Money1.YW3)-MIN(T_Money1.YW3)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
        UNION
        SELECT Name1,St1,St2,CAST('RW1' AS se_type),MAX(T_Money1.RW1)-MIN(T_Money1.RW1)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
        UNION
        SELECT Name1,St1,St2,CAST('RW2' AS se_type),MAX(T_Money1.RW2)-MIN(T_Money1.RW2)
        FROM T_Money1
        GROUP BY T_Money1.Name1,T_Money1.St1,T_Money1.St2
    ),
    -- 开始算余票
    TS_Num11(train,St1,St2,fromNum) AS
    (
                SELECT Train.T_Name,T5.St01,T5.St1,Train.T_StNum
                From Train,T5
                WHERE Train.T_Name = T5.Name1
                AND Train.T_Station = T5.St01
    ),
    TS_Num21(train,St1,St2,toNum) AS
    (
                SELECT Train.T_Name,T5.St01,T5.St1,Train.T_StNum
                From Train,T5
                WHERE Train.T_Name = T5.Name1
                AND Train.T_Station = T5.St1
    ),
```

```
    TS_Num1(train,St1,St2,fromNum,toNum) AS
    (
                SELECT
TS_Num11.train,TS_Num11.St1,TS_Num11.St2,TS_Num11.fromNum,TS_Num21.toNum
                FROM TS_Num11,TS_Num21
                WHERE TS_Num11.train = TS_Num21.train
                    AND TS_Num21.St1 = TS_Num11.St1
                    AND TS_Num21.St2 = TS_Num11.St2
    ),
    -- 得到每次列车的座位类型及其余票
    TS_Seat1(Train1,St1,St2,Type1,Num1) AS
    (
                SELECT
TS_Num1.train,TS_Num1.St1,TS_Num1.St2,Seat.Se_Type,MIN(Seat.Se_Num)
                FROM Train,Seat,TS_Num1
                WHERE Train.T_Name = TS_Num1.train
                    AND Seat.Se_Train = Train.T_Name
                    AND Seat.Se_Date = '2018-11-23'
                    AND Seat.Se_Station = Train.T_Station
                    AND TS_Num1.fromNum < Train.T_StNum
                    AND Train.T_StNum > TS_Num1.toNum-1
                GROUP BY TS_Num1.train,TS_Num1.St1,TS_Num1.St2,Seat.Se_Type
    ),
    -- 得到每次列车**有余票**的座位类型、价格及其余票数
    TS_leftseat1(Train1,St1,St2,Type1,Money1,Num1) AS
    (
                SELECT
TS_Seat1.Train1,TS_Seat1.St1,TS_Seat1.St2,TS_Seat1.Type1,T_Type1.Money1,TS_Seat1.Num1
                FROM TS_Seat1,T_Type1
                WHERE TS_Seat1.Num1 <> 0
                    AND T_Type1.Money1 <> 0
                    AND T_Type1.Name1 = TS_Seat1.Train1
                    AND T_Type1.Type1 = TS_Seat1.Type1
    ),
    -- 获得每次列车的最低票价所对应的座位类型、价格及其余票数
    T_MinMoney21(Name1,St1,St2,Type1,Money1,Num1) AS
    (
        SELECT Train1,St1,St2,Type1,MIN(Money1),Num1
        FROM TS_leftseat1
        GROUP BY Train1,Type1,Num1,St1,St2
    ),
-- 计算第二段旅程票价
    -- 搜出所有票价（未做减法）
    T_Money2(Name2,St1,St2,YZ,RZ,YW1,YW2,YW3,RW1,RW2) AS
    (
        SELECT
T5.Name2,T5.StO2,T5.St2,Train.T_YZMoney,Train.T_RZMoney,Train.T_YW1Money,Train.T_YW2Mon
ey,Train.T_YW3Money,Train.T_RW1Money,Train.T_RW2Money
        FROM Train,T5
        WHERE Train.T_Name = T5.Name2
            AND (Train.T_Station = T5.StO2 OR Train.T_Station = T5.St2)
    ),
    -- 获得每次列车各种座位类型,做减法并获得每次列车各种座位的票价
```

```
    T_Type2(Name2,St1,St2,Type2,Money2) AS
    (
        SELECT Name2,St1,St2,CAST('YZ' AS se_type),MAX(T_Money2.YZ)-MIN(T_Money2.YZ)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
        UNION
        SELECT Name2,St1,St2,CAST('RZ' AS se_type),MAX(T_Money2.RZ)-MIN(T_Money2.RZ)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
        UNION
        SELECT Name2,St1,St2,CAST('YW1' AS se_type),MAX(T_Money2.YW1)-MIN(T_Money2.YW1)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
        UNION
        SELECT Name2,St1,St2,CAST('YW2' AS se_type),MAX(T_Money2.YW2)-MIN(T_Money2.YW2)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
        UNION
        SELECT Name2,St1,St2,CAST('YW3' AS se_type),MAX(T_Money2.YW3)-MIN(T_Money2.YW3)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
        UNION
        SELECT Name2,St1,St2,CAST('RW1' AS se_type),MAX(T_Money2.RW1)-MIN(T_Money2.RW1)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
        UNION
        SELECT Name2,St1,St2,CAST('RW2' AS se_type),MAX(T_Money2.RW2)-MIN(T_Money2.RW2)
        FROM T_Money2
        GROUP BY T_Money2.Name2,T_Money2.St1,T_Money2.St2
    ),
    -- 开始算余票
    TS_Num12(train,St1,St2,fromNum) AS
    (
                SELECT Train.T_Name,T5.StO2,T5.St2,Train.T_StNum
                FROM Train,T5
                WHERE Train.T_Name = T5.Name2
                AND Train.T_Station = T5.St2
    ),
    TS_Num22(train,St1,St2,toNum) AS
    (
                SELECT Train.T_Name,T5.StO2,T5.St2,Train.T_StNum
                FROM Train,T5
                WHERE Train.T_Name = T5.Name2
                AND Train.T_Station = T5.StO2
    ),
    TS_Num2(train,St1,St2,fromNum,toNum) AS
    (
                SELECT
TS_Num12.train,TS_Num12.St1,TS_Num12.St2,TS_Num12.fromNum,TS_Num22.toNum
                FROM TS_Num12,TS_Num22
                WHERE TS_Num12.train = TS_Num22.train
                    AND TS_Num22.St1 = TS_Num12.St1
                    AND TS_Num22.St2 = TS_Num12.St2
```

```
    ),
    -- 得到每次列车的座位类型及其余票
    TS_Seat2(Train2,St1,St2,Type2,Num2) AS
    (
                SELECT
TS_Num2.train,TS_Num2.St1,TS_Num2.St2,Seat.Se_Type,MIN(Seat.Se_Num)
                FROM Train,Seat,TS_Num2,T5
                WHERE Train.T_Name = TS_Num2.train
                    AND Seat.Se_Train = Train.T_Name
                    AND Seat.Se_Date = '2018-11-23'
                    AND Seat.Se_Station = Train.T_Station
                    AND TS_Num2.fromNum < Train.T_StNum
                    AND Train.T_StNum > TS_Num2.toNum-1
                GROUP BY TS_Num2.train,TS_Num2.St1,TS_Num2.St2,Seat.Se_Type
    ),
    -- 得到每次列车**有余票**的座位类型、价格及其余票数
    TS_leftseat2(Train2,St1,St2,Type2,Money2,Num2) AS
    (
                SELECT
TS_Seat2.Train2,TS_Seat2.St1,TS_Seat2.St2,TS_Seat2.Type2,T_Type2.Money2,TS_Seat2.Num2
                FROM TS_Seat2,T_Type2
                WHERE TS_Seat2.Num2 <> 0
                    AND T_Type2.Money2 <> 0
                    AND T_Type2.Name2 = TS_Seat2.Train2
                    AND T_Type2.Type2 = TS_Seat2.Type2
    ),
    -- 获得每次列车的最低票价所对应的座位类型、价格及其余票数
    T_MinMoney22(Name2,St1,St2,Type2,Money2,Num2) AS
    (
        SELECT Train2,St1,St2,Type2,MIN(Money2),Num2
        FROM TS_leftseat2
        GROUP BY Train2,Type2,Num2,St1,St2
    ),
-- 获得两次列车的最低票价所对应的座位类型、价格及其余票数，并将两最低价格相加(附加时间)

T_MinMoneyFinal(Name1,St11,Time11,St12,T12,Type1,Money1,Num1,Name2,St21,Time21,St22,Tim
e22,Type2,Money2,Num2,AllTime,MoneySum) AS
    (
        SELECT
T_MinMoney21.Name1,T_MinMoney21.St1,S1.Time1,T_MinMoney21.St2,T5.Time1,T_MinMoney21.Typ
e1,T_MinMoney21.Money1,T_MinMoney21.Num1,

T_MinMoney22.Name2,T_MinMoney22.St1,S2.Time2,T_MinMoney22.St2,T5.Time2,T_MinMoney22.Typ
e2,T_MinMoney22.Money2,T_MinMoney22.Num2,
                (CASE
                    WHEN DATE_PART('hour', S2.Time2::time - S1.Time1::time) * 60 +
DATE_PART('minute', S2.Time2::time - S1.Time1::time) > 0
                    THEN DATE_PART('hour', S2.Time2::time - S1.Time1::time) * 60 +
DATE_PART('minute', S2.Time2::time - S1.Time1::time)
                    ELSE DATE_PART('hour', '24:00'::time - S1.Time1::time) * 60 +
DATE_PART('minute', '24:00'::time - S1.Time1::time) +
                    DATE_PART('hour', S2.Time2::time - '00:00'::time) * 60 +
DATE_PART('minute', S2.Time2::time - '00:00'::time)
```

```
                    END
                )AllTime,
                T_MinMoney21.Money1 + T_MinMoney22.Money2
        FROM T_MinMoney21,T_MinMoney22,T5,S1,S2
        WHERE T_MinMoney21.Name1 = T5.Name1
            AND T_MinMoney21.St1 = T5.St01
            AND T_MinMoney21.St2 = T5.St1
            AND T_MinMoney22.Name2 = T5.Name2
            AND T_MinMoney22.St1 = T5.St02
            AND T_MinMoney22.St2 = T5.St2
            AND S1.Name1 = T5.Name1
            AND S1.St1 = T5.St01
            AND S2.Name2 = T5.Name2
            AND S2.St2 = T5.St02
    )
     SELECT DISTINCT *
     FROM T_MinMoneyFinal
     ORDER BY MoneySum,AllTime,Time11;
```

4. 查询订单和删除订单

给定出发日期范围，显示订单列表。

订单具体信息：订单号、日期、出发到达站、总票价和订单状态（是否已经取消）。

查询订单的SQL语句如下，其中'ZDj9XWLR'、'2017-11-17'、'2018-11-17'是可改变的参数。

```
SELECT B.B_ID
FROM Book,User
WHERE B.B_User = U.ICNum
    AND U.UName = 'ZDj9XWLR'
    AND B.B_Date >= '2017-11-17'
    AND B.B_Date <= '2018-11-17';
```

查询每个订单详细信息的SQL语句如下，其中100是可改变的参数。

```
SELECT B_Id,B_Date,B_StartSt,B_ArrivalSt,B_Money,B_Status
FROM Book
WHERE B_Id = 100;
```

```
-- 获得每次列车的最低票价
WITH T_MinMoney2(TMM2_Name,TMM2_Money) AS
(
    SELECT TMM_Name,
            (SELECT MAX(TEMPDATA)
            FROM  (VALUES((TMM_YZ),(TMM_RZ),(TMM_YW1),(TMM_YW2),(TMM_YW3),(TMM_RW1),
(TMM_RW2)) AS value(TEMPDATA))
            )
    FROM T_MinMoney
    GROUP BY TMM_Name
)
```

```
-- 获得票价最低的十次列车(按照先票价、再行程总时间、最后起始时间排序)
SELECT T_MinMoney2.TMM2_Name,T_MinMoney2.TMM2_Money
FROM T_MinMoney2,Train
WHERE T_MinMoney2.TMM2_Name = Train.T_Name
ORDER BY T_MinMoney2.TMM2_Money,Train,Train;
```

用户问题

实际上除了给出的基本需求，还自己添加了一个“游客”登录途径，方便测试。

游客的实现操作很简单，实际上就是随机生成符合要求的信息，然后插入表中，和普通注册用户有同样的权限，只是姓名会统一为“游客”（因为允许重名）。

具体实现如下：

```
function str_rand_6($length = 6, $char =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ') {
if(!is_int($length) || $length < 0) {
    return false;
}

$string = '';
for($i = $length; $i > 0; $i--) {
    $string .= $char[mt_rand(0, strlen($char) - 1)];
}

return $string;

$name="游客";
$userid=num_rand_18();
$usertel=num_rand_11();
$cardid=num_rand_16();
$username=str_rand_8();
$password=str_rand_6();

}
```

补充说明

余票问题

1. 如何处理余票默认值为5

首先在需求分析中已经阐述，每个车次每天每个站点所有可提供的座位类型（即CSV表中有票价的座位类型）默认有5张票。

但是在初始查询的时候，是没有建立座位的表的，也就是说我们进行余票搜索得出来的结果为none，这样无法进行min等SQL操作。但是如果提前建好表填入默认值，由于日期不定，无法获得日期范围，这个表是无限大的，因此并不实际。

最后我们采取的方案是，在查询之后，对于我们将要获得余票信息的相应作为信息进行搜索判断，是否为空没有建表？如果是，就建表并默认余票为5，否则不用建表直接查询。

2. 如何处理余票增减

我们在预定成功时，对于余票进行-1操作。但是并不是出发站余票-1，而是根据实际情况，一旦购买此票，这个座位在起始站到到达站都是售出的状态，因此从from_station到to_station（不含to_station）的余票都要进行-1操作。

同理，在取消订单的时候，要将起始站到到达站的相应余票都+1.这里其实是运用T_Name，T_station和T_stnum一一对应，根据站号来进行的。

3. 如何进行余票查询

查询余票输出的时候，并不是获取当前站的余票，而是搜索出发站到到达站之间余票的最小值，即是：即便这之间其他站都有票，但只要有一站没票就意味着你在那一站无法获得座位，因此你无法购买这张票在同一个座位上乘坐到目的地。

基本实现逻辑如下：

```
SELECT MIN(Seat.Se_Num)
FROM Train,Seat
WHERE Train.T_Name = '$trainid'
AND Seat.Se_Train = Train.T_Name
AND Seat.Se_Date = '$train_date'
AND Seat.Se_Type = '$type'
AND Seat.Se_Station = Train.T_Station
AND Train.T_StNum BETWEEN $from_stnum AND $to_stnum;
```

日期问题

1. 日期链接余票

考虑到时刻表每天都是一样的，那么可以认为车辆连续运行时间不会超过24小时。因此我们其他信息并不会因为日期这一属性有任何改变，唯一会受到日期影响的就是余票。

因此，我们日期的主要作用在于查询座位余票，即在seat表中，需要根据日期进行动态查询。

2. 跨天问题

既然一列车连续运行时间不会超过24小时，那么如果 到达时间-出发时间<0 ，就表示运行时出现了跨天。而跨天主要

- 影响换乘时的余票问题
- 影响订单输出的到达日期问题

这样进行判断了之后，如果跨天就+1,否则就是当天到达，即是出发日期。