数据库第二次实验设计文档

中国科学院大学
乔怿凌 陈乐滢 付琳晴
2017.11.30

一、ER 图

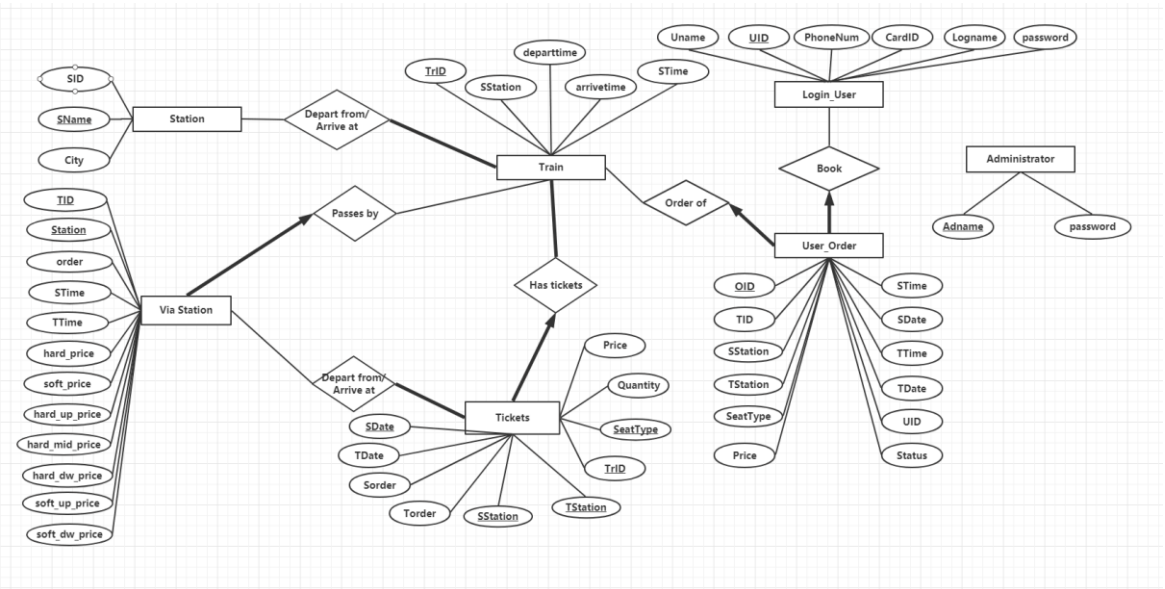

图 1. 总 ER 图

二、TABLE 分析

2.1 Station

```
create table Station   (
                        S_ID        integer unique,
                        S_name      varchar(20) primary key,
                        S_city      varchar(20) not null
                        );
```

图 2. Create table Station 代码

S_ID 是车站的 ID，用整型表示，是唯一的。

S_name 是车站的名字，作为 station 的主键。用它而非 S_ID 作为主键的原因是，在不需要通过城市查询的时候不用连接 station 表，可以提升部分性能。

S_city 是车站的城市，非空。

2.2 Login_User

```
create table Login_User (
    U_name      varchar(20) not null unique,
    U_ID        char(18) primary key,
    U_phonenum  char(11) unique,
    U_cardID    char(16) not null,
    U_logname   varchar(20) not null,
    U_password  varchar(20) not null
    );
```

图 3. Create table Login_User 代码

U_name 是用户的真实姓名，U_ID 是用户的身份证号，U_phonenum 是用户的手机号，U_cardID 是用户的信用卡号，U_logname 是用户的用户名，U_password 是用户的密码。

2.3 Train

```
create table Train      (
  Tr_ID      varchar(8) primary key,
  Tr_SStation varchar(20) not null,
  Tr_TStation varchar(20) not null,
  Tr_departtime time not null,
  Tr_arrivetime time not null,
  foreign key (Tr_SStation) references Station(S_name),
  foreign key (Tr_TStation) references Station(S_name)
);
```

图 4. Create table Train 代码

Tr_ID 是车次的 ID，Tr_SStation 是车次始发站的站名，Tr_TStation 是车次终点站的站名，Tr_departtime 是车次的发车时间，Tr_arrivetime 是车次的到站时间，其中 Tr_SStation 和 Tr_TStation 是来自 Station 的外键。

2.4 Via_Station

```
create table Via_Station   (
 V_TID              varchar(8) not null,
 V_order            integer not null,
 V_Station          varchar(20) not null,
 V_TTime            time not null,
 V_STime            time not null,
 V_hard_price       decimal(15,2) not null,
 V_soft_price       decimal(15,2) not null,
 V_hard_up_price    decimal(15,2) not null,
 V_hard_mid_price   decimal(15,2) not null,
 V_hard_dw_price    decimal(15,2) not null,
 V_soft_up_price    decimal(15,2) not null,
 V_soft_dw_price    decimal(15,2) not null,
 foreign key (V_TID) references Train(Tr_ID),
 foreign key (V_Station) references Station(S_name),
 primary key(V_TID, V_Station)
 );
```

图 5. Create table Via_Station 代码

V_TID 是该经停站所在车次的 ID，V_order 是该经停站在车次中的站序，V_Station 是该车

次的站名，V_TTime 是该站的到站时间，V_STime 是该站的出发时间，V_hard_price-V_soft_dw_price 分别是该车次起始站到该经停站硬座、软座、硬卧上、硬卧中、硬卧下、软卧上、软卧下的票价。

其中 V_TID 和 V_Station 组合形成主键，V_TID 是来自 Train 的外键，V_Station 是来自 Station 的外键。

2.5 Tickets

```
create table Tickets  (
                      Ti_TrID       varchar(8) not null,
                      Ti_Sorder     integer not null,
                      Ti_Torder     integer not null,
                      Ti_SStation   varchar(20) not null,
                      Ti_TStation   varchar(20) not null,
                      Ti_SeatType   varchar(20) not null,
                      Ti_price      decimal(15,2) not null,
                      Ti_quantity   integer not null,
                      Ti_SDate      date not null,
                      Ti_TDate      date not null,
                      primary key (Ti_TrID, Ti_TStation, Ti_SStation, Ti_SeatType, Ti_SDate),
                      foreign key (Ti_TrID) references Train(Tr_ID),
                      foreign key (Ti_SStation) references Station(S_name),
                      foreign key (Ti_TStation) references Station(S_name)
                      );
```

图 6. Create table Tickets 代码

Ti_TrID 是余票所在车次的 ID，Ti_Sorder 是余票起始站在车次中的站序，Ti_Torder 是余票终点站在该车次中的站序，Ti_SStation 是余票起始站的站名，Ti_TStation 是余票终点站的站名，Ti_SeatType 是余票的座型，Ti_price 是余票的价格，Ti_quantity 是余票的数量，Ti_SDate 是余票的发车日期，Ti_TDate 是余票的到达日期（可能跨夜）。

Ti_TrID、Ti_TStation、Ti_SStation、Ti_SeatType、Ti_SDate 共同作为 Tickets 的主键，Ti_TrID 是来自 Train 的外键，Ti_SStation、Ti_TStation 是来自 Station 的外键。

2.6 User_Order

```
create table User_Order    (
                           O_ID           char(18) primary key,
                           O_TID          varchar(8) not null,
                           O_SStation     varchar(20) not null,
                           O_TStation     varchar(20) not null,
                           O_seattype     varchar(20) not null,
                           O_price        decimal(15,2) not null,
                           O_STime        time not null,
                           O_SDate        date not null,
                           O_TTime        time not null,
                           O_TDate        date not null,
                           O_UID          char(18) not null,
                           O_status       varchar(20) not null,
                           foreign key (O_UID) references Login_User(U_ID),
                           foreign key (O_TID) references Train(Tr_ID),
                           foreign key (O_SStation) references Station(S_name),
                           foreign key (O_TStation) references Station(S_name)
                           );
```

图 7. Create table User_Order 代码

O_ID 是订单 ID，由 PHP 从时间戳生成，O_TID 是订单对应的车次，O_SStation 是订

单对应的起始站，O_TStation 是订单对应的终点站，O_seattype 是订单对应的座位类型，O_price 是订单对应的票价，O_STime 和 O_SDate 分别是订单对应票的出发时间和日期，O_TTime 和 O_TDate 分别是订单对应票的到达时间和日期，O_UID 是订单确认者的身份证号，O_status 是订单状态。

O_ID 是订单的主键，O_UID 是来自 Login_User 的外键，O_TID 是来自 Train 的外键，O_SStation、O_TStation 是来自 Station 的外键。

2.7 Administrator

```
create table Administrator (
                    Ad_name     varchar(20) primary key,
                    Ad_password varchar(20) not null
                    );
```

图 8. Create table Administrator 代码

Ad_name 是管理员的用户名，作为 Administrator 的主键；Ad_password 是管理员的密码。

2.8 数据处理说明

以上 table 的数据通过 python 处理，即用 python 抓取每一个文件的信息（包括文件名），处理后将 Insert 语句或者 csv 的信息写入目标文件，再将目标文件导入虚拟机中对应的 table。

处理 Tickets 表中过夜车的到达日期时，将行程中每两站之间经过的天数和每一站经停时经过的天数加起来，即为在火车上经过的总天数。判断是否过夜是通过前一个时间与零点的差和后一个时间与零点的差值得大小关系来判断的。

三、模式细化

3.1 2NF

除了 Tickets 均达到 2NF。

Station：候选键有两个：SID 和 SNAME，他们分别只有一个成员，显然达到 2NF。

Train：只有一个候选键 Tr_ID，显然达到 2NF。

Administrator：只有一个候选键 Ad_name，显然。

Login_user：候选键 UID、phonenum 都只有一个成员，显然。

Via_Station：主键是（V_TID 和 V_Station），主键中任意一个成员都无法独立确认其他任意一个非键属性。

User_Order：主键是 O_ID，没有其他候选键。

Tickets：当属性 SStation 或 TStation 确定时，其 Sorder 或 Torder 已经确定，存在部分依赖，因此不是 2NF。

3.2 3NF

除了 Tickets 均达到 3NF，其他表的所有包含非键属性的集合都无法确定一个非键属性。

四、需求分析

4.1 记录列车信息、座位情况、乘客信息

需求 1、2、3 均是分别将数据导入对应的表即可，列车信息存在 Train 表，座位情况存

在 Tickets 表，乘客信息存在 Login_User 表。在界面上每当用户注册一个账号，即一条 INSERT 语句，如图 9. 登录时，用户输入用户名和密码，语句进行查询，使用 count 语句，Login_User 中若有这个用户的信息，则返回 1，登陆成功，若不存在该用户，则返回 0，登录失败。

```
insert into Login_User (U_name, U_ID, U_phonenum, U_cardID, U_logname, U_password)
values (username, ID, phonenum, cardID, logname, password);

eg:
insert into Login_User (U_name, U_ID, U_phonenum, U_cardID, U_logname, U_password)
values ('Lynn', '410303100012345678', '13938836839', '123456789123456', 'qing', '970927');
```

图 9. INSERT 语句和例子

4.4 查询具体车次

需求 4 输入车次序号和查询日期，将得到车次途径的所有站的具体信息，包括：每一站的到达时间和发车时间、七种不同座位的票价，还需要得到从始发站到终点站的七种不同座位的余票。因此该需求我们使用了两条 query，产生两个表，一个表用来显示车次静态信息，另一个用来显示余票数。

首先，由于我们在数据处理时已经将车次的静态信息全部存在 Via_Station 表里，因此查询车次的静态信息只需要输入车次 ID，打印出 Via_Station 表所有 TID 等于查询车次的项的信息即可，query 语句如图 10.

```
select  Via_Station.V_order, Via_Station.V_Station,  Via_Station.V_TTime, Via_Station.V_STime,
        Via_Station.V_hard_price,
        Via_Station.V_soft_price,
        Via_Station.V_hard_up_price,
        Via_Station.V_hard_mid_price,
        Via_Station.V_hard_dw_price,
        Via_Station.V_soft_up_price,
        Via_Station.V_soft_dw_price
from Via_Station
where
    Via_Station.V_TID = trainid
order by Via_Station.V_order asc;
```

图 10. 查询车次静态信息

查询余票用到的是 Tickets 表，需要输入查询日期和查询车次。Tickets 表中存有不同日期不同座位类型的所有票，我们根据始发站终点站、车次 ID 和日期打出座位类型和对应的余票，query 如图 11.

```
#tickets left
select Tickets.Ti_TrID, Tickets.Ti_SeatType, Tickets.Ti_quantity
from Train, Tickets
where
    Train.Tr_ID = trainid and
    Tickets.Ti_TrID = Train.Tr_ID and
    Tickets.Ti_SStation = Train.Tr_SStation and Tickets.Ti_TStation = Train.Tr_TStation and
    Tickets.Ti_SDate = departdate and
    (Tickets.Ti_SeatType = 'hardseat' or Tickets.Ti_SeatType = 'softseat' or
    Tickets.Ti_SeatType = 'hardbedu' or Tickets.Ti_SeatType = 'hardbedm' or
    Tickets.Ti_SeatType = 'hardbedd' or Tickets.Ti_SeatType = 'softbedu' or
    Tickets.Ti_SeatType = 'softbedd');
```

图 11. 查询余票动态信息

4.5 查询两地之间的车次

需求 5 输入出发地、到达地、日期和出发时间，查询满足条件的直达列车信息，查询满足条件的换乘一次的乘车方案，分别按照票价选出最低的 10 种。

查询直达列车很简单，我们在处理数据时已经将任何两个车站之间存在的车票都存在了 Tickets 表中，查询时输入出发城市和到达城市，需要用到 Station 中的出发地的车站和到达地的车站，因此需要将 Station 重命名为两个表，用来分别连接出发地和到达地，需要把 Station 表和 Tickets 表连接，连接条件是 Station.S_name = Tickets.SStation. 约束条件有：发车时间大于查询的时间界限，余票数大于 0。Query 语句如图 12.

```
select Tickets.Ti_TrID, Tickets.Ti_SStation, Tickets.Ti_TStation, V1.V_STime, V2.V_TTime,
       Tickets.Ti_SeatType, Tickets.Ti_quantity, Tickets.Ti_price
from Tickets, Station as S1, Station as S2, Via_Station as V1, Via_Station as V2
where S1.S_city = departcity and S2.S_city = arrivecity and
    Tickets.Ti_SStation = S1.S_name and Tickets.Ti_TStation = S2.S_name and
    Tickets.Ti_TrID = V1.V_TID and Tickets.Ti_TrID = V2.V_TID and
    Tickets.Ti_SStation = V1.V_Station and Tickets.Ti_TStation = V2.V_Station and
    Tickets.Ti_SDate = departdate and V1.V_STime >= departtime and
    V1.V_Station = S1.S_name and V2.V_Station = S2.S_name and
    Tickets.Ti_quantity > 0
order by Tickets.Ti_price asc
limit 10;
```

图 12. 查询直达列车

换乘一次需要两张车票，其中第一张车票的终点站所在城市等于第二张车票的始发站所在城市。换乘时间也有约束：换乘地是同一车站则换乘经停时间在一小时到四小时之间；换乘地是同城的不同车站则换乘经停时间在两小时到四小时之间。其中若是隔夜车次，则麻烦一些，要分情况讨论。若两张车票在同一天，则第二张票的发车时间一定大于第一张票的到达时间，我们直接用这两个时间的差即可。若两张车票不在同一天，则存在一个减法溢出的问题，需要多加一个约束条件：要求第二张票的发车时间小于第一张票的到达时间，这样就保证了能查到隔夜车次。

```
###### 50s
select T1.Ti_TrID, T1.Ti_SStation, T1.Ti_TStation, T1.Ti_SDate, V1.V_STime, T1.Ti_TDate, V2.V_TTime,
       T1.Ti_SeatType, T1.Ti_quantity, T1.Ti_price,

       T2.Ti_TrID, T2.Ti_SStation, T2.Ti_TStation, T2.Ti_SDate, V3.V_STime, T2.Ti_TDate, V4.V_TTime,
       T2.Ti_SeatType, T2.Ti_quantity, T2.Ti_price,
       (T1.Ti_price + T2.Ti_price) as pricesum

from Tickets as T1, Tickets as T2,
    Station as S1, Station as S2, Station as S3, Station as S4,
    Via_Station as V1, Via_Station as V2, Via_Station as V3, Via_Station as V4

where S1.S_city = '郑州' and S4.S_city = '北京' and
    T1.Ti_SDate = '2017-12-01' and V1.V_STime >= '00:00' and

    T1.Ti_TrID = V1.V_TID and T1.Ti_TrID = V2.V_TID and
    T2.Ti_TrID = V3.V_TID and T2.Ti_TrID = V4.V_TID and
    T1.Ti_SStation = S1.S_name and T1.Ti_TStation = S2.S_name and
    T2.Ti_SStation = S3.S_name and T2.Ti_TStation = S4.S_name and

    S2.S_city = S3.S_city and
    V1.V_Station = S1.S_name and V2.V_Station = S2.S_name and
    V3.V_Station = S3.S_name and V4.V_Station = S4.S_name and
    T1.Ti_quantity > 0 and
    T2.Ti_quantity > 0 and


    ((T2.Ti_SDate = T1.Ti_TDate and V3.V_STime - V2.V_TTime <= '04:00:00' and
         (S2.S_ID  = S3.S_ID and V3.V_STime - V2.V_TTime >= '01:00:00'
       or S2.S_ID != S3.S_ID and V3.V_STime - V2.V_TTime >= '02:00:00'))

    or

    (T2.Ti_SDate > T1.Ti_TDate and  V3.V_STime < V2.V_TTime and V3.V_STime - V2.V_TTime <= '04:00:00'  and
         (S2.S_ID  = S3.S_ID and V3.V_STime  - V2.V_TTime >= '01:00:00'
       or S2.S_ID != S3.S_ID and V3.V_STime  - V2.V_TTime >= '02:00:00')))
order by pricesum asc
limit 10;
```

图 13. 换乘车次查询

4.6 查询返程信息

需求 6 主要由前端完成，利用的 query 还是需求 5 的语句。

4.7 预定车次座位

在需求 5 中查找到满意车次后即可点击订票，订票时，显示出预定车次的具体信息，需要一个简单的 select 语句，需要在总票价中加上订票费，一辆车 5 元。

```
select Tickets.Ti_TrID, Tickets.Ti_Sorder, Tickets.Ti_Torder, Tickets.Ti_SDate, V1.V_STime, Tickets.Ti_SStation,
       Tickets.Ti_TDate, V2.V_TTime, Tickets.Ti_TStation,
       Tickets.Ti_SeatType, Tickets.Ti_price, Tickets.Ti_price + 5 as totalprice

from Tickets, Via_Station as V1, Via_Station as V2

where Tickets.Ti_TrID = trainid and Tickets.Ti_SeatType = seattype and
    Tickets.Ti_TrID = V1.V_TID and Tickets.Ti_TrID = V2.V_TID and
    Tickets.Ti_SeatType = seattype and
    Tickets.Ti_SDate = departdate and
    Tickets.Ti_SStation = dstop and Tickets.Ti_TStation = astop and
    V1.V_Station = Tickets.Ti_SStation and V2.V_Station = Tickets.Ti_TStation;
```

图 14. 预定车次的显示

选择确认后，需要将订单加到当前用户的订单列表中，即需要 insert 一项到 User_Order 表中，并且由于订了票，所有经过这张票的途径地的余票 都要减一，因此还要更新 Tickets 表。INSERT 语句如图 15. UPDATE 语句如图 16.

```
insert into User_Order
values (orderID, TrID, dstop, astop, seattype, totalprice, stime, sdate, ttime, tdate, uid, 'confirmed');
```

图 15. INSERT 语句

```
update Tickets
set Ti_quantity = Ti_quantity - 1
where
    Tickets.Ti_TrID = trainid and
    (Tickets.Ti_Sorder <=
    (select min(Tickets.Ti_Torder) from Tickets where Tickets.Ti_TrID = trainid and Tickets.Ti_SStation = dstop and Tickets.Ti_TStation = astop)

    or Tickets.Ti_Torder >=
    (select min(Tickets.Ti_Sorder) from Tickets where Tickets.Ti_TrID = trainid and Tickets.Ti_SStation = dstop and Tickets.Ti_TStation = astop)) and
    Tickets.Ti_SeatType = seattype and Tickets.Ti_SDate = departdate;
```

图 16. UPDATE 语句

4.8 查询订单和删除订单

需求 8 传入当前用户的 UID，在 User_Order 表中查询该用户的订单，并且可以选择取消订单。查询订单很容易，只需要在 User_Oder 中根据用户 ID 查询即可。若用户选择取消订单，则需要更新该订单状态改为 canceled，并且将这张票还回余票中，即需要 UPDATE Tickets 表，UPDATE 语句和需求 7 中一样，只不过是符合条件的余票都加 1.

4.9 管理员操作

管理员可以接触到所有的信息，即可以查看所有的表，所有的订单。本需求中，在管理员表中插入一条管理员的登录信息，管理员登陆后需要查看总订单数，即 COUNT 语句对 User_Order 中所有表项求和，查看订单总票价，需要将所有状态为 confirmed 的票的价钱相加，使用 SUM 语句进行求和。对于最热点的车次排序，用到 GROUP BY 语句，根据车次号进行统计，最后按数量从大到小排列。查看注册的用户列表只需要打印出 Login_User 的所有项，点击其中一个用户，在 User_Order 表中根据用户 ID 查找，就可以查看这个用户的订单。所有的 query 如图 17.

```sql
select count(*)
from User_Order
where O_status = 'confirmed';

select sum(O_price)
from User_Order
where O_status = 'confirmed';

select User_Order.O_TID, count(*) as Num
from User_Order
where O_status = 'confirmed'
group by O_TID
order by Num desc;

select Login_User.U_name, Login_User.U_logname
from Login_User;

select Login_User.U_name, User_Order.O_ID, User_Order.O_SDate, User_Order.O_SStation,
       User_Order.O_TStation, User_Order.O_price, User_Order.O_status
from Login_User, User_Order
where
    Login_User.U_name = username and
    Login_User.U_ID = User_Order.O_UID;
```

图 17. 管理员操作的查询语句