数据库系统实验 2 设计报告

高梓源　2019K8009929026

肖展琪　2019K8009926014

桂庭辉　2019K8009929019

1　模型设计

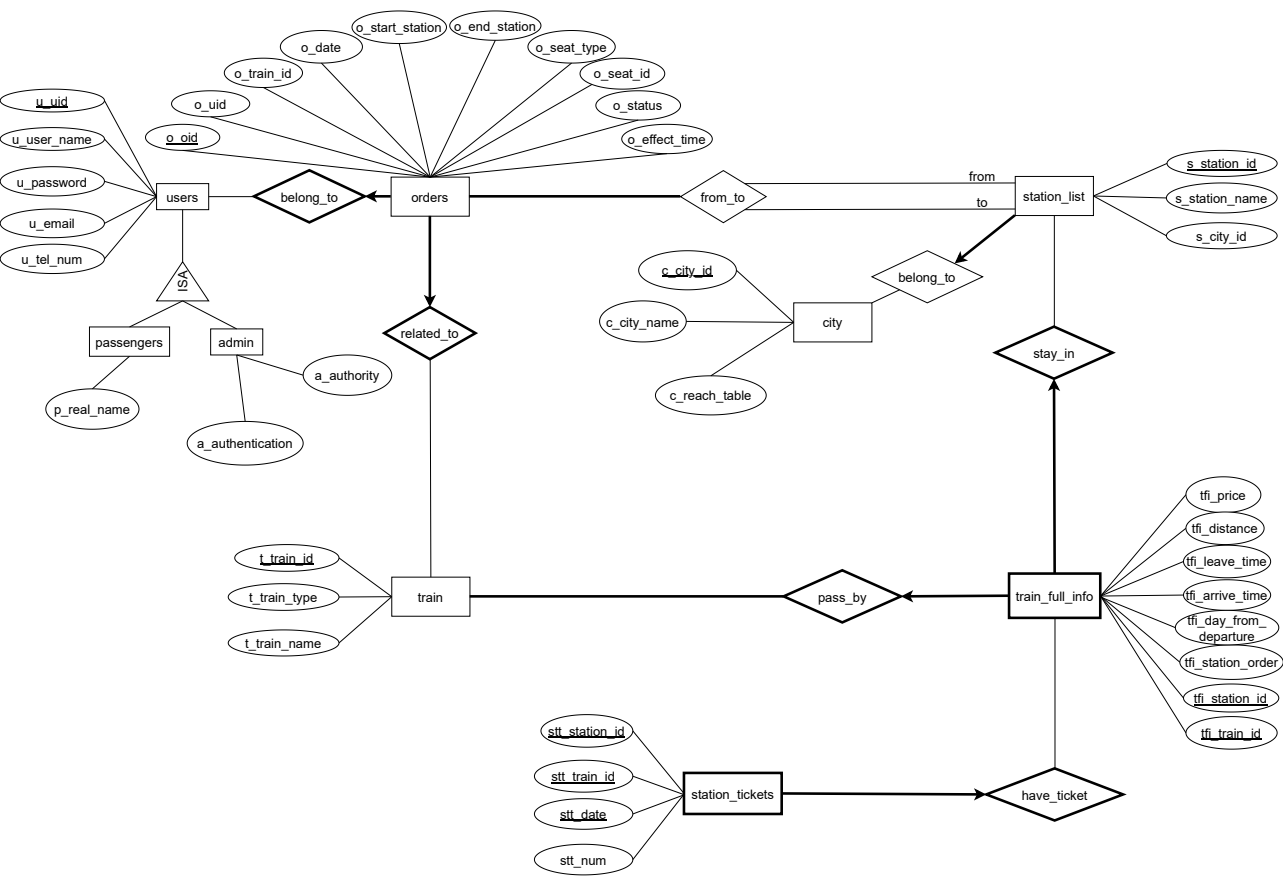## 1.1　ER 图与范式分析

图 1: 火车订票系统 ER 图

1.1.1　users

候选键：uid，user_name，tel_num。作为非键属性的 password 与 email 没有其他的依赖关系。故 users 满足 BCNF。

passengers 中候选键为 pid，函数依赖关系仅有 pid→real_name，故 passengers 满足 BCNF。

admin 中候选键为 aid，函数依赖关系仅有 aid→authority authentication，故 admin 满足 BCNF。

```
1  create table if not exists users (
2      u_uid       serial primary key,
3      u_user_name varchar(20) unique,
4      u_password  varchar(20) not null,
5      u_email     varchar(20) not null,
6      u_tel_num   integer[11] unique
7  );
8
9  create table if not exists passengers (
10     p_pid   integer not null,
11     p_real_name varchar(20) not null,
12     primary key (p_pid),
13     foreign key (p_pid) references users (u_uid)
14 );
15
16 create table if not exists admin (
17     a_aid           integer not null,
18     a_authentication varchar(20) not null ,
19     a_authority     admin_authority not null ,
20     primary key (a_aid),
21     foreign key (a_aid) references users (u_uid)
22 );
```

1.1.2　orders

主键为 oid，函数依赖关系仅有由 oid 确定其他非键属性，故 orders 满足 BCNF。

```
1  create table if not exists orders (
2      o_oid           serial primary key,
3      o_uid           integer,
4      o_train_id      integer         not null,
5      o_date          date            not null,
6      o_start_station integer         not null,
7      o_end_station   integer         not null,
8      o_seat_type     seat_type       not null,
```

```
9      o_seat_id        integer      not null,
10     o_status         order_status not null,
11     o_effect_time    timestamp    not null,
12     foreign key (o_uid) references users (u_uid),
13     foreign key (o_train_id) references train (t_train_id),
14     foreign key (o_start_station) references station_list (s_station_id),
15     foreign key (o_end_station) references station_list (s_station_id)
16  );
```

1.1.3 train

候选键包括 train_id 与 train_name，二者均可确定 train_type，没有非平凡的依赖关系，故 train 满足 BCNF。

```
1  create table if not exists train (
2      t_train_id   serial primary key,
3      t_train_type varchar(1)  not null,
4      t_train_name varchar(10) not null
5  );
```

1.1.4 city

主键为 city_id，其可唯一确定 city_name。此处引入的 reach_table，用于记录城市间在本实验需求下的可达关系，可通过 train、train_full_info、station_list 等信息推导得到，在存储上构成了一定冗余，但用于实现需求 5 时有助于查找两地间车次的时间开销，后文讨论需求时会详细讨论。除 reach_table 外该表满足 BCNF。

```
1  create table if not exists city (
2      c_city_id     integer primary key,
3      c_city_name   varchar(20) not null,
4      c_reach_table boolean[]
5  );
```

1.1.5 station_list

候选键包括 station_id, station_name，二者均可确定 city_id，没有非平凡的依赖关系，故 station_list 满足 BCNF。

```
1  create table if not exists station_list (
2      s_station_id     serial primary key,
3      s_station_name   varchar(20) not null,
4      s_station_city_id integer     not null,
5      foreign key (s_station_city_id) references city (c_city_id)
6  );
```

1.1.6 train_full_info

该表记录每次列车每经停站的信息,主键为 (train_id, station_id),每个条目中其他信息可由该二元组确定。由于存在跨天运行的列车,故而无法仅通过 train_id 与 arrive_time 或 leave_time 确定所有信息。故 train_full_info 满足 BCNF。

在获取数据时,每车次列车初始域 day_from_departure 置 0,列车运行时间每次跨越午夜零点则域 day_from_departure 自增 1,由此记录列车准确的运行历时与进出站时间。

```
1  create table if not exists train_full_info (
2      tfi_train_id          integer,
3      tfi_station_id        integer,
4      tfi_station_order     integer          not null,
5      tfi_arrive_time       time             not null,
6      tfi_leave_time        time             not null,
7      tfi_day_from_departure integer          not null,
8      tfi_distance          integer          not null,
9      tfi_price             decimal(5, 1)[7] not null default array [0.0,
       ↪  0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
10     primary key (tfi_train_id, tfi_station_id),
11     foreign key (tfi_train_id) references train (t_train_id),
12     foreign key (tfi_station_id) references station_list (s_station_id)
13 );
```

1.1.7 station_tickets

该表记录某站某日经停的某车各类型的余票,主键为 (train_id, station_id, date),没有非平凡的依赖关系,station_tickets 满足 BCNF。

```
1  create table if not exists station_tickets (
2      stt_station_id integer,
3      stt_train_id   integer,
4      stt_date       date        not null,
```

```
5    stt_num             integer[7] not null default array [5, 5, 5, 5, 5, 5,
  ↪    5],
6    primary key (stt_station_id, stt_train_id),
7    foreign key (stt_station_id) references station_list (s_station_id),
8    foreign key (stt_train_id) references train (t_train_id),
9    foreign key (stt_station_id, stt_train_id) references train_full_info
  ↪    (tfi_station_id, tfi_train_id)
10   );
```

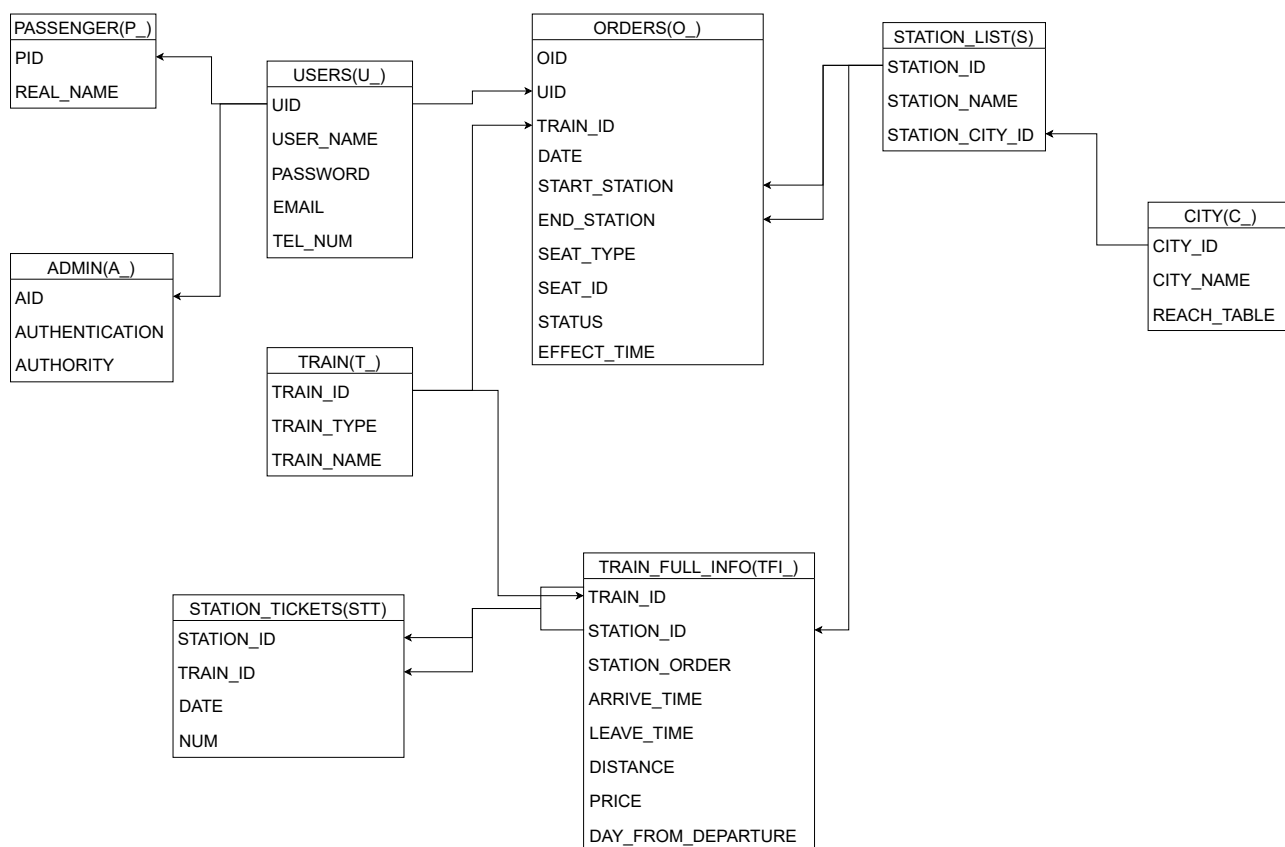## 1.2 关系模式

参考 TPCH 文档可画出如下关系模式图：

图 2: 火车订票系统关系模式图

具体的关系模式可见上一小节中的 `create table` 语句，其中涉及到的自定义枚举类型补充如下：

```
1  create type seat_type as enum ('YZ', 'RZ', 'YW_S', 'YW_Z', 'YW_X',
  ↪    'RW_S', 'RW_X');
2  create type order_status as enum ('COMPLETE', 'PRE_ORDERED', 'CANCELED');
3  create type admin_authority as enum ('ALL');
```

2　需求实现

2.1　需求 1 ∼ 3：记录相关信息

需求 1 要求记录每车次列车相关信息，可通过 1.1.3、1.1.5、1.1.6 中讨论的表记录。

需求 2 要求记录列车座位情况，可通过 1.1.7 中讨论的表与上述列车信息记录。

需求 3 中对乘客信息的记录部分可通过 1.1.1 中 users 与 passengers 表记录。注册与登录操作对应 sql 语句如下，其中对密码的加密工作交由前端 php 完成。

```
1  /* user registration */
2  /* @param: user_name */
3  /*        : user_password */
4  /*        : phone_num */
5  /*        : user_email */
6  /* @return: uid */
7  begin
8      select * into uid from insert_all_info_into__u__(user_name,
    ↪ user_password, phone_num, user_email);
9  end;
10
11 /* user login */
12 /* @param: user_name */
13 /*        : user_password */
14 /* @return: uid */
15 /*        : error */
16 begin
17     select * from query_uid_from_uname_password__u__(user_name,
    ↪ user_password) into uid, error;
18 end;
```

上述代码中使用到的两个函数的 sql 实现如下：

```
1  /* query_uid_from_uname_password__u__ */
2  /* @param: user_name */
3  /*        : user_password */
4  /* @return: uid */
5  /*        : error_type */
6  begin
7      if (select * from users where u_user_name = user_name) is null then
8          uid := 0;
9          error := 'ERROR_NOT_FOUND_UNAME';
```

```
10    else
11        select u_uid into uid from users where u_user_name = user_name
          ↪  and u_password = user_password;
12        if uid is null then
13            uid := 0;
14            error := 'ERROR_NOT_CORRECT_PASSWORD';
15        else
16            error := 'NO_ERROR';
17        end if;
18    end if;
19  end;
20
21  /* insert_all_info_into__u__ */
22  /* @param: user_name */
23  /*       : user_password */
24  /*       : phone_num */
25  /*       : user_email */
26  /* @return: uid */
27  /*       : err */
28  begin
29    if (select * from users where u_user_name = user_name) is not null
          ↪  then
30        uid := 0;
31        err := 'ERROR_DUPLICATE_UNAME';
32    else
33        if (select * from users where u_tel_num = phone_num) is not null
              ↪  then
34            uid := 0;
35            err := 'ERROR_DUPLICATE_U_TEL_NUM';
36        else
37            insert into users (u_user_name, u_password, u_email,
                  ↪  u_tel_num)
38                values (user_name, user_password, user_email, phone_num);
39            select currval(pg_get_serial_sequence('users', 'u_uid')) into
                  ↪  uid;
40            err := 'NO_ERROR';
41        end if;
42    end if;
43  end;
```

2.2 需求 4：查询具体车次

需求 1 中已根据车次记录列车信息，根据表 train、station_list、train_full_info 查询信息并组织正确的输出形式即可。

```
1  /*@param: train_name  */
2  /*       : q_date */
3  begin
4      select query_train_id_from_name__t__(train_name) into train_id;
5      select query_start_time_from_id__tfi__(train_id) into start_time;
6      return query select tfi_station_order as station_order,
7                          s_station_name as station,
8                          s_station_id as station_id,
9                          c_city_name as city_name,
10                         c_city_id as city_id,
11                         tfi_arrive_time as arrive_time,
12                         tfi_leave_time as leave_time,
13                         (select * from
                        ↪  get_actual_interval_bt_time(tfi_arrive_time,
                        ↪  tfi_leave_time, 0)) as stay_time,
14                         (select * from
                        ↪  get_actual_interval_bt_time(start_time,
                        ↪  tfi_arrive_time, (select
                        ↪  query_day_from_departure_from_id__tfi__(train_id,
                        ↪  tfi_station_id)))) as durance,
15                         tfi_distance as distance,
16                         tfi_price as seat_price,
17                         stt_num as seat_num
18                    from train_full_info tfi
19                         left join station_list s on
                        ↪  tfi.tfi_station_id = s.s_station_id
20                         left join city c on s.s_station_city_id =
                        ↪  c.c_city_id
21                         left join train t on tfi.tfi_train_id =
                        ↪  t.t_train_id
22                         left join station_tickets stt on
                        ↪  tfi.tfi_station_id = stt.stt_station_id
23                   where t_train_id = train_id
24                     and stt.stt_date = q_date;
25  end;
26
```

```
27  /* get_actual_interval_bt_time */
28  /* @param: start_time */
29  /*       : end_time */
30  /*       : days_added */
31  /* @return: actual_interval */
32  begin
33      if days_added = 0 and start_time > end_time then
34          actual_interval := interval '24 hours' + end_time - start_time;
35      else
36  ^^I    actual_interval := (days_added || 'days')::interval + end_time -
    ↪   start_time;
37      end if;
38  end
```

此处使用的函数 `query_train_id_from_name__t__` 等均为简单的匹配查找，此处与后文均不详述此类函数。

2.3　需求 5、6：查询两地之间的车次与返程信息

需求 6 相对需求 5 仅交换起始城市，可交由前端完成，sql 逻辑复用需求 5 即可。

直达列车的查询逻辑较为简单，对起始站点查询此处离开的列车，遍历这些可能的列车，通过输入的出发时间过滤掉发车时间早于出发时间的列车，通过车次与城市即可确定可能的终点站，而后则是余票与价格的计算。余票与车次座位的设计维护在需求 7 中说明。

```
1   /* @func: get_train_bt_cities_directly */
2   /* @param: from_city_id */
3   /*       : to_city_id */
4   /*       : q_date */
5   /*       : q_time */
6   begin
7       select check_reach_table(from_city_id, to_city_id) into
    ↪   city_reachable;
8       if city_reachable then
9           train_id_list := array(
10              select from_city_train.ct_train_id
11                  from city_train from_city_train
12                      join city_train to_city_train on
                          ↪   from_city_train.ct_train_id =
                          ↪   to_city_train.ct_train_id
13                  where (select get_ct_priority(from_city_id,
                      ↪   from_city_train.ct_train_id)) <
```

```
14                          (select get_ct_priority(to_city_id,
                            ↪  to_city_train.ct_train_id))
15                  );
16          <<scan_train_list>>
17          foreach train_idi in array train_id_list
18              loop
19              -- 2 ways of accomplishment --
20              -- leave station --
21                  select get_station_id_from_cid_tid(from_city_id,
                        ↪  train_idi) into station_leave_id;
22                  select query_station_name_from_id__s__(station_leave_id)
                        ↪  into station_leave_name;
23                  select q_all_info_leave.leave_time,
24                          q_all_info_leave.day_from_departure,
25                          q_all_info_leave.distance,
26                          q_all_info_leave.price
27                      into station_leave_time, station_leave_day,
                            ↪  station_leave_distance, station_arrive_price
28                      from
                            ↪  query_train_all_info_from_tid_sid__tfi__(train_idi,
                            ↪  station_leave_id) q_all_info_leave;
29                  -- check time --
30                  if station_leave_time < q_time then
31                      continue scan_train_list;
32                  end if;
33                  select query_train_name_from_id__t__(train_idi) into
                        ↪  train_namei;
34                  -- arrive station --
35                  select get_station_id_from_cid_tid(to_city_id, train_idi)
                        ↪  into station_arrive_id;
36                  select query_station_name_from_id__s__(station_arrive_id)
                        ↪  into station_arrive_name;
37                  select q_all_info_arrive.leave_time,
38                          q_all_info_arrive.day_from_departure,
39                          q_all_info_arrive.distance,
40                          q_all_info_arrive.price
41                      into station_arrive_time, station_arrive_day,
                            ↪  station_arrive_distance, station_leave_price
```

```sql
42              from
    ↪   query_train_all_info_from_tid_sid__tfi__(train_idi,
    ↪   station_arrive_id) q_all_info_arrive;
43          -- seats and price calculation --
44          for seat_i in 1..7
45              loop
46                  select array_set(station_arrive_price,
    ↪   station_arrive_price[seat_i],
47                                  station_arrive_price[seat_i] -
    ↪   station_leave_price[seat_i])
48                      into res_price;
49              end loop;
50          select get_min_seat.seat_num
51              into seat_nums
52              from get_min_seats(train_idi, q_date,
    ↪   station_leave_id, station_arrive_id,
53                              array ['YZ', 'RZ', 'YW_S', 'YW_Z',
    ↪   'YW_X', 'RW_S', 'RW_X'])
    ↪   get_min_seat;
54          -- return row --
55          for r in
56              select train_namei as train_name,
57                  train_idi as train_id,
58                  station_leave_name as station_from_name,
59                  station_leave_id as station_from_id,
60                  station_arrive_name as station_to_name,
61                  station_arrive_id as station_to_id,
62                  station_leave_time as leave_time,
63                  station_arrive_time as arrive_time,
64                  (station_arrive_day - station_leave_day ||
    ↪   'days')::interval + station_arrive_time -
65                  station_leave_time as durance,
66                  station_arrive_distance -
    ↪   station_leave_distance as distance,
67                  res_price as seat_price,
68                  seat_nums as seat_nums,
69                  false as transfer_first,
70                  false as transfer_late
71              loop
72                  return next r;
```

```
73                    end loop;
74                end loop;
75        end if;
76        return;
77    end;
```

对于换乘一次的情况，对于出发城市，查找其可直达的所有城市，遍历这些城市以它们为换乘城市，查找到目的城市的可能列车（即查询从换乘城市到目的城市的直达列车），事先通过 city 表中 reach_table 域记录城市间的可达与否可以在遍历换乘城市时避免与目的城市完全不可达的城市作为换乘城市时的查询操作（即上文函数`get_train_bt_cities_directly`开头的`city_reachable`检查），减小开销。

具体实现上类似 BFS，通过队列记录可能的换乘城市，遍历所有可能的情况直到清空队列。

以下代码指定`query_transfer`为真时查询换乘一次的结果，为否则调用上文函数查询直达列车。

```
1    /* @param: city_from */
2    /*       : city_to */
3    /*       : q_date */
4    /*       : q_time */
5    /*       : query_transfer */
6    begin
7        select query_city_id_from_name__c__(city_from) into from_city_id;
8        select query_city_id_from_name__c__(city_to) into to_city_id;
9        select check_reach_table(from_city_id, to_city_id) into
         ↪  city_reachable;
10       if city_reachable then
11           if not query_transfer then
12               for r in
13                   select * from get_train_bt_cities_directly(from_city_id,
                     ↪  to_city_id, q_date, q_time)
14                   loop
15                       return next r;
16                   end loop;
17           else
18               -- first set of transfer trains must be ones passing from
                 ↪  city --
19               -- so outside loop --
20               passing_trains := array(select
                 ↪  query_train_id_list_from_cid__ct__(from_city_id));
```

12

```
21          while (select array_length(src_city, 1)) > 0 and (select
      ↪   array_position(src_city, to_city_id)) is not null
22              loop
23                  select array_length(src_city, 1) into
                  ↪   current_level_city_num;
24                  for city_i in 1..current_level_city_num
25                      loop
26                          neighbour_city := array(select
                          ↪   get_ct_next_city_list(src_city[1],
                          ↪   passing_trains));
27                          src_city := array(select array_cat(src_city,
                          ↪   neighbour_city));
28                          -- initially from_city_id was in src_city --
29                          -- so remove it first because we have dealt
                          ↪    with it --
30                          src_city := array(select
                          ↪   array_remove_elem(src_city, 1));
31                          -- then src_city[1] is middle city to
                          ↪    transfer --
32                          if (select array_length(src_city, 1)) > 1
                          ↪    then
33                              -- ... --
34                          end if;
35                      end loop;
36              end loop;
37          return;
38      end if;
39  end if;
40 end;
```

对于每个可能的换乘城市，遍历出发城市到其的每次直达列车，查询其到目的城市的每次直达列车，根据上游列车与下游列车是否在同站换乘检查不同的时间要求是否满足。

```
1 if (select array_length(src_city, 1)) > 1 then
2     for r in
3         (select *
4             from get_train_bt_cities_directly(from_city_id,
             ↪   src_city[1], q_date, q_time))
5         loop
6             for j in
7                 (select *
```

```
8                          from get_train_bt_cities_directly(src_city[1],
                           ↪  to_city_id, q_date, q_time + r.durance +
                           ↪  interval '1 hour'))
9                  loop
10                      if r.station_to_id = j.station_from_id then
11                          select *
12                              from
                                 ↪  get_actual_interval_bt_time(r.arrive_time,
                                 ↪  j.leave_time, 0)
13                              into transfer_interval;
14                          if transfer_interval >= interval '1 hour'
15                              and transfer_interval <= interval '4 hours'
16                          then
17                              return next r;
18                              return next j;
19                          end if;
20                      else
21                          if transfer_interval >= interval '2 hours'
22                              and transfer_interval <= interval '4 hours'
23                          then
24                              return next r;
25                              return next j;
26                          end if;
27                      end if;
28                  end loop;
29          end loop;
30  end if;
```

2.4 需求 7：预订车次座位

余票与车次座位的设计上，我们在 station_tickets 表中记录某天某车在某站的各类型余票数，每当乘客购买某天某一区间车票，则将该天该车该区间内（包括左端不包括右端）所有对应类型余票数减 1。查询时对某车某天某区间的某类型余票数则为该区间内每站该类型余票数的最小值。获取座位时先查找区间内合法的座位，其后更新 station_tickets 表内容。

```
1  /* try_occupy_seats */
2  /* @param: train_id */
3  /*       : order_date */
4  /*       : station_from_id */
5  /*       : station_to_id */
```

```
6   /*          : seat_type */
7   /*          : seat_num */
8   /* @return: succeed */
9   /*          : left_seat */
10  begin
11      select query_station_order_from_tid_sid__tfi__(train_id,
        ↪  station_from_id) into station_start_order;
12      station_order_ptr = station_start_order;
13      select query_station_order_from_tid_sid__tfi__(train_id,
        ↪  station_to_id) into station_end_order;
14      -- find min_seat --
15      select get_min_seat.seat_num
16          into min_seat
17          from get_min_seats(train_id, order_date, station_from_id,
            ↪  station_to_id, array [seat_type]) get_min_seat
18          where in_order = 1;
19      -- check satisfiability --
20      if min_seat < seat_num then
21          succeed := false;
22          left_seat := min_seat;
23      else
24          succeed := true;
25          left_seat := 5 - min_seat;
26          -- second loop, update station tickets --
27          while station_order_ptr != station_end_order
28              loop
29                  update station_tickets
30                  set stt_num = (select array_set(stt_num, seat_type,
                    ↪  stt_num[seat_type] - seat_num))
31                      where stt_train_id = train_id
32                        and stt_station_id = station_id_ptr
33                        and stt_date = order_date;
34                  station_order_ptr := station_order_ptr + 1;
35                  select query_station_id_from_tid_so__tfi__(train_id,
                    ↪  station_order_ptr) into station_id_ptr;
36              end loop;
37      end if;
38  end;
```

用户预定座位时，在确定区间车次日期等信息申请订单后即暂时拥有座位（如有），此时记订单状态为 PRE_ORDERED，点击确认后将订单置为 ORDERED。有关订单状态的维护将在需

求 8 内讨论。

```
1   /* pre_order_train */
2   /* @param: train_id */
3   /*       : station_from_id */
4   /*       : station_to_id */
5   /*       : seat_type */
6   /*       : seat_num */
7   /*       : order_date */
8   /* @return: succeed */
9   /*        : seat_id */
10  /*        : order_id */
11  begin
12      select succeed, left_seat
13          into succeed, seat_id
14          from try_occupy_seats(train_id, order_date, station_from_id,
              ↪  station_to_id, seat_type, seat_num);
15      if succeed then
16          insert into orders (o_train_id, o_date, o_start_station,
              ↪  o_end_station, o_seat_type, o_seat_id,
17                              o_status, o_effect_time)
18          select train_id,
19                 order_date,
20                 station_from_id,
21                 station_to_id,
22                 seat_type,
23                 seat_id,
24                 'PRE_ORDERED',
25                 now();
26          select currval(pg_get_serial_sequence('orders', 'o_oid')) into
              ↪  order_id;
27      else
28          order_id := 0;
29      end if;
30  end;
31
32  /* order_train_seats */
33  /* @param: order_id */
34  /*       : uid */
35  /* @return: succeed */
36  begin
```

```
37   update orders
38   set (o_uid, o_status) = (uid, 'ORDERED')
39       where o_oid = order_id;
40   end;
```

2.5 需求 8：查询订单和删除订单

订单相关信息记录在 orders 表内，给定用户、出发日期范围等信息后即可查询获得。

```
1    select o_oid as order_id,
2        t_train_name as train_name,
3        o_train_id as train_id,
4        s_start.s_station_name as station_leave,
5        o_start_station as station_id,
6        s_arrive.s_station_name as station_arrive,
7        tfi_start.tfi_leave_time as start_time,
8        tfi_end.tfi_arrive_time as arrive_time,
9        (select * from get_actual_interval_bt_time(tfi_start.tfi_leave_time,
     ↪   tfi_end.tfi_arrive_time,
10           tfi_start.tfi_day_from_departure -
             ↪   tfi_end.tfi_day_from_departure)) as durance,
11       tfi_end.tfi_distance - tfi_start.tfi_distance as distance,
12       o_seat_type as seat_type,
13       o_seat_id as seat_id,
14       o_status as status,
15       tfi_end.tfi_price[o_seat_type] - tfi_start.tfi_price[o_seat_type] + 5
     ↪   as price
16       from orders
17               left join station_list s_start on orders.o_start_station =
                 ↪   s_start.s_station_id
18               left join station_list s_arrive on orders.o_end_station =
                 ↪   s_arrive.s_station_id
19               left join train on orders.o_train_id = train.t_train_id
20               left join train_full_info tfi_start on s_start.s_station_id =
                 ↪   tfi_start.tfi_station_id
21           and orders.o_train_id = tfi_start.tfi_train_id
22               left join train_full_info tfi_end on s_start.s_station_id =
                 ↪   tfi_end.tfi_station_id
23           and orders.o_train_id = tfi_end.tfi_train_id
24       where o_uid = uid
```

```
25      and o_date >= start_query_date
26      and o_date <= end_query_date;
```

我们将订单的状态分为三种：取消、预约、确认。用户发起申请后确认订单前订单为预约态，此时对订单的操作仅有点击确认完成订单进入确认态，30 分钟内未确认的订单将会自动老化删除。确认态的订单可被手动取消。确认操作的 sql 语句在需求 7 中已经给出，以下给出取消与老化操作的 sql 语句：

```
1   /* user_cancel sql */
2   /* @param: order_id */
3   /* @return: succeed */
4   begin
5       select o_train_id, o_date, o_start_station, o_end_station,
        ↪  o_seat_type
6           into train_id, order_date, start_station, end_station, seat_type
7           from orders
8           where o_oid = order_id
9             and o_status = 'COMPLETE';
10      select release_seats(train_id, order_date, start_station,
        ↪  end_station, seat_type, 1);
11      update orders
12      set o_status = 'CANCELED'
13          where o_oid = order_id;
14  end;
15
16  /* remove_outdated_order */
17  delete
18      from orders
19      where (select * from
        ↪  get_actual_interval_bt_time(orders.o_effect_time, now(), 0))
20              > interval '30 minutes'
21          and orders.o_status = 'PRE_ORDERED';
```

2.6 需求 9：管理员

管理员相关需求中，查看每个用户订单一项可复用需求 8 中查看订单的实现。查询当前注册用户列表的实现也较简单，对 passengers 表[①]查询即可。

```
1   select p_pid as uid,
2       u_user_name as uname,
3       array(
```

[①]我们将所有用户 users 分为 passengers 和 admin 两类，约定管理员的查询功能为查询所有 passengers

```
4            select o_oid from orders where o_uid = p_pid
5        ) as orders
6    from passengers
7        left join users u on passengers.p_pid = u.u_uid;
```

对总订单数、总票价、最热点车次排序的查询涉及到订单的取消态，由于我们额外引入了预约态，此处约定这三处查询仅包括已确认的订单，不统计取消订单与已申请未确认的订单。

```
1  -- 2 views to select top 10 train --
2  create or replace view top_10_train_tickets(train_id, count_num)
3  as
4  select o_train_id as train_id, count(*) as count_num
5      from orders
6      where o_status = 'COMPLETE'
7      group by o_train_id
8      order by count_num
9      limit 10;
10
11 create or replace view top_10_train_ids(train_id)
12 as
13 select train_id
14     from top_10_train_tickets;
15
16
17 /* admin_query_orders */
18 /* @return: total_order_num */
19 /*        : total_price */
20 /*        : hot_trains */
21 begin
22    select count(*),
23           sum(tfi_end.tfi_price[o_seat_type] -
           ↪    tfi_start.tfi_price[o_seat_type] + 5)
24        into total_order_num, total_price
25        from orders
26             left join train_full_info tfi_start on o_start_station =
                 ↪    tfi_start.tfi_station_id
27        and orders.o_train_id = tfi_start.tfi_train_id
28             left join train_full_info tfi_end on o_end_station =
                 ↪    tfi_end.tfi_station_id
29        and orders.o_train_id = tfi_end.tfi_train_id
```

19

```
30              and orders.o_status = 'COMPLETE';
31      hot_trains := array(select t_train_name
32                              from train
33                                  left join top_10_train_ids on
                                    ↪   train.t_train_id =
                                    ↪   top_10_train_ids.train_id);
34  end;
```

成员分工

高梓源：

肖展琪：

桂庭辉：