Домашнее задание №9
Тельнов Сергей

Используемая база данных

```
create table planes (
  id int PRIMARY KEY
);
create table seats (
  seat_no int PRIMARY KEY,
  plane_id int REFERENCES planes (id)
);
create table flights (
  id int PRIMARY KEY,
  flight_time timestamp not null,
  plane_id int REFERENCES planes (id)
);
create table users (
  id int PRIMARY KEY,
  password varchar(20)
);
create table booking (
  user_id int REFERENCES users (id),
  booking_status boolean not null, -- isBought flag
  seat_no int REFERENCES seats (seat_no),
  flight_id int REFERENCES flights (id),
  reserved_time timestamp,
  CONSTRAINT tickets_unique UNIQUE (seat_no, flight_id)
);
```

Общая функция для проверки пароля

```
create function check_authorization(userId int, pass varchar)
  returns boolean as
  $$
    begin
      return exists (
        select *
        from users
        where id = userId and password = pass
      );
    end;
  $$ language plpgsql;
```

1. `FreeSeats(FlightId)` — список мест, доступных для продажи и бронирования

```
create function free_seats(flightId int)
  returns table (
    seat_no int,
    can_reserve boolean,
    can_buy boolean
  ) as
  $$
    begin
      return query (
        select seats.seat_no,
               (case
        when (now() + interval '1 day') < flights.flight_time then true
        else false
        end),
        (case
         when (now() + interval '2 hour') < flights.flight_time then true
         else false
         end)
    from seats
    join flights
      on flights.plane_id = seats.plane_id
    left join booking
      on seats.seat_no = booking.seat_no
        and booking.flight_id = flights.id
    where flights.id = flightId
      and booking.seat_no is null
   );
 end;
 $$ language plpgsql;
```

2. `Reserve(UserId, Pass, FlightId, SeatNo)` — пытается забронировать место. Возвращает *истину*, если удалось и *ложь* — в противном случае.

```
create function reserve(userId int, pass varchar, flightId int, seatNo int)
 returns boolean as
 $$
  declare flightTime timestamp;
  begin
   if not check_authorization(userId, pass) then
    return false;
   end if;

   flightTime := (select flight_time from flights where flights.id = flightId);

   if flightTime is null or flightTime > (now() - interval '1 day') then
    return false;
   end if;

   insert into booking
   (user_id, booking_status, seat_no, flight_id, reserved_time)
   values
   (userId, false, seatNo, flightId, now())
   on conflict do nothing;

   return found;
  end;
 $$ language plpgsql;
```

3. `ExtendReservation(UserId, Pass, FlightId, SeatNo)` — пытается продлить бронь места. Возвращает *истину*, если удалось и *ложь* — в противном случае.

```
create function extend_reservation(userId int, pass varchar, flightId int, seatNo int)
 returns boolean as
 $$
  begin
   if not check_authorization(userId, pass) then
```

```
    return false;
   end if;

   update booking
   set reserved_time = now()
   where user_id = userId
     and flight_id = flightId
     and seat_no = seatNo
     and booking_status = false;

   return found;
  end;
 $$ language plpgsql;
```

4. `BuyFree(FlightId, SeatNo)` — пытается купить свободное место. Возвращает *истину*, если удалось и *ложь* — в противном случае.

```
create function buy_free(flightId int, seatNo int)
 returns boolean as
 $$
  begin
   insert into booking
   (user_id, booking_status, seat_no, flight_id, reserved_time)
   values
--  покупаем билет на имя авиакомпании
   (null, true, seatNo, flightId, now())
   on conflict do nothing;

   return found;
  end;
 $$ language plpgsql;
```

5. `BuyReserved(UserId, Pass, FlightId, SeatNo)` — пытается выкупить забронированное место (пользователи должны совпадать). Возвращает *истину*, если удалось и *ложь* — в противном случае.

```
create function buy_reserved(userId int, pass varchar, flightId int, seatNo int)
 returns boolean as
 $$
  begin
   if not check_authorization(userId, pass) then
    return false;
   end if;

   update booking
   set booking_status = true
   where user_id = userId
     and flight_id = flightId
     and seat_no = seatNo
     and booking_status = false;

   return found;
  end;
 $$ language plpgsql;
```

6. `FlightStatistics(UserId, Pass)` — статистика по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест.

```
create function flight_statistics(userId int, pass varchar)
 returns table (
   flight_id int,
   can_buy boolean,
   can_reserve boolean,
   free_count int,
   reserved_count int,
   bought_count int
 ) as
 $$
   begin
     if not check_authorization(userId, pass) then
       raise exception 'authorization error';
     end if;

     return query (
       with booked as (
         select booking.flight_id,
                flights.flight_time,
                sum(case
                 when booking_status then 0
                 else 1
               end) as reserved_count,
                sum(case
                 when booking_status then 1
                 else 0
               end) as bought_count
         from booking
         join flights
           on booking.flight_id = flights.id
         group by booking.flight_id, flights.flight_time
       ),
       free as (
         select flights.id as flight_id,
                flights.flight_time,
                count(*) as free_count
         from seats
         join flights
           on flights.plane_id = seats.plane_id
         left join booking
           on seats.seat_no = booking.seat_no
             and booking.flight_id = flights.id
         where
           booking.seat_no is null
         group by flights.id
       )
       select COALESCE(free.flight_id, booked.flight_id),
              now() + interval '1 day' < COALESCE(free.flight_time, booked.flight_time),
              now() + interval '2 hour' < COALESCE(free.flight_time, booked.flight_time),
              COALESCE(free.free_count, 0)::int,
              COALESCE(booked.reserved_count, 0)::int,
              COALESCE(booked.bought_count, 0)::int
       from free
       full outer join booked
```

```
        on free.flight_id = booked.flight_id
    );
  end;
$$ language plpgsql;
```

7. `FlightStat(UserId, Pass, FlightId)` — статистика по рейсу: возможность бронирования и покупки, число свободных, забронированных и проданных мест.

```
create function flight_stat(userId int, pass varchar, flightId int)
 returns table (
  can_buy boolean,
  can_reserve boolean,
  free_count int,
  reserved_count int,
  bought_count int
 ) as
 $$
  begin
    return query (
      select t.can_buy,
             t.can_reserve,
             t.free_count,
             t.reserved_count,
             t.bought_count
      from flight_statistics(userId, pass) as t
      where flight_id = flightId
    );
  end;
 $$ language plpgsql;
```