Домашнее задание №8
Тельнов Сергей

Используемая база данных на SQL

```sql
create database Airline;

create table planes (
  id int PRIMARY KEY
);

create table seats (
  seatNo varchar(5) 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 type ticket_status as enum ('booked', 'bought', 'unavailable');

create table tickets (
  id int PRIMARY KEY,
  status ticket_status not null,
  seatNo varchar(5) REFERENCES seats (seatNo),
  flight_id int REFERENCES flights (id),
  reserved_time timestamp,
  CONSTRAINT tickets_unique UNIQUE (seatNo, flight_id)
);
```

Добавим триггеры для первого задания

```
create function limit_time_check()
  returns trigger as
  $$
    DECLARE
      curr_flight_time timestamp;
    begin
      curr_flight_time := (select flight_time from flights where flights.id =
NEW.id);
      if NEW.status = 'booked' then
        if NEW.reserved_time > (curr_flight_time - interval '1 day') then
          RAISE EXCEPTION 'cannot booking seats later than one day before
flight';
        end if;
      elsif NEW.status = 'bought' then
        if NEW.reserved_time > (curr_flight_time - interval '2 hour') then
          RAISE EXCEPTION 'cannot buy tickets later than one day before flight';
        end if;
      end if;
      return NEW;
    end;
  $$ language plpgsql;

create trigger limit_time_trigger
before update or insert on tickets
for each row execute procedure limit_time_check();
```

```
create function booking_timeout()
  returns trigger as
  $$
    DECLARE
     curr_reserved_time timestamp;
    begin
      if NEW.status = 'booked' then
        curr_reserved_time := (select reserved_time from tickets where id =
NEW.id);
        if curr_reserved_time > (now() - interval '1 day') then
          delete from tickets where id = NEW.id;
        end if;
      end if;
      return NEW;
    end;
  $$ language plpgsql;

create trigger booking_timeout_trigger
after update or insert on tickets
for each row execute procedure booking_timeout();
```

```
create function already_bought()
  returns trigger as
  $$
    begin
      if OLD.status = 'bought' then
        RAISE EXCEPTION 'this ticket had already bought';
      end if;
      return NEW;
    end;
  $$ language plpgsql;

create trigger already_bought_trigger
before update on tickets
for each row execute procedure already_bought();
create function remove_booking()
  returns trigger as
  $$
    DECLARE
      curr_reserved_time timestamp;
      curr_flight_time timestamp;
    begin
      if OLD.status = 'booked' then
        curr_reserved_time := (select reserved_time from tickets where id =
OLD.id);
        curr_flight_time := (select flight_time from flights where flights.id =
NEW.id);

        if curr_reserved_time > (curr_flight_time - interval '1 day') then
          delete from tickets where id = OLD.id;
        end if;
      end if;
      return OLD;
    end;
  $$ language plpgsql;

create trigger remove_booking_trigger
before select on tickets
for each row execute procedure remove_booking();
```

1. Одно место не может быть продано или забронировано более чем один раз (в том числе, продано и забронировано одновременно).

На tickets стоит ограничения

```
CONSTRAINT tickets_unique UNIQUE (seatNo, flight_id)
```

Из-за этого не может быть куплено несколько билетов на одно место

2. Бронь можно обновить, после чего она будет действительна ещё одни сутки.

Можно обновить tickets по id и изменить **reserved_time** на **now()**. За удаление брони будет отвечать функция **booking_timeout()**.

3. Бронь автоматически снимается через сутки после последнего обновления, но не позже, чем за сутки, до вылета рейса.

Для автоматического обновления сделал триггер (как сделать на таймаут я не понял). За удаления будут отвечать два триггера **remove_booking_trigger** и **booking_timeout_trigger**.

4. Бронирование автоматически закрывается за сутки до времени рейса.

Для этого сделал функцию **limit_time_check()**, которая не позволит сделать insert/update за какое-то время до вылета.

5. Продажи автоматически закрываются не позднее двух часов до вылета рейса, либо при распродаже всех мест либо по запросу администратора.

Билет нельзя купить за 2 часа до вылета из-за **limit_time_check()**. Если все билеты скуплены, то нельзя купить новые из-за

```
CONSTRAINT tickets_unique UNIQUE (seatNo, flight_id)
```

По запросу администратора можно поставить не скупленным билетам **status 'unavailable'**.

2. Запишите определение базы данных `Airline` на языке SQL.

Описал выше

3. Определите, какие индексы требуется добавить к таблицам базы данных `Airline`.

В таблице **flight** часто использовались поля **id** и **flight_time**, в **tickets** поля **id** и **reserved_time**.

Можно попробовать добавить индексы на пары **(flight.id, flight_time), (tickets.id, reserved_time)**.

4. Пусть частым запросом является определение средней заполненности самолёта по рейсу. Какие индексы могут помочь при исполнении данного запроса?

В таком случае будет много запросов с использование **(tickets.flight_id, tickets.status)**. Можно попробовать добавить индексы на эту пару полей.

5. Запишите добавление индексов на языке SQL.

```
create index idx_flights_time on flights (id, flight_time);
create index idx_tickets_time on tickets (id, reserved_time);
create index idx_tickets_flights on tickets (flight_id, status);
```