Королева, М3437

Создание базы данных:

```
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;


create table Planes (
        PlaneId int primary key,
        SeatsAmount int not null
);

create table Flights (
        FlightId serial primary key,
        FlightTime timestamp not null,
        PlaneId int not null,
        Closed boolean default false
);

create table Seats (
        PlaneId int not null,
        SeatNo int not null,
        Booked boolean default false not null,
        Sold boolean default false not null,
        ClientId int default null,
        ReservedTime timestamp default null,
        primary key (PlaneId, SeatNo),
        foreign key (PlaneId) references Planes(PlaneId) on delete cascade
);

create or replace function createSeats() returns trigger as $$
        declare
                i int default 1;
    begin
        while i <= new.SeatsAmount
        loop
                insert into Seats values (new.PlaneId, i) on conflict do nothing;
                i := i + 1;
        end loop;
        return new;
    end;
    $$ LANGUAGE plpgsql;

create trigger createSeats
        after insert or update on Planes
        for each row
        execute procedure createSeats();

insert into Planes (PlaneId, SeatsAmount) values
        (1, 3), (2, 3), (3, 3), (4, 3), (5, 3);


insert into Flights (FlightTime, PlaneId) values
        ('2017-12-04 10:00:00', 1),
        ('2017-12-04 23:00:00', 2),
        ('2017-12-05 11:30:00', 3),
        ('2017-12-05 14:00:00', 4),
        ('2017-12-03 09:30:00', 5);
```

Условие того, что букинг будет закрыт за 1 день до полета: передается время попытки букинга + 1 день
Условие того, что покупка будет закрыта за 2 часа до полета: передается время попытки покупки + 2 часа

-- Условие того, что операция произойдет в правильные временные рамки

```
create or replace function CheckOperation(FlightTime_ timestamp, Closed_ boolean, PlaneId_ int,
Threshold timestamp, Exception_ varchar)
        returns boolean as $$
   begin
       if Closed_ then
               raise exception 'Букинг и покупка мест на полет закрыты';
       end if;

       if Threshold > FlightTime_
       then
               raise exception '%', Exception_;
       end if;
       return true;
   end;
   $$ LANGUAGE plpgsql;
```

-- Удаляет все брони на момент, когда до полета осталось меньше дня, или брони больше суток, или букинг закрыт

```
create or replace function Unbooking(FlightId_ int, FlightTime_ timestamp, Closed_ boolean)
returns
        void as $$
        declare
                curr timestamp default now();
                PlaneId_ int;
        begin
                select PlaneId into PlaneId_ from Seats where PlaneId in (select PlaneId from
Flights where FlightId = FlightId_) for update;
                if FlightTime_ < curr + interval '1 day' or Closed_ then
                        update Seats set Booked = false, ClientId = null, ReservedTime = null where
PlaneId = PlaneId_;
                end if;

                update Seats set Booked = (select case when Seats.ReservedTime < curr then
false else Seats.Booked end),
                ClientId = (select case when Seats.ReservedTime < curr then null else
Seats.ClientId end),
                ReservedTime = (select case when Seats.ReservedTime < curr then null else
Seats.ReservedTime end) where PlaneId = PlaneId_;

        end;
        $$ LANGUAGE plpgsql;
```

-- Удаляет бронь на место, если до полета осталось меньше дня или брони больше суток, или букинг закрыт

```
create or replace function UnbookPlace(PlaneId_ int, SeatNo_ int, FlightTime_ timestamp,
Closed_ boolean) returns
        void as $$
        declare
                curr timestamp default now();
    begin
        if FlightTime_ < curr + interval '1 day' or Closed_ then
                    update Seats set Booked = false, ClientId = null, ReservedTime = null where
PlaneId = PlaneId_;
                end if;
                if (select ReservedTime from Seats where PlaneId = PlaneId_ and SeatNo =
SeatNo_) < curr then
                    update Seats set Booked = false, ClientId = null, ReservedTime = null where
PlaneId = PlaneId_ and SeatNo = SeatNo_;
                end if;
    end;
    $$ LANGUAGE plpgsql;
```

-- 1. По номеру рейса — список мест, доступных для продажи и бронирования.

```
create or replace function GetSeatsByFlightId(FlightId_ int) returns
        table(gsbfi_PlaneId int, gsbfi_SeatNo int) as $$
        declare
                FlightTime_ timestamp;
                Closed_ boolean;
        begin
                select FlightTime, Closed into FlightTime_, Closed_ from Flights where FlightId =
FlightId_ for share;
                if not exists (select * from Flights where FlightId = FlightId_)
                then
                raise exception 'Рейс с таким номером не найден';
        end if;
        perform Unbooking(FlightId_, FlightTime_, Closed_);
                return query
                select PlaneId, SeatNo from
                        Seats where PlaneId in (select PlaneId from Flights where FlightId =
FlightId_ and Closed = false) and
                                        Booked = false and
                                        Sold = false;
        end;
        $$ LANGUAGE plpgsql;

begin;
select * from GetSeatsByFlightId(3);
commit;
```

-- 2. Бронирование места.

```
create or replace function BookSeat(PlaneId_ int, SeatNo_ int, ClientId_ int) returns
        boolean as $$
        declare
                FlightTime_ timestamp;
                Closed_ boolean;
                curr timestamp default now();
        begin
                select FlightTime, Closed into FlightTime_, Closed_ from Flights where PlaneId =
PlaneId_ for share;
                perform * from Seats where PlaneId = PlaneId_ and SeatNo = SeatNo_ for update;

                perform UnbookPlace(PlaneId_, SeatNo_, FlightTime_, Closed_);
                perform * from CheckOperation(FlightTime_, Closed_, PlaneId_, curr + interval '1
day', 'Букинг на полет закрыт');
                update Seats set Booked = True, ClientId = ClientId_, ReservedTime = curr +
interval '1 day' where
                        PlaneId = PlaneId_ and
                        SeatNo = SeatNo_ and
                        Booked = False and
                        Sold = False;
                return found;
        end;
        $$ LANGUAGE plpgsql;

begin;
select * from BookSeat(3, 1, 1);
commit;
```

-- 3. Продление брони.

```
create or replace function UpdateBooking(PlaneId_ int, SeatNo_ int, ClientId_ int) returns
        boolean as $$
        declare
                FlightTime_ timestamp;
                Closed_ boolean;
                curr timestamp default now();
        begin
                select FlightTime, Closed into FlightTime_, Closed_ from Flights where PlaneId =
PlaneId_ for share;
                perform * from Seats where PlaneId = PlaneId_ and SeatNo = SeatNo_ for update;

                perform UnbookPlace(PlaneId_, SeatNo_, FlightTime_, Closed_);
                perform * from CheckOperation(FlightTime_, Closed_, PlaneId_, curr + interval '1
day', 'Букинг на полет закрыт');
                update Seats set ReservedTime = curr + interval '1 day' where
                        PlaneId = PlaneId_ and
                        SeatNo = SeatNo_ and
                        Booked = True and
                        ClientId = ClientId_;
                return found;
        end;
        $$ LANGUAGE plpgsql;

begin;
select * from UpdateBooking(3, 1, 1);
commit;
```

-- 4. Покупка места.

```
create or replace function BuySeat(PlaneId_ int, SeatNo_ int, ClientId_ int) returns
    boolean as $$
    declare
        FlightTime_ timestamp;
        Closed_ boolean;
        curr timestamp default now();
    begin
        select FlightTime, Closed into FlightTime_, Closed_ from Flights where PlaneId =
PlaneId_ for share;
        perform * from Seats where PlaneId = PlaneId_ and SeatNo = SeatNo_ for update;

        perform UnbookPlace(PlaneId_, SeatNo_, FlightTime_, Closed_);
        perform * from CheckOperation(FlightTime_, Closed_, PlaneId_, curr + interval '2
hours', 'Покупка мест на полет закрыта');
        update Seats set Sold = True, ClientId = ClientId_, ReservedTime = curr where
            PlaneId = PlaneId_ and
            SeatNo = SeatNo_ and
            Booked = False and
            Sold = false;
        return found;
    end;
    $$ LANGUAGE plpgsql;


begin;
select * from BuySeat(3, 1, 1);
commit;
```

-- 5. Покупка места по брони.

```
create or replace function BuyBookedSeat(PlaneId_ int, SeatNo_ int, ClientId_ int) returns
    boolean as $$
    declare
        FlightTime_ timestamp;
        Closed_ boolean;
        curr timestamp default now();
    begin
        select FlightTime, Closed into FlightTime_, Closed_ from Flights where PlaneId =
PlaneId_ for share;
        perform * from Seats where PlaneId = PlaneId_ and SeatNo = SeatNo_ for update;

        perform UnbookPlace(PlaneId_, SeatNo_, FlightTime_, Closed_);
        perform * from CheckOperation(FlightTime_, Closed_, PlaneId_, curr + interval '2
hours', 'Покупка мест на полет закрыта');
        update Seats set Sold = True, Booked = False, ReservedTime = curr where
            PlaneId = PlaneId_ and
            SeatNo = SeatNo_ and
            Booked = True and
            ClientId = ClientId_;
        return found;
    end;
    $$ LANGUAGE plpgsql;


begin;
select * from BuyBookedSeat(3, 1, 1);
commit;
```

-- 6. Закрытие продаж на рейс по запросу администратора.

```
create or replace function CloseFlight(FlightId_ int) returns
        boolean as $$
        begin
                perform FlightTime, Closed from Flights where FlightId = FlightId_ for update;

                update Flights set Closed = True where
                        FlightId = FlightId_;
                return found;
        end;
        $$ LANGUAGE plpgsql;


begin;
select * from CloseFlight(2);
commit;
```

-- 7. Статистика по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест.

(Под «Числом свободных» мест считаются те места, которые никто не купил и не забронировал)

```
create or replace function CheckAvailable(FlightId_ int) returns
        boolean as $$
        declare
                PlaneId_ int;
                Closed_ boolean;
        begin
                select PlaneId, Closed into PlaneId_, Closed_ from Flights where FlightId =
FlightId_;
                if Closed_ or (select count(*) from Seats where PlaneId = PlaneId_ and Sold) =
(select SeatsAmount from Planes where PlaneId = PlaneId_)
                then
                        return true;
                end if;
                return false;
        end;
        $$ LANGUAGE plpgsql;

create or replace function ShowStatistics() returns
        table(stat_flightid int, stat_isclosed boolean, stat_available bigint, stat_booked bigint,
stat_sold bigint) as $$
        begin
                perform * from Flights for share;
                perform * from Seats for share;

                perform Unbooking(t2.FlightId, t2.FlightTime, t2.Closed) from (select FlightId,
FlightTime, Closed from flights) as t2;

                return query
                (select t1.FlightId, CheckAvailable(t1.FlightId),
                        (select count(*) as countAvailable from Seats
                                where PlaneId = t1.PlaneId and not Booked and not Sold),
                        (select count(*) as countBooked from Seats
                                where PlaneId = t1.PlaneId and Booked),
                        (select count(*) as countSold from Seats
                                where PlaneId = t1.PlaneId and Sold)
                        from Flights as t1);
        end;
        $$ LANGUAGE plpgsql;
begin; \n select * from ShowStatistics(); \n commit;
```