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;

1. 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;

1. 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;

1. 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;

1. 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;

1. 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;

1. 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;