Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Волжева М. И.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы 3
Практическое задание 3
Вариант 19. БД «Пассажир» 3
Выполнение 3
Вывод 8

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

Вариант 19. БД «Пассажир»

Описание предметной области:

Информационная система служит для продажи железнодорожных билетов. Билеты могут продаваться на текущие сутки или предварительно (не более чем за 45 суток). Цена билета при предварительной продаже снижается на 5%. Билет может быть приобретен в кассе или онлайн. Если билет приобретен в кассе, необходимо знать, в какой. Для каждой кассы известны номер и адрес. Кассы могут располагаться в различных населенных пунктах.

Поезда курсируют по расписанию, но могут назначаться дополнительные поезда на заданный период или определенные даты.

По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки.

Необходимо учитывать, что местом посадки и высадки пассажира могут быть промежуточные пункты по маршруту.

БД должна содержать следующий минимальный набор сведений: Номер поезда. Название поезда. Тип поезда. Пункт назначения. Пункт назначения для проданного билета. Номер вагона. Тип вагона. Количество мест в вагоне. Цена билета. Дата отправления. Дата прибытия. Дата прибытия для пункта назначения проданного билета. Время отправления. Номер вагона в поезде. Номер билета. Место. Тип места. Фамилия пассажира. Имя пассажира. Отчество пассажира. Паспортные данные.

Задание 4. Создать хранимые процедуры:

1. Для повышения цен в пригородные поезда на 20%.
2. Для создания нового рейса на поезд.
3. Для формирования общей выручки по продаже билетов за сутки.

Задание 5. Создать необходимые триггеры

Выполнение

Создание хранимых процедур:

1. Для повышения цен в пригородные поезда на 20%.

	seat_number	price	train_type
1	18	907,20 ?	suburban
2	1	3NBSP000,00 ?	suburban
3	2	3NBSP000,00 ?	suburban
4	3	3NBSP000,00 ?	suburban
5	4	3NBSP000,00 ?	suburban
6	5	3NBSP000,00 ?	suburban
7	6	3NBSP000,00 ?	suburban
8	7	3NBSP000,00 ?	suburban

```sql
--Задание 4 - Хранимые процедуры
--Для повышения цен в пригородные поезда на 20%
create or replace function railways.up_price_to_suburban_trains(
) returns void as $up_price_to_suburban_trains$
begin
            UPDATE railways.seats
            SET price = price * 1.2
            WHERE scheduled_train_carriage_id in
            (
                select
                    scheduled_train_carriages.scheduled_train_carriage_id
                from
                    railways.trains,
                    railways.timetable,
                    railways.scheduled_train_carriages
                where
                        train_type = 'suburban'
                  and trains.train_id = timetable.train_id
                  and railways.scheduled_train_carriages.scheduled_train_id = railways.timetable.scheduled_train_id

            );
end;
$up_price_to_suburban_trains$ language plpgsql;

select railways.up_price_to_suburban_trains();
```

	seat_number	price	train_type
1	18	1NBSP088,64 ?	suburban
2	1	1NBSP088,64 ?	suburban
3	2	1NBSP088,64 ?	suburban
4	3	1NBSP088,64 ?	suburban
5	4	1NBSP088,64 ?	suburban
6	5	1NBSP088,64 ?	suburban
7	6	1NBSP088,64 ?	suburban
8	7	1NBSP088,64 ?	suburban

2. Для создания нового рейса на поезд.

	timetable_id	train_id	status	departure_time	arrival_time	schedule
1	2	1	departured	2023-11-27 04:05:06.000000	2023-11-27 08:05:00.000000	
2	16	1	scheduled	2023-11-27 04:05:06.000000	2023-11-26 08:05:00.000000	
3	14	1	scheduled	2023-11-25 04:05:06.000000	2023-11-26 08:05:06.000000	
4	0	1	departured	2023-11-26 04:05:06.000000	2023-11-26 08:05:00.000000	
5	1003	1	scheduled	2023-12-26 04:05:06.000000	2023-12-26 08:05:00.000000	
6	3	1	scheduled	2023-10-31 04:05:06.000000	2023-01-31 08:05:00.000000	

```
--Для создания нового рейса на поезд.
create or replace function railways.create_scheduled_train(
    id_train        int,
    _price          money,
    _departure_time  timestamp,
    _arrival_time    timestamp
) returns integer as $create_scheduled_train$
declare
    number_of_seats_1             integer;
    number_of_seats_2             integer;
    id_scheduled_train_carriage_1 integer;
    id_scheduled_train_carriage_2 integer;
    id__scheduled_train  integer;
begin
    id_scheduled_train_carriage_1 = random()*(16);
    number_of_seats_1 = (
        select
            carriages_types.number_of_seats
        from
            railways.carriages_types,
            railways.carriages,
            railways.scheduled_train_carriages
        where
                scheduled_train_carriages.scheduled_train_carriage_id = id_scheduled_train_carriage_1
          and  scheduled_train_carriages.carriage_id = carriages.carriage_id
          and  carriages.carriage_type_id = carriages_types.carriage_type_id
    );
```

```
id_scheduled_train_carriage_2 = random()*(16);
number_of_seats_2 = (
    select
        carriages_types.number_of_seats
    from
        railways.carriages_types,
        railways.carriages,
        railways.scheduled_train_carriages
    where
            scheduled_train_carriages.scheduled_train_carriage_id = id_scheduled_train_carriage_2
      and  scheduled_train_carriages.carriage_id = carriages.carriage_id
      and  carriages.carriage_type_id = carriages_types.carriage_type_id
);

if number_of_seats_1 is null then number_of_seats_1 = 0;
end if;
if number_of_seats_2 is null then number_of_seats_2 = 0;
end if;

id__scheduled_train = nextval('railways.scheduled_train_id_seq'::regclass);
insert into railways.scheduled_trains(scheduled_train_id)
values (id__scheduled_train);
```

```
    insert into railways.timetable(timetable_id, train_id, status, departure_time, arrival_time, scheduled_train_id)
    values (nextval('railways.scheduled_train_carriage_id_seq'::regclass), id_train, 'scheduled', _departure_time, _arrival_time, id__scheduled_train);


    insert into railways.scheduled_train_carriages(scheduled_train_carriage_id, carriage_id, scheduled_train_id, carriage_order_number)
    values (nextval('railways.scheduled_train_carriage_id_seq'::regclass), id_scheduled_train_carriage_1, id__scheduled_train, 1);

    insert into railways.scheduled_train_carriages(scheduled_train_carriage_id, carriage_id, scheduled_train_id, carriage_order_number)
    values (nextval('railways.scheduled_train_carriage_id_seq'::regclass), id_scheduled_train_carriage_2, id__scheduled_train, 2);

    for i in 1..number_of_seats_1 loop
            insert into railways.seats(seat_id, seat_number, is_empty, price, scheduled_train_carriage_id)
            values (nextval('railways.seat_id_seq'::regclass), i, true, _price, id_scheduled_train_carriage_1);
        end loop;

    for i in 1..number_of_seats_2 loop
            insert into railways.seats(seat_id, seat_number, is_empty, price, scheduled_train_carriage_id)
            values (nextval('railways.seat_id_seq'::regclass), i, true, _price, id_scheduled_train_carriage_2);
        end loop;

    return number_of_seats_1 + number_of_seats_2;
end;
$create_scheduled_train$ language plpgsql;

select railways.create_scheduled_train( id_train: 1, price: 3000::money, '2023-12-26 04:05:06'::timestamp, '2023-12-26 08:05:00'::timestamp);
```

	timetable_id	train_id	status	departure_time	arrival_time	schedule
1	2	1	departured	2023-11-27 04:05:06.000000	2023-11-27 08:05:00.000000	
2	16	1	scheduled	2023-11-27 04:05:06.000000	2023-11-26 08:05:00.000000	
3	14	1	scheduled	2023-11-25 04:05:06.000000	2023-11-26 08:05:06.000000	
4	0	1	departured	2023-11-26 04:05:06.000000	2023-11-26 08:05:00.000000	
5	1003	1	scheduled	2023-12-26 04:05:06.000000	2023-12-26 08:05:00.000000	
6	1006	1	scheduled	2023-12-26 04:05:06.000000	2023-12-26 08:05:00.000000	
7	3	1	scheduled	2023-10-31 04:05:06.000000	2023-01-31 08:05:00.000000	

3. Для формирования общей выручки по продаже билетов за сутки.

```
--Для формирования общей выручки по продаже билетов за сутки.
create or replace function railways.get_money_for_day(
    data timestamp
) returns money as $get_money_for_day$
declare
    get_money_plus  money;
    get_money_minus  money;
    get_money  money;
begin
    get_money_plus = (
        select sum(seats.price)
        from railways.seats, railways.tickets
        where
            tickets.buying_time <= date_trunc('day', data + interval '1 day')
            and tickets.buying_time >= date_trunc('day', data)
            and tickets.seat = seats.seat_id
            and tickets.status = 'sold'
    );
    get_money_minus = (
        select sum(seats.price)
        from railways.seats, railways.tickets
        where
                tickets.buying_time <= date_trunc('day', data + interval '1 day')
          and tickets.buying_time >= date_trunc('day', data)
          and tickets.seat = seats.seat_id
          and tickets.status = 'returned'
    );
    if get_money_plus::numeric is null then get_money_plus = 0::money;
    end if;
    if get_money_minus::numeric is null then get_money_minus = 0::money;
    end if;
    get_money = (get_money_plus::numeric - get_money_minus::numeric);
    return get_money;
end;
$get_money_for_day$ language plpgsql;

select railways.get_money_for_day( data: '2023-11-25 09:05:06'::timestamp);
```

```
select railways.get_money_for_day( data: '2023-11-25 09:05:06'::timestamp);
```

Output | railways.get_money_f...06'::timestamp):money

1 row

	get_money_for_day
1	1NBSP000,00 ?

```
select tickets.buying_time, tickets.seat, seats.price
from railways.tickets, railways.seats
where tickets.seat = seats.seat_id;
```

Output | Result 9

2 rows

	buying_time	seat	price
1	2023-11-25 09:05:06.000000	1080	500,00 ?
2	2023-11-25 09:05:06.000000	1081	500,00 ?

Создание необходимых триггеров:

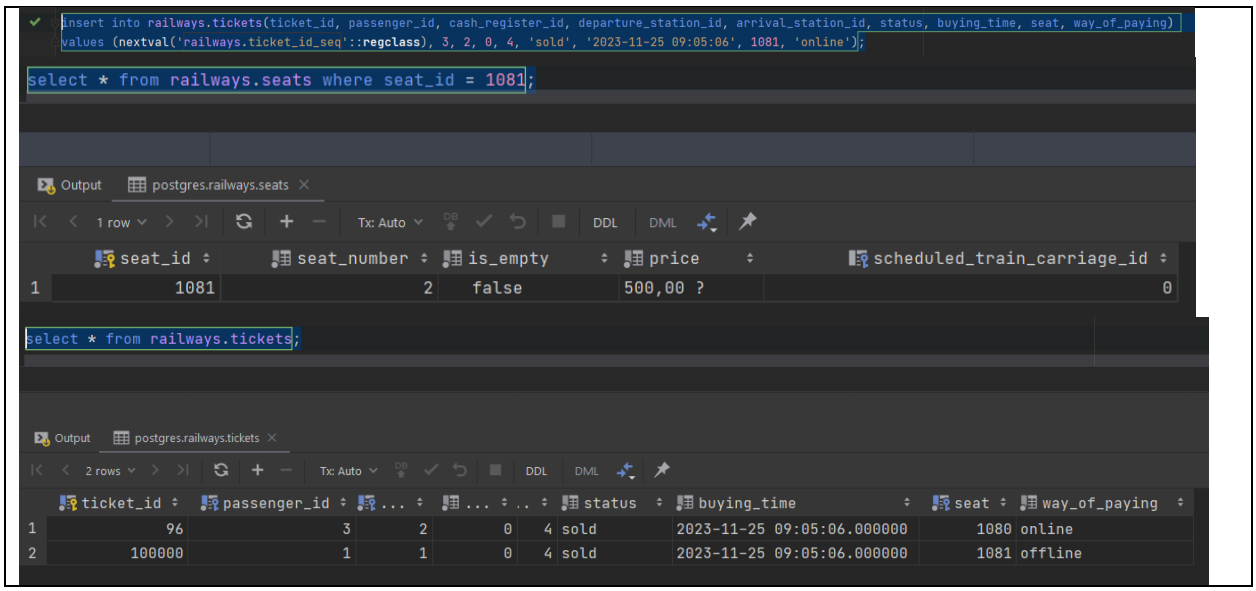
```
create trigger check_is_empty_status before insert on railways.tickets
    for each row execute procedure fn_check_is_empty_status();
drop trigger check_is_empty_status on railways.tickets;

create or replace function fn_check_is_empty_status() returns trigger as $fn_check_is_empty_status$
begin
    if (select distinct seats.is_empty
        from railways.seats, railways.tickets
        where new.seat = seats.seat_id) then return new;
    else
        return null;
    end if;
end;
$fn_check_is_empty_status$ language plpgsql;
```

```
select * from railways.tickets;
```

Output | postgres.railways.tickets

2 rows | Tx: Auto | DDL | DML

	ticket_id	...	...	de...	a...	status	buying_time	seat	way_of_paying
1	96	3	2	0	4	sold	2023-11-25 09:05:06.000000	1080	online
2	100000	1	1	0	4	sold	2023-11-25 09:05:06.000000	1081	offline



Вывод

В ходе лабораторной работы была освоена работа с процедурами и триггерами.