# Министерство науки и высшего образования Российской ФедерацииФедеральное государственное автономное образовательное учреждение высшего образования

# «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТИТМО»

Факультет инфокоммуникационных технологий

#### Отчет

по лабораторной работе №3 «процедуры, функции, триггеры в PostgreSQL» по дисциплине **«Базы данных»** 

Выполнила: Кириллова В.Е.

Факультет: ИКТ

Группа: К3240

Проверила: Говорова М. М.



**Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

#### Практическое задание:

- 1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4):
  - Для поиска билетов в заданный пункт назначения.
  - Создание новой кассы продажи билетов.
  - Определить расход топлива по всем маршрутам за истекший месяц.
- 2. событий Создать логирования вставки, триггер для удаления, базе PostgreSQL (согласно редактирования данных В данных индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

#### Вариант 8. БД «Аэропорт»

Описание предметной области: Необходимо обеспечить продажу билетов на нужный рейс, при отсутствии билетов (необходимого количества билетов) предложить билет на ближайший рейс.

БД должна содержать следующий минимальный набор сведений: Бортовой номер самолета. Тип самолета. Количество мест. Страна. Производитель. Грузоподъемность. Скорость. Дата выпуска. Налет в часах. Дата последнего ремонта. Назначение самолета. Расход топлива. Код экипажа. Паспортные данные членов экипажа. Номер рейса. Дата вылета. Время вылета. Аэропорт вылета. Аэропорт назначения. Расстояние. Транзитные посадки (прилет, вылет, аэропорт, время в аэропорту). ФИО пассажира. Паспортные данные. Номер места. Тип места. Цена билета. Касса продажи билета (возможен электронный билет) (номер и адрес).

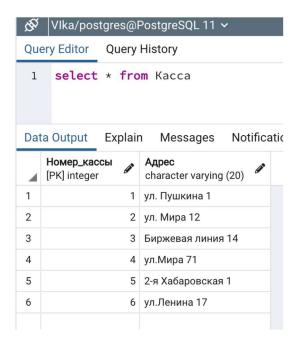
#### ВЫПОЛНЕНИЕ

- 1) Создать процедуры/функции согласно индивидуальному заданию
- 1. Создание новой кассы продажи билетов:
- Выполнение функции

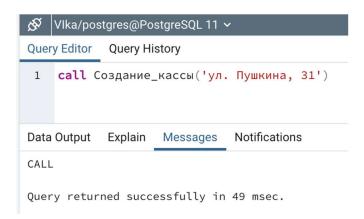
Query Editor Query History

```
1 create procedure Создание_кассы
2 (
3 Aдрес varchar
4 )
5 LANGUAGE SQL
6 AS $$
7 INSERT INTO Kacca VALUES ((select max(Номер_кассы) from Kacca), Адрес);
8 $$;
```

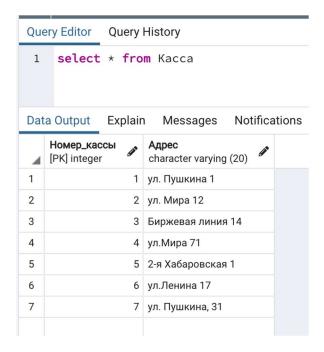
• Адреса касс до выполнения процедуры:



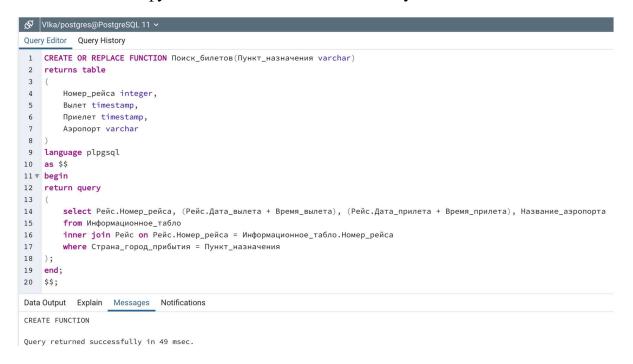
• Выполнения процедуры (Добавление нового адреса для кассы):



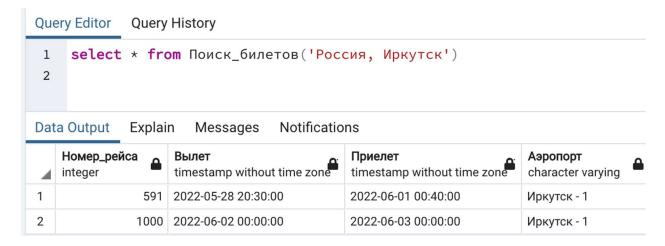
• Новый список адресов:



- 2. Для поиска билетов в заданный пункт назначения:
- Создание функции для поиска билетов в пункт назначения:



• Результат выполнения функции для поиска билетов:

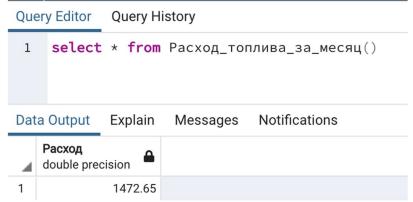


### 3. Определить расход топлива по всем маршрутам за истекший месяц:

• Создание функции для определения расхода топлива:

```
create or replace function Расход_топлива_за_месяц()
   returns table
       Pacxoд double precision
   language plpgsql
   as $$
10
  select sum(Время_в_воздухе * Расход_топлива) as Расход
12
13
       select Время_в_воздухе, Марка_самолета
15
16
          select (extract(epoch from (Время_полета - Пересадка))/3600) as Время_в_воздухе, Бортовой_номер
18
19
              select Рейс.Номер_рейса, Рейс.Бортовой_номер, (Дата_прилета + Время_прилета) - (Дата_вылета + Время_вылета) аs Время_полета, sum(Дата_и_врем
21
              inner join Рейс on Рейс.Номер_рейса = Информационное_табло.Номер_рейса
22
              inner join Транзитные_посадки on Транзитные_посадки.Номер_рейса = Информационное_табло.Номер_рейса
              where Дата_вылета > ('now'::date - '1 month'::interval)
24
             group by Рейс.Номер_рейса, Время_полета
25
27
       as Список
28
       inner join Самолет on Самолет.Бортовой_номер = Список.Бортовой_номер
             inner join Cамолет on Cамолет.Бортовой_номер = Список.Бортовой_номер
 29
 30 )
 31
      as Списочек
       inner join Марка_самолета on Марка_самолета.Марка_самолета = Списочек.Марка_самолета
 32
 33
       );
 34
       end;
 35
       $$;
 Data Output
                 Explain
                                           Notifications
                            Messages
 CREATE FUNCTION
 Query returned successfully in 42 msec.
```

• Результат выполнения функции:



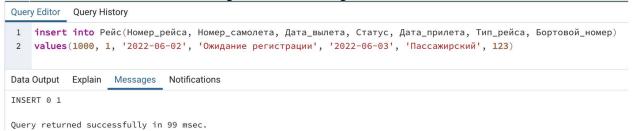
2) Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL.

1. Триггер добавляет данные в таблицу Билет (добавляет новые билеты) при добавлении нового рейса:

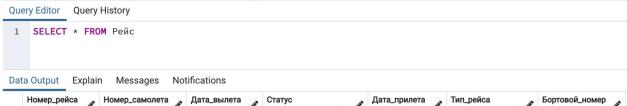


Query returned successfully in 54 msec.

• Добавляем новый рейс под номером 1000:



• Видим, что появился рейс под номером 1000:



4	<b>Номер_рейса</b> [PK] integer	<b>Hoмep_самолета</b> integer	<b>Дата_вылета</b> date	Статус character varying (20)	<b>Дата_прилета</b> date	Тип_рейса character varying (20)	Бортовой_номер integer
1	591	1	2022-05-28	В пути	2022-06-01	Пассажирский	123
2	1713	4	2022-05-28	Ожидание регистрации	2022-05-28	Пассажирский	123456
3	2368	3	2022-05-31	Завершен	2022-06-01	Пассажирский	12345
4	2637	5	2022-05-29	Ожидание регистрации	2022-05-29	Пассажирский	1234567
5	4679	2	2022-05-30	Завершен	2022-05-31	Пассажирский	1234
6	7375	6	2022-05-28	Ожидание регистрации	2022-05-29	Пассажирский	12345678
7	1000	1	2022-06-02	Ожидание регистрации	2022-06-03	Пассажирский	123

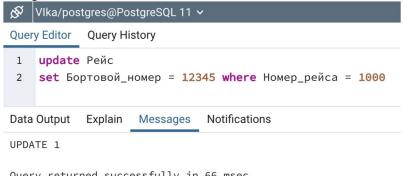
• В таблице Билет появились новые билеты на рейс 1000:

VIka/postgres@PostgreSQL 11   ✓										
Query	Editor Query History									
1	1 SELECT * FROM Билет									
Data Output Explain Messages Notifications										
4	<b>Номер_билета</b> [PK] character varying (20)	Hoмер_рейса_б integer	Номер_места integer	<b>Цена_</b> intege						
32	41	1000	12							
33	42	1000	13							
34	43	1000	14							
35	44	1000	15							
36	45	1000	16							
37	46	1000	17							
38	47	1000	18							
39	48	1000	19							

2. Триггер обновляет данные в таблице Билет при изменении самолета в рейсе (если самолет меняется на больший — добавляются новые билеты, при изменении самолета на меньший — у «лишних» билетов ставится "null" в месте):

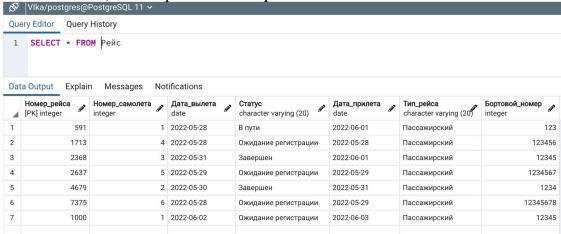
```
🔗 VIka/postgres@PostgreSQL 11 🗸
  Query Editor Query History
    1 create or replace function Обновление_рейса()
    2 returns trigger
    3 as $$
     4 ▼ begin
                         if (TG_OP = 'UPDATE')
    5 ₹
     6
    7 ₹
                                    if ((select Количество_мест from Марка_самолета where Марка_самолета = (select Марка_самолета from Самолет where Бортс
     8
                                   then
     9
                                               update Билет
  10
                                               set Homep\_mecta = null where (Homep\_mecta > (select Kоличество\_mect from Mapka\_camoлeta where Mapka\_camoлeta = (select Konuvectbo\_mecta from Mapka\_camoneta where Mapka\_camoneta = (select Konuvectbo\_mecta from Mapka\_camoneta from Mapka\_camon
  11
                                 return new;
  12
                                    elsif ((select Количество_мест from Марка_самолета where Марка_самолета = (select Марка_самолета from Самолет where Бо
  13
                                               for i in (select max(Hoмep_места::integer) + 1 from Билет where Homep_peйca_6 = new.Homep_peйca)..(select Количест
  14 ▼
  15
                                                          insert into Билет(Номер_билета, Номер_рейса_6, Номер_места) values((select max(Номер_билета::integer) + 1 from
                                             end loop;
  16
  17
                        return new;
  18
                         end if;
  19
  20 end;
  21 $$ language 'plpgsql';
  22
  23 --create trigger Триггер_на_рейс_UPDATE
  24 --after update on Рейс
  25
                        --for each row execute procedure Обновление_рейса()
  26
  Data Output Explain Messages Notifications
  CREATE FUNCTION
  Ouerv returned successfully in 36 msec.
```

Обновляем рейс – меняем самолет на меньший:

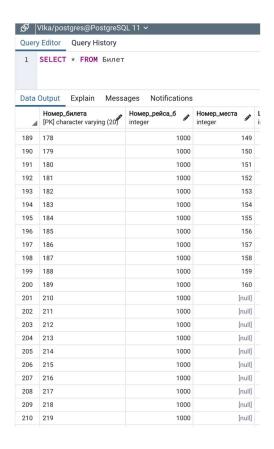


Query returned successfully in 66 msec.

• Видим, что бортовой номер изменился:



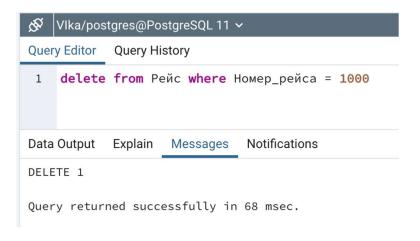
Видим, что у билетов 210-219 в номере места null:



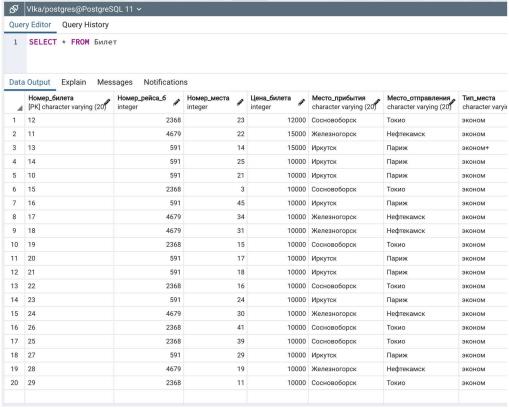
3) Триггер удаляет соответствующие билеты из таблицы Билет при удалении рейса:

```
Query Editor
           Query History
   create or replace function Удаление_рейс()
    returns trigger
 3
    as $$
 4 ▼ begin
 5 ▼
        if (TG_OP = 'DELETE')
 6
 7
            delete from Билет where Homep_peйca_6 = old.Homep_peйca;
 8
        return old;
 9
        end if;
10
    $$ language 'plpgsql';
11
12
13
    --create trigger Триггер_на_рейс_Delete
    --before delete on Рейс
14
15
        --for each row execute procedure Удаление_рейс()
Data Output
                             Notifications
          Explain
                   Messages
CREATE FUNCTION
Query returned successfully in 49 msec.
```

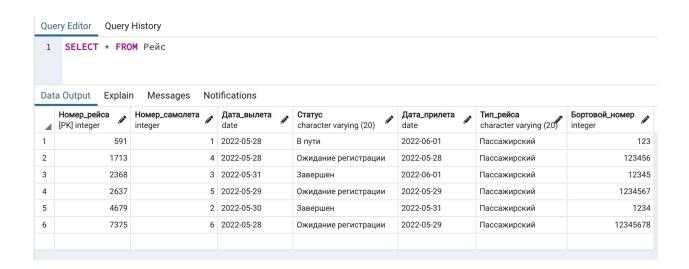
• Удаляем рейс 1000:



• Видим, что билетов на рейс номер 1000 теперь нет:



• Рейс удален:



## ВЫВОДЫ

В ходе выполнения работы, были освоены практические навыки создания представлений и запросов на выборку данных PostgreSQL, использования подзапросов при модификации данных и индексов.