Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет ИТМО»
Факультет инфокоммуникационных технологий

Лабораторная работа №3

«Процедуры, функции, триггеры в PostgreSQL»

По дисциплине
«Проектирование и реализация баз данных»

Выполнил:
Баландин И.О.
Группа:
K3240
Преподаватель:
Говорова М.М.

Санкт-Петербург

2022 г

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. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно 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. Налѐт 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. Процедура продажи билета:

- Выполнение функции

```
create or replace procedure Продажа_билета
(
    Паспорт varchar,
    Номер_рейса_ integer
)
language sql
as $$
update Билет
set Статус_билета = 'Куплен', ФИО_пассажира = Паспорт, Дата_и_время_покупки = 'now'::timestamp
where Номер_рейса = Номер_рейса_ and Номер_места = (select min(Номер_места) from Билет where Номер_рейса = Номер_рей
$$;
```

- Таблица билет до процедуры

	Номер_билета [PK] integer	Цена_билета integer	Статус_билета character varying (25)	ФИО_пассажира character varying (70)	Место_прибытия character varying (20)	Номер_места integer	Номер_рейса integer	Дата_и_время_покупки timestamp without time zone
1	1002	500	in_proccess	9876543211	Pushkino	12	1002	2022-05-25 18:39:05
2	1009	1000	Продается	[null]	Moscow_city	5	603	[null]
3	1007	1000	Продается	[null]	Moscow_city	3	603	[null]
4	1008	1000	Продается	[null]	Moscow_city	4	603	[null]

- Выполнения процедуры :

```
call Продажа_билета('1234567899', 603)
```

- Обновленная таблица:

	Номер_билета [PK] integer	Цена_билета integer	Статус_билета character varying (25)	ФИО_пассажира character varying (70)	Место_прибытия character varying (20)	Номер_места integer	Номер_рейса integer	Дата_и_время_покупки timestamp without time zone
1	1004	20000	in_proccess	0990123321	Krasnoyarsk	11	1001	2022-05-30 07:30:45
2	1002	500	in_proccess	9876543211	Pushkino	12	1002	2022-05-25 18:39:05
3	1001	20000	in_proccess	0990123321	Krasnoyarsk	10	1001	2022-05-30 07:20:45
4	1005	1000	Куплен	1234567899	Moscow_city	1	603	2022-06-27 13:42:30.184475

2. Для возврата билетов:

- Создание процедуры для возврата билетов:

```
create or replace procedure Возврат_билета
(
    Паспорт varchar,
    Номер_рейса_ integer
)
language sql
as $$
update Билет
set Статус_билета = 'Продается', ФИО_пассажира = null, Дата_и_время_покупки = null
where ФИО_пассажира = Паспорт and Номер_рейса = Номер_рейса_;
$$;
```

- Удаляем, добавленный в предыдущем пункте, билет:

```
call Возврат_билета('1234567899', 603)
```

29	1005	1000	Продается	[null]	Moscow_city	1	603	[null]

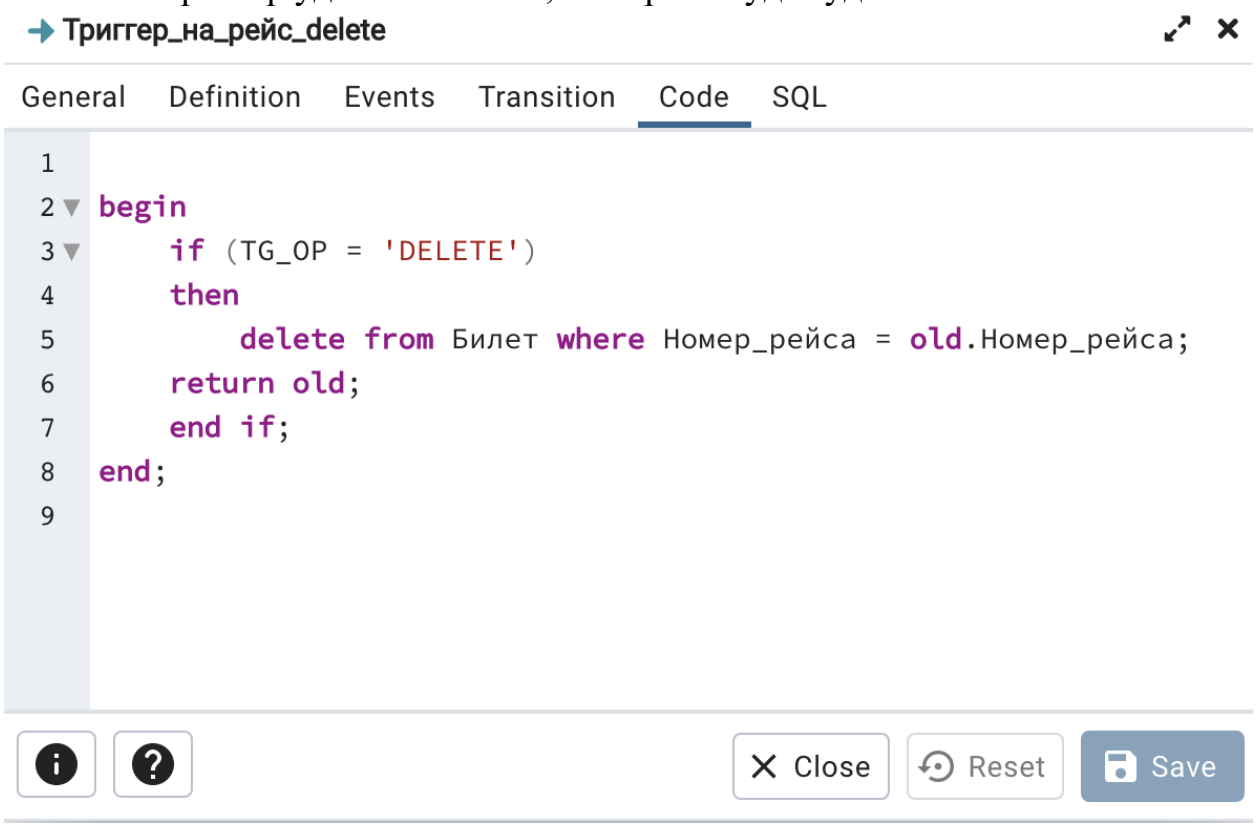
3. Добавление нового рейса:

- Процедура для добавления рейса:

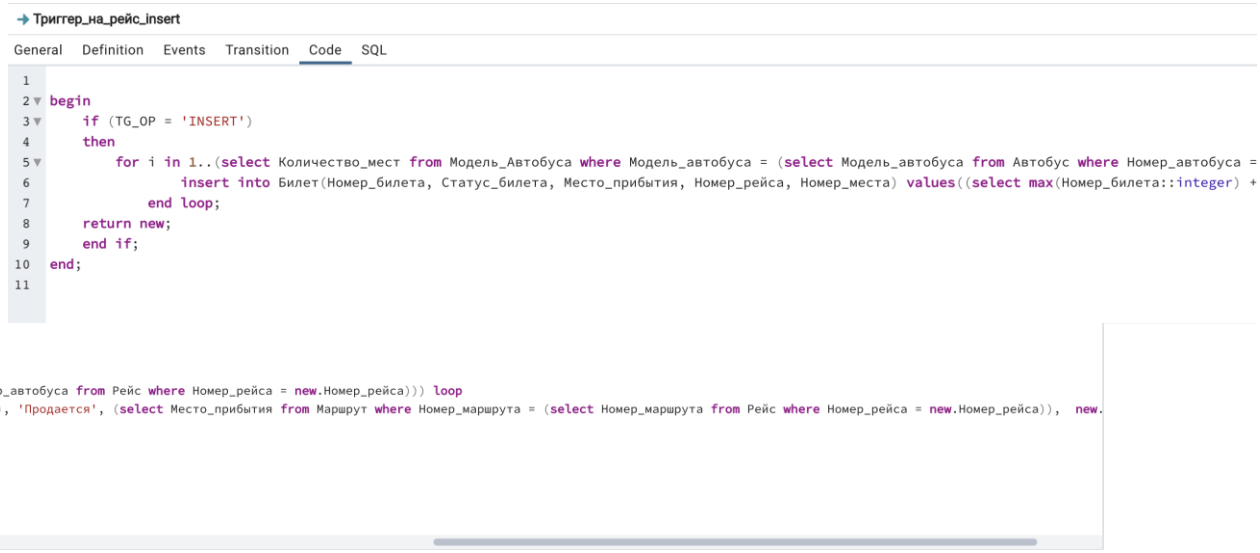
```
create or replace procedure Добавление_рейса
(
    Номер_рейса_ integer,
    Дата_отправления_ date,
    Дата_прибытия_ date,
    Номер_маршрута_ integer,
    Номер_автобуса_ varchar,
    Номер_водителя_ integer
)
language sql
as $$
insert into Рейс(Номер_рейса, Дата_отправления, Дата_прибытия, Статус_рейса, Номер_маршрута, Номер_автобуса, Номер_
values(Номер_рейса_, Дата_отправления_, Дата_прибытия_, 'Ожидание регистрации', Номер_маршрута_, Номер_автобуса_, Н
$$;
```

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

1. Триггер удаляет Билеты, если рейс будет удален



1. Триггер создает билеты по количеству мест в автобусе, назначенному на рейс:




```
begin
    if (TG_OP = 'INSERT')
    then
        for i in 1..(select Количество_мест from Модель_Автобуса where Модель_автобуса = (select Модель_автобуса from Автобус where Номер_автобуса =
                insert into Билет(Номер_билета, Статус_билета, Место_прибытия, Номер_рейса, Номер_места) values((select max(Номер_билета::integer) +
            end loop;
    return new;
    end if;
end;
```

```
= (select Номер_автобуса from Рейс where Номер_рейса = new.Номер_рейса))) loop
+ 1 from Билет), 'Продается', (select Место_прибытия from Маршрут where Номер_маршрута = (select Номер_маршрута from Рейс where Номер_рейса = new.Номер_рейса)),  new.
```

1. Триггер обновляет количество билетов при изменении номера автобуса

```
begin
    if (TG_OP = 'UPDATE')
    then
        if ((select Количество_мест from Модель_Автобуса where Модель_автобуса = (select Модель_автобуса from Автобус where Номер_автобуса = new.Номер_автобуса)) < (s
        then
            update Билет
            set Номер_места = null where (Номер_места > (select Количество_мест from Модель_Автобуса where Модель_автобуса = (select Модель_автобуса from Автобус wher
        return new;
        elsif ((select Количество_мест from Модель_Автобуса where Модель_автобуса = (select Модель_автобуса from Автобус where Номер_автобуса = new.Номер_автобуса)) >
        then
            for i in (select max(Номер_места::integer) + 1 from Билет where Номер_рейса = new.Номер_рейса)..(select Количество_мест from Модель_Автобуса where Модель_
                insert into Билет(Номер_билета, Статус_билета, Место_прибытия, Номер_рейса, Номер_места) values((select max(Номер_билета::integer) + 1 from Билет), 'П
            end loop;
    return new;
    end if;
    end if;
end;
```

ВЫВОДЫ

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