

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»

Выполнила:

студентка II курса ИКТ
группы К3243

Махнева Анастасия Денисовна

Проверила:

Говорова Марина Михайловна

Санкт-Петербург
2022

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:

Вариант 1:

Создать хранимые процедуры и функции (согласно индивидуальному заданию, часть 4):

- для поиска билетов в заданный пункт назначения;
- создания новой кассы продажи билетов;
- определить расход топлива по всем маршрутам за истекший месяц.

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.

Выполнение:

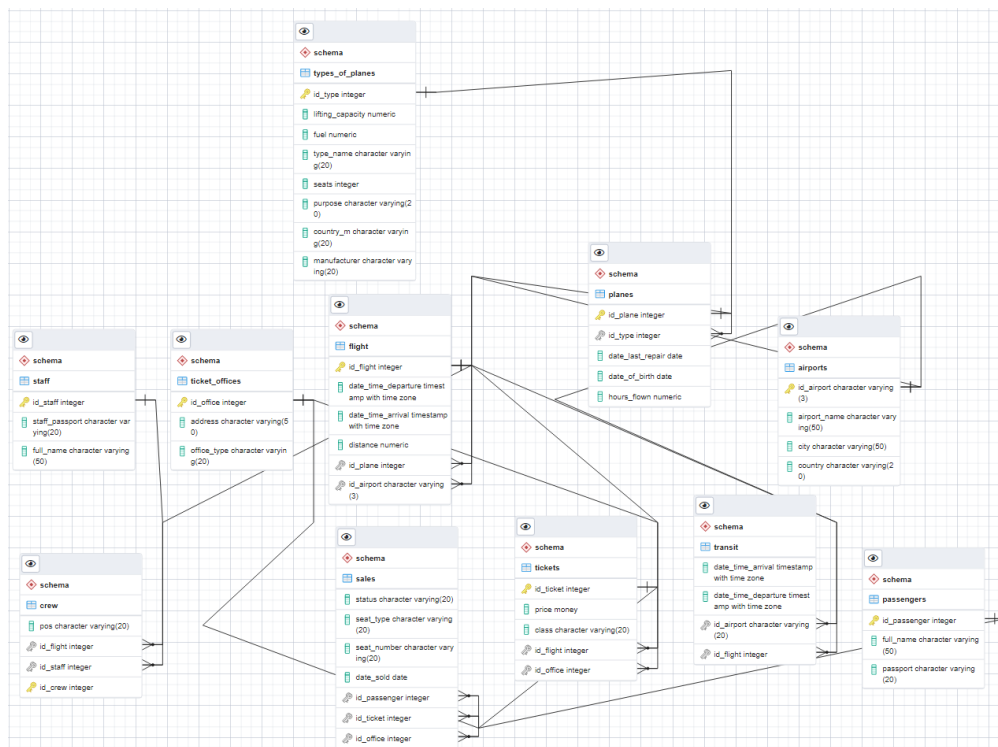


Рисунок 1 – Схема базы данных

Сначала была создана хранимая функция:

1. Поиск билетов в заданный пункт назначения. (Созданная БД не учитывает пункт назначения, а только пункт отправления, поэтому будем искать билеты из заданного пункта отправления. Пусть наш заданный пункт – Москва).

```

1 CREATE FUNCTION schema.show_tickets_MSK()
2 RETURNS TABLE(id_flight int, "Тип билета" varchar(50), "Дата отправления" timestamp with time zone)
3 AS
4 $$
5 select schema.flight.id_flight, schema.tickets.class, schema.flight.date_time_departure
6 from schema.tickets
7 left join schema.flight on tickets.id_flight=flight.id_flight
8 left join schema.airports on flight.id_airport=airports.id_airport
9 where airports.city='Moscow' and airports.country='Russia'
10 $$
11 LANGUAGE SQL

```

Рисунок 2 – Создание функции

	id_flight integer	Тип билета character varying	Дата отправления timestamp with time zone
1	5	economy	2022-06-12 15:00:00+03
2	9	business	2022-06-17 15:00:00+03
3	7	economy	2022-06-15 17:00:00+03

Рисунок 3 – Результат функции

2. Определение расхода топлива по всем маршрутам за истекший месяц:

```

1 CREATE FUNCTION schema.fuel_used_last_month() RETURNS TABLE(id_flight int, "количество топлива" numeric)
2 LANGUAGE SQL
3 AS $$
4 select schema.flight.id_flight, distance*fuel as fuel_used
5 from schema.flight
6 left join schema.planes on flight.id_plane=planes.id_plane
7 left join schema.types_of_planes on planes.id_type=types_of_planes.id_type
8 where flight.date_time_departure >= current_date - interval '1 month'
9 and flight.date_time_departure <= current_date
10 $$;
```

Рисунок 4 – Создание функции

	id_flight integer	количество топлива numeric
1	1	1902
2	3	5070
3	2	3741.4
4	4	5676
5	5	4750.0
6	6	5940
7	7	6000

Рисунок 5 – Результат функции

Затем были создана хранимая процедура – создание новой кассы продажи билетов:

```

1 CREATE PROCEDURE schema.new_ticket_office(_address varchar(50), _officetype varchar(20))
2 LANGUAGE SQL
3 AS $$
4 insert into schema.ticket_offices (address, office_type)
5 values (_address, _officetype);
6 $$;
```

Рисунок 6 – Создание процедуры

```
1 call schema.new_ticket_office('Birzhevaya liniya, 14', 'offline')
```

Рисунок 7 – Вызов процедуры




	 id_office [PK] integer	 address character varying (50)	 office_type character varying (20)
1	1	aviasales.ru	online
2	2	airlines.aero/aeroflot/	online
3	3	Pulkovo Departure TO	offline
4	4	Bolshoe Savino TO	offline
5	5	Birzhevaya liniya, 14	offline

Рисунок 8 – Результат вызова процедуры

Далее производилась работа над триггером. Сначала была добавлена соответствующая функция, а затем добавлен сам триггер.

```
1 create or replace function schema.add_to_log() returns trigger as $$
2 DECLARE
3     mstr varchar(30);
4     astr varchar(100);
5     retstr varchar(254);
6 BEGIN
7     IF TG_OP = 'INSERT' THEN
8         astr = NEW;
9         mstr := 'Add new data ';
10        retstr := mstr||astr;
11        INSERT INTO schema.logs(logtext, added, table_name) values (retstr, NOW(),
12        TG_TABLE_NAME);
13        RETURN NEW;
14    ELSIF TG_OP = 'UPDATE' THEN
15        astr = NEW;
16        mstr := 'Update data ';
17        retstr := mstr||astr;
18        INSERT INTO schema.logs(logtext, added, table_name) values (retstr, NOW(),
19        TG_TABLE_NAME);
20        RETURN NEW;
21    ELSIF TG_OP = 'DELETE' THEN
22        astr = OLD;
23        mstr := 'Remove data ';
24        retstr := mstr || astr;
25        INSERT INTO schema.logs(logtext, added, table_name) values (retstr, NOW(),
26        TG_TABLE_NAME);
27        RETURN OLD;
28    END IF;
29 END;
30 $$ LANGUAGE plpgsql;
```

Рисунок 9 – Функция для триггера

```

1 create trigger trg_flight after insert or update or delete
2 on schema.flight for each row execute function schema.add_to_log()

```

Рисунок 10 – Создание триггера

Далее попробуем модифицировать данные в таблице schema.flight.

```

1 insert into schema.flight
2 (date_time_departure, date_time_arrival, distance, id_plane, id_airport)
3 values
4 ('2022-06-16 17:05:00', '2022-06-16 19:30:00', '2290', '6', 'MSQ')

```

Рисунок 11 – Добавление данных

```

1 update schema.flight
2 set distance='2090'
3 where id_flight='21'

```

Рисунок 12 – Изменение данных

```

1 delete from schema.flight where id_flight=21

```

Рисунок 13 – Удаление данных

Проверим работу триггера, просмотрев таблицу логов:

	logtext text	added timestamp without time zone	table_name character varying
1	Add new data (21,"2022-06-16 17:05:00+03","2022-06-16 19:30:00+03",2290,6,MSQ)	2022-06-21 15:17:05.978926	flight
2	Update data (21,"2022-06-16 17:05:00+03","2022-06-16 19:30:00+03",2090,6,MSQ)	2022-06-21 15:21:45.495548	flight
3	Remove data (20,"2022-06-16 17:05:00+03","2022-06-16 19:30:00+03",2290,6,MSQ)	2022-06-21 15:23:17.334485	flight
4	Remove data (21,"2022-06-16 17:05:00+03","2022-06-16 19:30:00+03",2090,6,MSQ)	2022-06-21 15:24:04.950269	flight

Рисунок 14 – Таблица schema.logs

Выводы: были изучены методы создания хранимых процедур и функций, а также триггеров.