Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Интеллектуальные системы в гуманитарной сфере**

Направление подготовки (специальность) **45.03.04 Интеллектуальные системы в гуманитарной сфере**

Лабораторная работа №3

по дисциплине «Проектирование и реализация баз данных»

тема: «Процедуры, функции, триггеры в PostgreSQL»

Обучающийся: Шикалова Софья Сергеевна, K3242

Работа выполнена с оценкой _____

Преподаватель (и): Говорова М.М.

_____
(подпись)

Дата 15.05.2022

Санкт-Петербург 2022

Цель работы: Овладеть практическими навыками создания и использования функций и триггеров в базе данных PostgreSQL.

Практическое задание:

- Создать хранимые процедуры:
  1. Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма).
  2. Количество видов топлива, поставляемых каждой фирмой-поставщиком.
  3. Самый непопулярный вид топлива за прошедшую неделю.
- Создать необходимые триггеры

Схема, полученная с помощью Generate ERD.

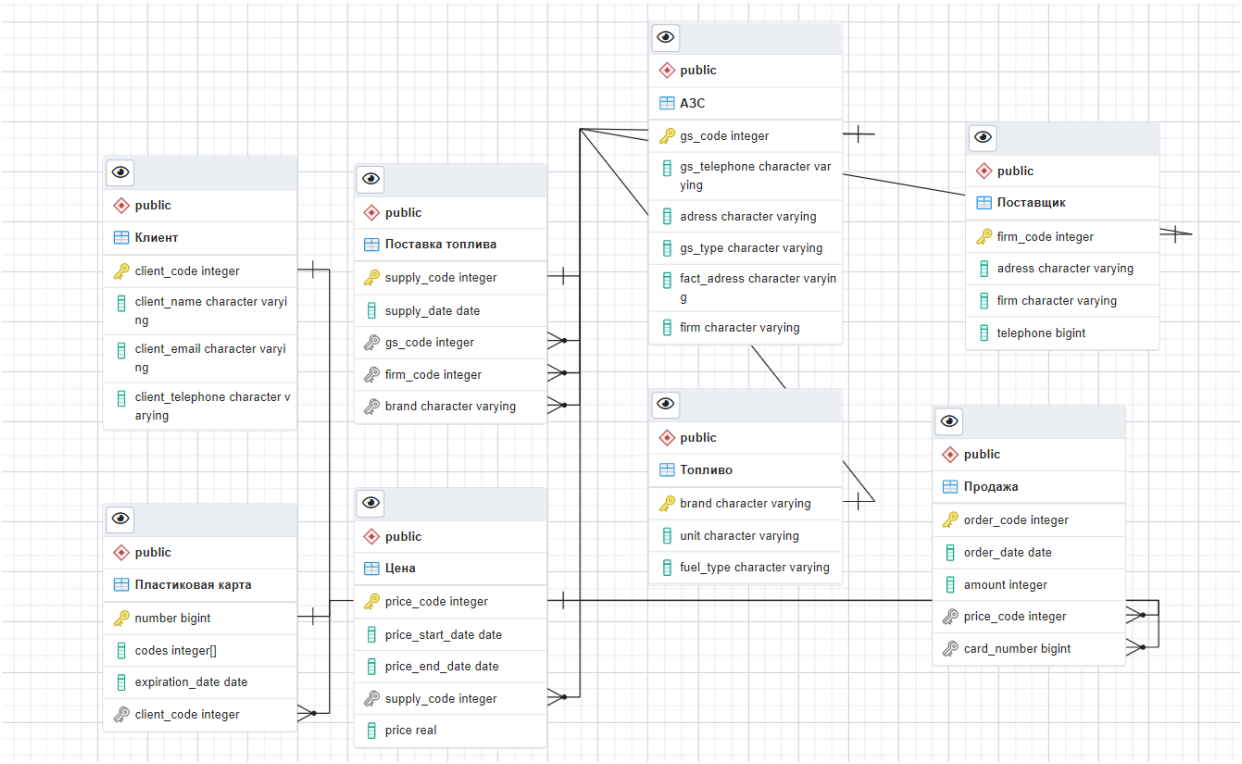

Рисунок 1 — Схема логической модели БД

Ход выполнения работы:

I. Хранимые процедуры:

1. Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма).

Query Editor

```
5  CREATE OR REPLACE FUNCTION public.order_data(
6      code integer,
7      cur_date date)
8      RETURNS TABLE(client_code integer, client_name character varying, client_telephone character varying,
9                    client_email character varying, order_date date, amount integer, total double precision)
10     LANGUAGE 'plpgsql'
11     COST 100
12     VOLATILE PARALLEL UNSAFE
13     ROWS 1000
14
15 AS $BODY$
16 BEGIN
17 RETURN QUERY
18 SELECT CLIENT.client_code, CLIENT.client_name, CLIENT.client_telephone,
19     CLIENT.client_email, SELL.order_date, SELL.amount, (SELL.amount * PRICE.price)
20 FROM public.Клиент CLIENT
21 INNER JOIN public."Пластиковая карта" CARD ON CLIENT.client_code = CARD.client_code
22 INNER JOIN public.Продажа SELL ON SELL.card_number = CARD.number
23 INNER JOIN public.Цена PRICE ON PRICE.price_code = SELL.price_code
24 WHERE CLIENT.client_code = code AND SELL.order_date = cur_date;
25 END;
26 $BODY$;
27
28 ALTER FUNCTION public.order_data(integer, date)
29     OWNER TO postgres;
```

Messages | Query History | Data Output | Explain | Notifications

Рисунок 2 — Первая функция

Query Editor

```
1  SELECT * FROM order_data(4, '2022-01-13')
```

Messages | Query History | Data Output | Explain | Notifications

	client_code integer	client_name character varying	client_telephone character varying	client_email character varying	order_date date	amount integer	total double precision
1	4	Zulhiza Galiakberdina	79133425463	tatarochka@mail.ru	2022-01-13	12	665.400009155273

Рисунок 3 — Вызов первой функции

2. Количество видов топлива, поставляемых каждой фирмой-поставщиком.

Query Editor

```
CREATE OR REPLACE FUNCTION public.fuel_types(
    )
    RETURNS TABLE(firm character varying, number_fuel bigint)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
RETURN QUERY
SELECT SUPPLIER.firm, COUNT(FUEL.brand)
FROM public."Поставка топлива" SUPPLY
INNER JOIN public.Поставщик SUPPLIER ON SUPPLIER.firm_code = SUPPLY.firm_code
INNER JOIN public.Топливо FUEL ON FUEL.brand = SUPPLY.brand
GROUP BY SUPPLIER.firm, SUPPLY.firm_code;
END;
$BODY$;

ALTER FUNCTION public.fuel_types()
    OWNER TO postgres;
```

Рисунок 4 — Вторая функция

Query Editor

```
SELECT public.fuel_types()
```

Messages | Query History | Data Output | Explain | Notifications

	fuel_types record
1	(Gasprom,1)
2	(Rosneft,5)
3	(Lukoil,3)

Рисунок 5 — Вызов второй функции

3. Самый непопулярный вид топлива за прошедшую неделю.

Query Editor

```
CREATE FUNCTION unpopular() RETURNS TABLE(
    brand varchar
)
AS
$$
BEGIN
RETURN QUERY
SELECT FUEL.brand
FROM public.Топливо FUEL
WHERE FUEL.brand = (SELECT FUEL.brand
FROM public."Поставка топлива" SUPPLY
INNER JOIN public.Поставщик SUPPLIER ON SUPPLIER.firm_code = SUPPLY.firm_code
INNER JOIN public.Топливо FUEL ON FUEL.brand = SUPPLY.brand
INNER JOIN public.Цена PRICE ON PRICE.supply_code = SUPPLY.supply_code
INNER JOIN public.Продажа SELL ON PRICE.price_code = SELL.price_code
WHERE SELL.order_date > (current_timestamp - interval '1 week') AND SELL.amount = (
SELECT SUM(SELL.amount)
FROM public.Продажа SELL
GROUP BY SELL.price_code
ORDER BY SUM(SELL.amount) DESC
LIMIT 1)
GROUP BY FUEL.brand);
END;
$$
LANGUAGE plpgsql;
```

Рисунок 6 — Третья функция

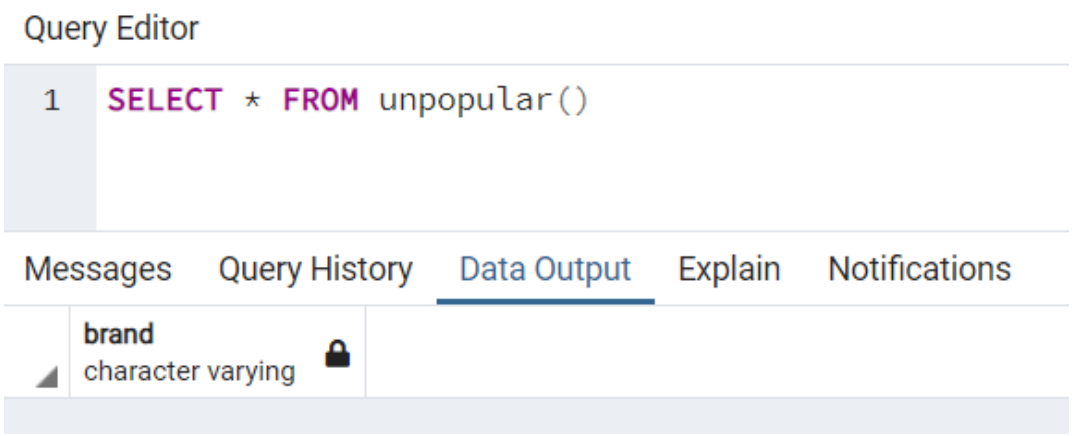

Рисунок 7 — Вызов третьей функции (За прошлую неделю в БД нет записей)

II. Триггеры:

Создаём логгер для таблицы с данными о клиентах с триггером на добавление, изменение и удаление записей.

```
CREATE OR REPLACE FUNCTION logger() RETURNS TRIGGER AS $$
DECLARE
   mstr varchar;
   astr varchar;
   retstr varchar;
BEGIN
```

```
            IF TG_OP = 'UPDATE' THEN
               astr = NEW;
               mstr := 'Updated';
               retstr := mstr || astr;
               INSERT INTO logs("message", "timestamp") VALUES (retstr,NOW());
                        RETURN NEW;
            ELSIF TG_OP = 'INSERT' THEN
               astr = NEW;
               mstr := 'Added';
               retstr := mstr || astr;
               INSERT INTO logs("message", "timestamp") VALUES (retstr,NOW());
               RETURN NEW;
            END IF;
         END;
$$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION

Query returned successfully in 282 msec.
```

Рисунок 8 — Создание функции

Query Editor

```
1  CREATE TRIGGER trigger_clients
2  AFTER INSERT OR UPDATE OR DELETE ON Клиент FOR EACH ROW EXECUTE PROCEDURE logger();
3
```

Messages | Query History | Data Output | Explain | Notifications

```
CREATE TRIGGER

Query returned successfully in 140 msec.
```

Рисунок 9 — Создание триггера

Query Editor

```
1  INSERT INTO public.Клиент (client_code, client_name, client_email, client_telephone)
2  VALUES (9, 'Gennadiy Antipov', 'antigena@gmail.com', '74645335250')
```

Messages | Query History | Data Output | Explain | Notifications

```
INSERT 0 1

Query returned successfully in 131 msec.
```

Рисунок 10 — Добавление клиента

Query Editor

```
1  UPDATE public.Клиент SET client_telephone = '79465448118'
2  WHERE client_code = 2
```

Messages | Query History | Data Output | Explain | Notifications

```
UPDATE 1

Query returned successfully in 118 msec.
```

Рисунок 11 — Обновление телефона клиента

Query Editor

```
1  SELECT * FROM public.logs
```

Messages | Query History | Data Output | Explain | Notifications

	message text	timestamp timestamp with time zone
1	Added(8,"Milka Ivanova",moo@yahoo.com,70382724431)	2022-05-27 21:35:23.538485...
2	Updated(2,"Varvara Sladkova",sladenko@yahoo.com,79...	2022-05-27 21:37:58.357167...

Рисунок 12 — Таблица логов

Вывод:

В ходе выполнения данной лабораторной работы я создала хранимые процедуры и триггер для базы данных PostgreSQL согласно индивидуальному заданию.