Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №3

«Работа с триггерами и функциями в POSTGRESQL»

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

Автор:

Бахирева Ирина Константиновна

Факультет:

Инфокоммуникационных технологий

Группа:

К3240

Преподаватель: 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

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4)

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.

Выполнение:

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4)

Задание 4. Создать хранимые процедуры:

- Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива).
- Количество видов топлива, поставляемых каждой фирмой-поставщиком.
- Самый непопулярный вид топлива за прошедшую неделю.

Сведения

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

Создание типа данных:

```
CREATE TYPE t_infoff AS (bill int, date date, qnt_fuel bigint);
```

Создание функции:

```
CREATE OR REPLACE FUNCTION infoff(bll integer, dt date) RETURNS SETOF t_infoff
```

```
AS
$$

BEGIN

        RETURN QUERY

        SELECT bll,dt, (SELECT SUM(station.service.qnt_fuel) FROM
station.service WHERE station.service.bill_number=bll GROUP BY
station.service.bill_number )

        FROM station.service, station.client

        WHERE station.service.bill_number=bll AND

        station.service.date=dt GROUP BY station.service.bill_number;


END;

$$ LANGUAGE plpgsql;
```

```
avto=# CREATE OR REPLACE FUNCTION infoff(bll integer, dt date) RETURNS SETOF t_infoff AS
avto-# $$
avto$# BEGIN
avto$# RETURN QUERY
avto$# SELECT bll,dt, (SELECT SUM(station.service.qnt_fuel) FROM station.service WHERE station.service.bill_number=bll G
ROUP BY station.service.bill_number )
avto$# FROM station.service, station.client
avto$# WHERE station.service.bill_number=bll AND
avto$# station.service.date=dt GROUP BY station.service.bill_number;
avto$# END;
avto$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Вызов:

SELECT * FROM infoff(111111, ‘2021-12-10’);

```
avto=# SELECT * FROM station.infoff(111111, '2021-12-10');
  bill  |    date    | qnt_fuel
--------+------------+----------
 111111 | 2021-12-10 |       25
(1 строка)
```

Виды топлива

Количество видов топлива, поставляемых каждой фирмой-поставщиком

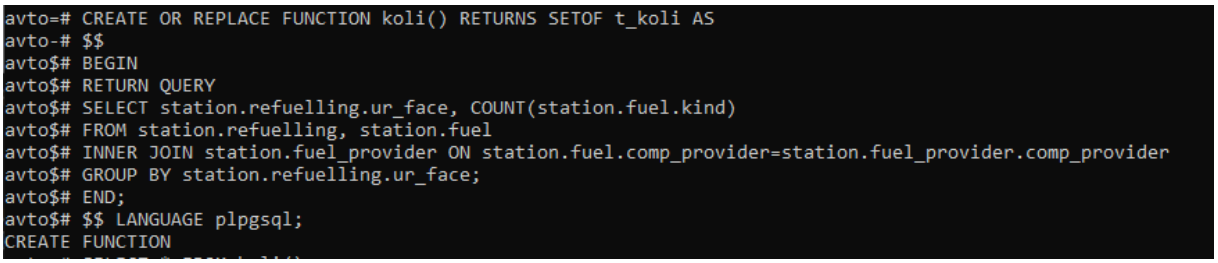
Создание типа данных:

CREATE TYPE **t_koli** AS (ur_face VARCHAR(100), qntkind bigint);

```
avto=# CREATE TYPE t_koli AS (ur_face VARCHAR(100), qntkind bigint);
CREATE TYPE
```

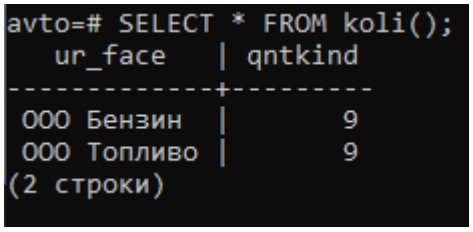
Создание функции:

```
CREATE OR REPLACE FUNCTION koli() RETURNS SETOF t_koli AS
$$
BEGIN
	RETURN QUERY
		SELECT station.refuelling.ur_face, COUNT(station.fuel.kind)
		FROM station.refuelling, station.fuel
		INNER JOIN station.fuel_provider ON
		station.fuel.comp_provider=station.fuel_provider.comp_provider
		GROUP BY station.refuelling.ur_face;
END;
$$ LANGUAGE plpgsql;
```



Вызов:

```
SELECT * FROM koli();
```

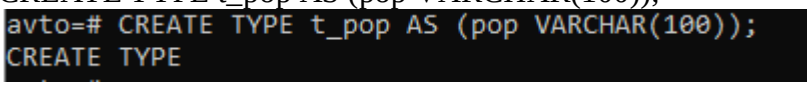


Непопулярный

Самый непопулярный вид топлива за прошедшую неделю (прошедший год)

Создание типа:

```
CREATE TYPE t_pop AS (pop VARCHAR(100));
```



Создание функции:

```
CREATE OR REPLACE FUNCTION pop() RETURNS SETOF t_pop
AS $$
BEGIN
	RETURN QUERY
	SELECT station.fuel.kind
	FROM station.fuel, station.service
	WHERE station.fuel.code=station.service.code AND
	(SELECT (EXTRACT (YEAR FROM station.service.date)))=(SELECT (EXTRACT
	(YEAR FROM CURRENT_DATE)))
```

```
        GROUP BY station.fuel.kind
        HAVING COUNT(station.service.code)<=all (SELECT
        COUNT(station.service.code) FROM station.fuel GROUP BY station.fuel.kind);
END;
$$ LANGUAGE plpgsql;
```

```
avto=# CREATE OR REPLACE FUNCTION pop() RETURNS SETOF t_pop AS $$
avto$# BEGIN
avto$# RETURN QUERY
avto$# SELECT station.fuel.kind
avto$# FROM station.fuel, station.service
avto$# WHERE station.fuel.code=station.service.code AND
avto$# (SELECT (EXTRACT (YEAR FROM station.service.date)))=(SELECT (EXTRACT (YEAR FROM CURRENT_DATE)))
avto$# GROUP BY station.fuel.kind
avto$# HAVING COUNT(station.service.code)<=all (SELECT COUNT(station.service.code) FROM station.fuel GROUP BY station.fu
el.kind);
avto$# END;
avto$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Вызов:

```
avto=# SELECT * FROM pop();
     pop
-------------
 Нормаль-АИ80
 Премиум-АИ95
 Супер-АИ95+
 Экстра-АИ98
 ЭКТО-АИ100
(5 строк)
```

Задание 5. Создать необходимые триггеры.

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.

Триггер

Создана таблица logs

Триггерная функция из лекции:

```
CREATE OR REPLACE FUNCTION station.add_to_log() RETURNS
TRIGGER AS $$
DECLARE
mstr varchar(30);
astr varchar(100);
retstr varchar(254);
BEGIN
IF TG_OP = 'INSERT' THEN
astr = NEW;
mstr := 'Add new';
retstr := mstr || astr;
INSERT INTO
station.logs(text,added,table name) values
(retstr,NOW(), TG TABLE NAME);
RETURN NEW;
ELSIF TG_OP = 'UPDATE' THEN
astr = NEW;
mstr := 'Update';
retstr := mstr || astr;
INSERT INTO
station.logs(text,added,table name) values
(retstr,NOW(), TG TABLE NAME);
RETURN NEW;
ELSIF TG_OP = 'DELETE' THEN
astr = OLD;
mstr := 'Remove';
```

```
retstr := mstr || astr;
INSERT INTO
station.logs(text,added,table name) values
(retstr,NOW(), TG TABLE NAME);
RETURN OLD;
END IF;
END;
$$ LANGUAGE plpgsql;
```

```
avto$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

CREATE TRIGGER t_trgg AFTER INSERT OR UPDATE OR DELETE ON station.service FOR EACH ROW EXECUTE PROCEDURE station.add_to_log ();

```
avto=# CREATE TRIGGER t_trgg AFTER INSERT OR UPDATE OR DELETE ON station.service FOR EACH ROW EXECUTE PROCEDURE station.
add_to_log ();
CREATE TRIGGER
```

- INSERT INTO station.service(service_sur, bill_number, rfl_code, code, qnt_fuel, date) VALUES (25, 111111, (SELECT station.refuelling.rfl_code FROM station.refuelling WHERE station.refuelling.type='АГЗС'), 7, 7, (SELECT (NOW()-INTERVAL '5 MONTH')));

```
L '5 MONTH')));
INSERT 0 1
```

- UPDATE station.service SET date=(NOW()-INTERVAL '5 MONTH') WHERE bill_number=111111;

```
avto=# UPDA
UPDATE 10
```

- DELETE FROM station.service WHERE station.service.qnt_fuel=7;

```
avto=# DEL
DELETE 5
```

```
avto=# SELECT * FROM station.logs;
                text                |           added            | table_name
------------------------------------+----------------------------+------------
 Add new(25,111111,2,7,7,2021-12-10) | 2022-05-10 00:04:06.037025 | service
 Update(1,111111,2,6,10,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(10,111111,1,5,1,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(20,111111,1,9,2,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(21,111111,1,6,3,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(7,111111,1,1,4,2021-12-10)   | 2022-05-10 00:05:45.540036 | service
 Update(15,111111,1,1,7,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(22,111111,2,7,4,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(9,111111,1,1,1,2021-12-10)   | 2022-05-10 00:05:45.540036 | service
 Update(24,111111,2,7,7,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Update(25,111111,2,7,7,2021-12-10)  | 2022-05-10 00:05:45.540036 | service
 Remove(5,111115,2,10,7,2022-02-14)  | 2022-05-10 00:07:26.064748 | service
 Remove(23,111112,2,7,7,2021-12-09)  | 2022-05-10 00:07:26.064748 | service
 Remove(15,111111,1,1,7,2021-12-10)  | 2022-05-10 00:07:26.064748 | service
 Remove(24,111111,2,7,7,2021-12-10)  | 2022-05-10 00:07:26.064748 | service
 Remove(25,111111,2,7,7,2021-12-10)  | 2022-05-10 00:07:26.064748 | service
(16 строк)
```

Удаление триггерной функции:
DROP TRIGGER t_trgg ON station,service;

```
avto=# DROP TRIGGER t_trgg ON station.service;
DROP TRIGGER
```

Выводы:

Созданы функции на запросы и триггер на вставку, удаление и редактирование данных.