

Министерство науки и высшего образования Российской Федерации  
федеральное государственное автономное образовательное учреждение  
высшего образования  
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

## **ЛАБОРАТОРНАЯ РАБОТА №5**

**Процедуры, функции, триггеры в PostgreSQL**  
**по дисциплине:**  
**«Проектирование и реализация баз данных»**

**Выполнил:**

Кондратьев Алексей Алексеевич  
Группа К3241

**Проверила:**

Горова Марина Михайловна

Санкт-Петербург  
2022

## ЦЕЛЬ РАБОТЫ:

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

**Оборудование:** компьютерный класс.

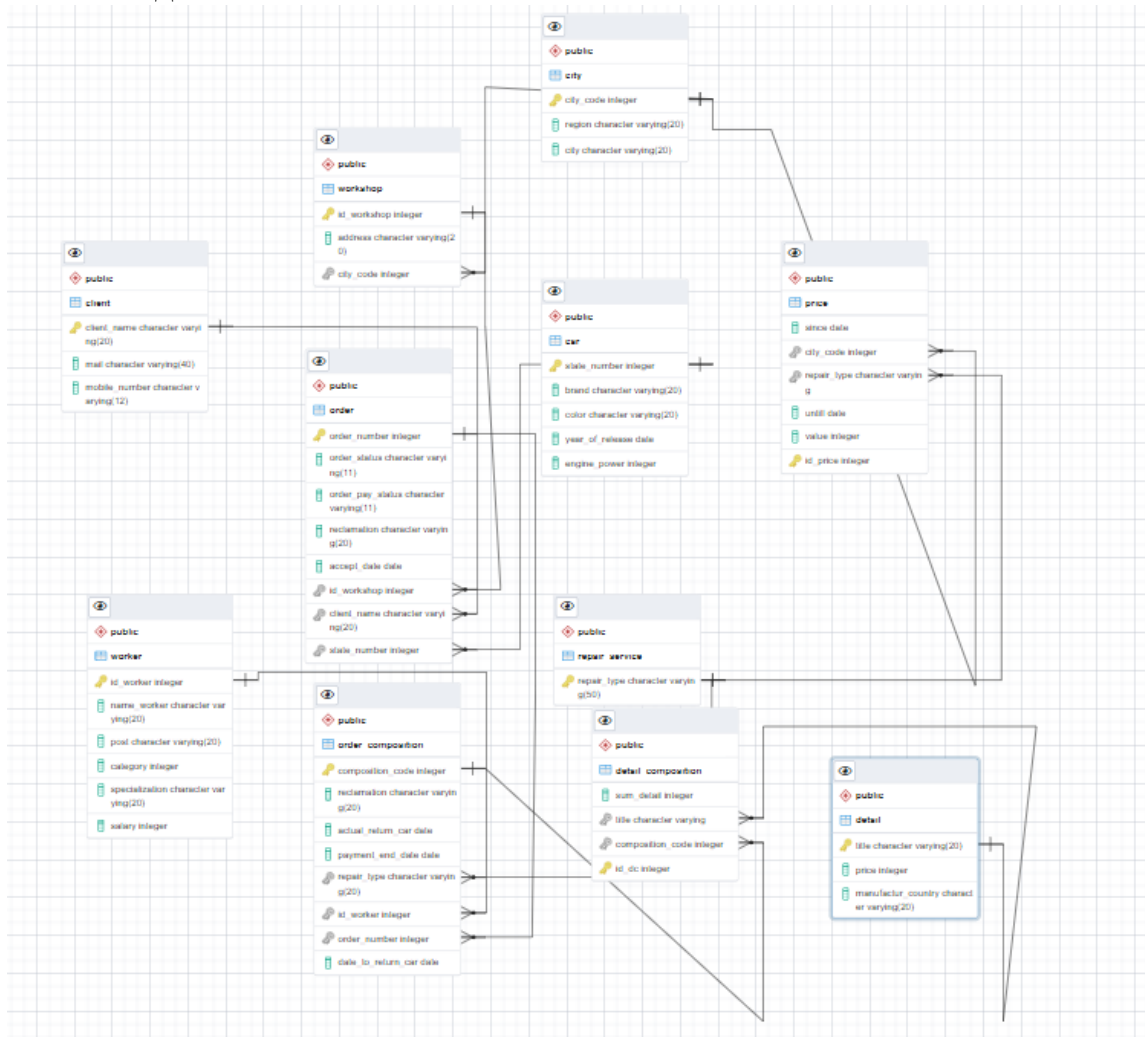
**Программное обеспечение:** СУБД PostgreSQL, SQL Shell (psql).

## Вариант 1

### Практическое задание:

- I. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
- II. 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.

## База данных:



## Выполнение:

### Задание 1. Создайте хранимые процедуры.

1) Повышения цены деталей на 10 %

```
1 select * from detail
```

Data Output Explain Messages Notifications

	<b>title</b> [PK] character varying (20)	<b>price</b> integer	<b>manufactur_country</b> character varying (20)
1	bolt	10000	China
2	screw	5000	France
3	tire	17000	Japan

```
1 create procedure detail_price_raise_10()  
2 as  
3 $$  
4 begin  
5 update public.detail  
6 set price = price * 1.1  
7 where price in (select price from detail);  
8 end;  
9 $$ language plpgsql;
```

Data Output Explain Messages Notifications

CALL

Query returned successfully in 104 msec.

	<b>title</b> [PK] character varying (20)	<b>price</b> integer	<b>manufactur_country</b> character varying (20)
1	bolt	11000	China
2	screw	5500	France
3	tire	18700	Japan

2) Для повышения разряда тех мастеров, которые отремонтировали больше 3 автомобилей.

	id_worker [PK] integer	name_worker character varying (20)	post character varying (20)	category integer	specialization character varying (20)	salary integer
1	1	Ivanov Ivan	worker	2	tire_fitter	40000
2	3	Olegov Oleg	worker	1	universal	40000
3	2	Vladov Vlad	worker	1	polisher	40000

```

1 create procedure increase_category(days int)
2 as
3 $$
4 begin
5 update public.worker
6 set category = category + 1
7 where id_worker in (select id_worker from order_composition
8 group by id_worker
9 having count(composition_code) >= days);
10 end;
11 $$ language plpgsql;

```

Data Output Explain Messages Notifications

CREATE PROCEDURE

Query returned successfully in 138 msec.

	id_worker [PK] integer	name_worker character varying (20)	post character varying (20)	category integer	specialization character varying (20)	salary integer
1	1	Ivanov Ivan	worker	2	tire_fitter	40000
2	3	Olegov Oleg	worker	2	universal	40000
3	2	Vladov Vlad	worker	2	polisher	40000

3) Сколько автомобилей отремонтировал каждый механик за истекший квартал.

```

1 create function count_repair_car(date1 date, date2 date) returns table(employee varchar, count bigint)
2 as
3 $$
4 ▼ begin
5 return query
6 select name_worker, count(composition_code)
7 from order_composition join worker on order_composition.id_worker = worker.id_worker
8 where date_to_return_car between date1 and date2
9 group by order_composition.id_worker, name_worker;
10 end;
11 $$ language plpgsql;

```

Data Output Explain Messages Notifications

CREATE FUNCTION



Query returned successfully in 51 msec.

```

1 select * from count_repair_car('2020-09-01', '2020-12-31');

```

Data Output Explain Messages Notifications

	employee character varying 	count bigint 
1	Ivanov Ivan	1
2	Vladov Vlad	3
3	Olegov Oleg	1

## Задание 2. Создать триггер для логирования событий вставки, удаления, редактирования

Создана таблица logs

```
1 create or replace function add_to_log() returns trigger as $$
2 declare
3     mstr varchar(30);
4     astr varchar(100);
5     retstr varchar(254);
6 begin
7     if TG_OP = 'INSERT' then
8         astr = NEW;
9         mstr := 'Add new data ';
10        retstr := mstr||astr;
11        insert into public.logs(text, added, table_name) values (retstr, NOW(), TG_TABLE_NAME);
12        return NEW;
13    elsif TG_OP = 'UPDATE' then
14        astr = NEW;
15        mstr := 'Update data ';
16        retstr := mstr||astr;
17        insert into public.logs(text, added, table_name) values (retstr, NOW(), TG_TABLE_NAME);
18        return NEW;
19    elsif TG_OP = 'DELETE' then
20        astr = OLD;
21        mstr := 'Remove data ';
22        retstr := mstr||astr;
23        insert into public.logs(text, added, table_name) values (retstr, NOW(), TG_TABLE_NAME);
24        return OLD;
25    end if;
26 end;
27 $$ language plpgsql;
```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 52 msec.

```
1 create trigger t_client after insert or update or delete
2 on public.client for each row execute procedure add_to_log();
```

### Вывод:

В данной работе были изучены функции и процедуры, и созданы триггеры для корректного хранения данных.