## Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

# «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

#### Отчет

по лабораторной работе №5 «ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL» по дисциплине «Проектирование и реализация баз данных»

Автор: Шалунов Андрей Ильич

Факультет: ИКТ

Группа: К3240 Преподаватель:

Говорова М.М.



Санкт-Петербург 2023

**Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

### Практическое задание:

#### Вариант 2 (тах - 8 баллов)

- 1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
- 2.1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).
  - 2.2. Создать авторский триггер по варианту индивидуального задания.

Указание. Работа выполняется в консоли SQL Shell (psql).

## Вариант 11. БД «Автомастерская»

## Описание предметной области:

Описание предметной области: Сеть автомастерских осуществляет ремонт автомобилей, используя для этих целей штат мастеров и свои мастерские. Стоимость ремонта включает цену деталей и стоимость работы.

Заработная плата мастеров составляет 50% стоимости работы.

С клиентом заключается договор на выполнение авторемонтных и профилактических работ, который сопровождается администратором. В каждом договоре может быть несколько видов услуг. Для выполнения видов работ могут требоваться детали или расходные материалы, которые предоставляет либо клиент, либо автомастерская. Если детали предоставляет автомастерская, то их стоимость включается в смету по договору.

Каждый вид работ могут выполнять разные мастера, в зависимости от их специализации. Распределение мастеров выполняет администратор.

БД должна содержать следующий минимальный набор сведений: Табельный номер сотрудника. ФИО сотрудника. Должность. Разряд мастера. Специализация. Адрес автомастерской. Дата заказа. Гос. Номер автомобиля. Марка. Мощность автомобиля. Год выпуска. Цвет автомобиля. Дата принятия в ремонт. Плановая дата окончания ремонта. Фактическая дата окончания ремонта. Вид ремонта. Стоимость вида ремонта. Название детали. Цена детали. Марка и модель автомобиля. Страна производителя. Госномер автомобиля. ФИО владельца. Номер телефона владельца. Е-mail владельца.

## Задание 4. Создать хранимые процедуры:

• Повышения цены деталей для автомобиля "Ford" на 10 %.

CREATE OR REPLACE PROCEDURE increase ford detail prices()

LANGUAGE plpgsql AS \$\$

**DECLARE** 

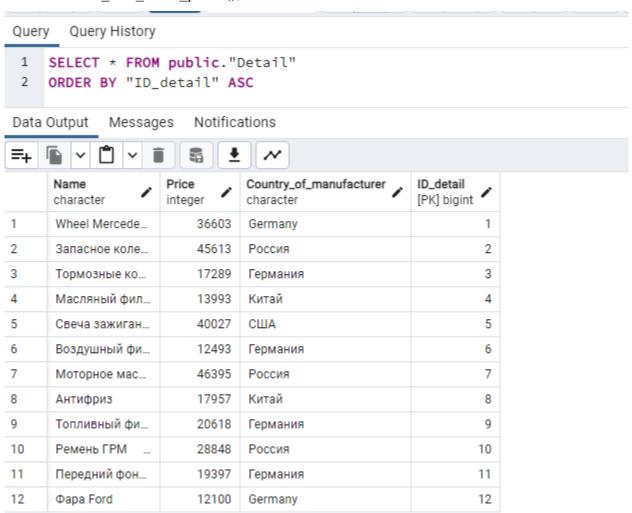
ford car brand character(100) := 'Ford';

**BEGIN** 

```
UPDATE public."Detail"
  SET "Price" = "Price" * 1.1
  WHERE "ID_detail" IN (
    SELECT d."ID_detail"
    FROM public."Detail" d
    JOIN public."Details_from_client" dfc ON d."ID_detail" = dfc."ID_detail"
                                                            dfc."ID_distribution"
    JOIN
             public."Distribution of work"
                                             dow
                                                    ON
dow."ID destribution of work"
    JOIN public."Contract" c ON dow."ID_contract" = c."ID_contract"
    JOIN public."Automobile" a ON c."ID_auto" = a."ID_auto"
    JOIN public."Model" m ON a."ID_model" = m."ID_model"
    WHERE m. "Car_brand" = ford_car_brand
  );
END;
$$;
```

2	SELECT * FROM ORDER BY "ID_			
Data	a Output Messag	es Notifica	ations	
=+				
	Name character	Price integer	Country_of_manufacturer	ID_detail [PK] bigint
1	Wheel Mercede	33275	Germany	1
2	Запасное коле	45613	Россия	2
3	Тормозные ко	17289	Германия	3
4	Масляный фил	13993	Китай	4
5	Свеча зажиган	40027	США	5
6	Воздушный фи	12493	Германия	6
7	Моторное мас	46395	Россия	7
8	Антифриз	17957	Китай	8
9	Топливный фи	20618	Германия	9
10	Ремень ГРМ	28848	Россия	10
11	Передний фон	19397	Германия	11
12	Фара Ford	11000	Germany	12

### CALL increase ford detail prices();



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

```
CREATE OR REPLACE PROCEDURE increase_mechanic_rank()

LANGUAGE plpgsql AS $$

DECLARE

repaired_cars_count integer := 3;

BEGIN

UPDATE public."Employee" as e

SET "rank" = "rank" + 1

WHERE e."ID_employee" IN (

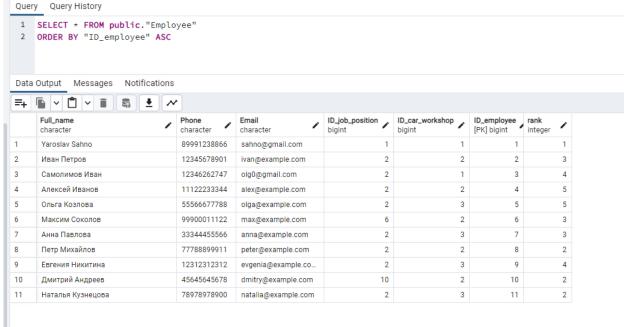
SELECT e."ID_employee"

FROM public."Employee" e

JOIN public."Contract" c ON c."ID_employee" = e."ID_employee"

JOIN public."Job_position" jb ON jb."ID_job_position" = e."ID_job_position"
```

WHERE jb. "Name" = 'Мастер-механик' GROUP BY e."ID employee" HAVING COUNT(DISTINCT c."ID auto") > repaired cars count ); END; \$\$; Query Query History SELECT \* FROM public."Employee" ORDER BY "ID\_employee" ASC Data Output Messages Notifications **=**+ ID\_employee ID\_job\_position Full\_name Phone Email ID\_car\_workshop character character character [PK] bigint integer 89991238866 1 Yaroslav Sahno sahno@gmail.com 2 Иван Петров 12345678901 ivan@example.com 2 2 2 2 3 Самолимов Иван 12346262747 olg0@gmail.com 3 11122233344 2 2 4 Алексей Иванов 4 alex@example.com 5 Ольга Козлова 55566677788 olga@example.com 2 3 5 4 99900011122 6 2 6 3 6 Максим Соколов max@example.com 33344455566 2 3 7 2 Анна Павлова anna@example.com 8 Петр Михайлов 77788899911 peter@example.com 2 2 8 1 9 Евгения Никитина 12312312312 evgenia@example.co.. 2 3 9 3 2 2 10 Дмитрий Андреев 45645645678 10 10 dmitry@example.com 78978978900 2 3 11 1 11 Наталья Кузнецова natalia@example.com CALL increase mechanic rank();



• Сколько автомобилей отремонтировал каждый механик за истекший квартал.

```
CREATE OR REPLACE FUNCTION count repair cars()
RETURNS TABLE (
  mechanic id bigint,
  mechanic name character(50),
  repaired count cars bigint
) AS $$
BEGIN
  RETURN QUERY
  SELECT
    e."ID employee" AS mechanic id,
    e."Full name" AS mechanic name,
    COUNT(DISTINCT c."ID auto") AS repaired count cars
  FROM
    public."Employee" e
  JOIN public."Contract" c ON c."ID employee" = e."ID employee"
  JOIN public."Job_position" jb ON jb."ID job position" = e."ID job position"
  WHERE
    jb."Name" = 'Мастер-механик'
    AND c."Actual_date_end_of_repair" >= CURRENT_DATE - INTERVAL '3 months'
  GROUP BY
    e."ID employee";
END;
$$ LANGUAGE plpgsql;
Вызов функции: select * from count repair cars();
```

```
AutorepairShop=# CREATE OR REPLACE FUNCTION count_repair_cars()
AutorepairShop-# RETURNS TABLE (
                      mechanic_id bigint,
AutorepairShop(#
AutorepairShop(#
                      mechanic name character(50),
AutorepairShop(#
                     repaired_count_cars bigint
AutorepairShop(# ) AS $$
AutorepairShop$# BEGIN
                      RETURN QUERY
AutorepairShop$#
AutorepairShop$#
                      SELECT
                          e."ID_employee" AS mechanic_id,
e."Full_name" AS mechanic_name,
AutorepairShop$#
AutorepairShop$#
AutorepairShop$#
                          COUNT(DISTINCT c."ID_auto") AS repaired_count_cars
AutorepairShop$#
AutorepairShop$#
                         public."Employee" e
                      JOIN public. "Contract" c ON c. "ID_employee" = e. "ID_employee"

JOIN public. "Job_position" jb ON jb. "ID_job_position" = e. "ID_job_position"
AutorepairShop$#
AutorepairShop$#
AutorepairShop$#
AutorepairShop$#
                           jb."Name" = 'Мастер-механик'
                           AND c."Actual_date_end_of_repair" >= CURRENT_DATE - INTERVAL '3 months'
AutorepairShop$#
AutorepairShop$#
                      GROUP BY
AutorepairShop$#
                           e."ID employee";
AutorepairShop$# END;
AutorepairShop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
AutorepairShop=# select * from count_repair_cars();
                                    mechanic_name
                                                                         | repaired_count_cars
mechanic_id |
           2 Иван Петров
           3 | Самолимов Иван
4 | Алексей Иванов
                                                                                              73
                                                                                              68
              Ольга Козлова
                                                                                              56
              Анна Павлова
               Петр Михайлов
                                                                                              60
               Евгения Никитина
                                                                                              47
              Наталья Кузнецова
8 строк)
```

#### 2.2. Создать авторский триггер по варианту индивидуального задания.

При вставке детали в таблицу Details\_from\_client, изменяем стоимость ремонта авто в таблице Contract на сумму, равную стоимостью деталей.

```
CREATE OR REPLACE FUNCTION update_total_payment_on_detail_insert()
```

**RETURNS TRIGGER AS \$\$** 

**DECLARE** 

total\_detail\_price INTEGER;

**BEGIN** 

SELECT dfc."Amount\_of\_detail" \* d."Price"

INTO total\_detail\_price

```
FROM public."Detail" d
```

JOIN public. "Details\_from\_client" dfc ON d. "ID\_detail" = dfc. "ID\_detail"

WHERE dfc."ID\_detail" = NEW."ID\_detail";

UPDATE public."Contract"

SET "Total\_payment" = "Total\_payment" - total\_detail\_price

WHERE "ID\_contract" = (

SELECT "ID\_contract"

FROM public."Distribution\_of\_work"

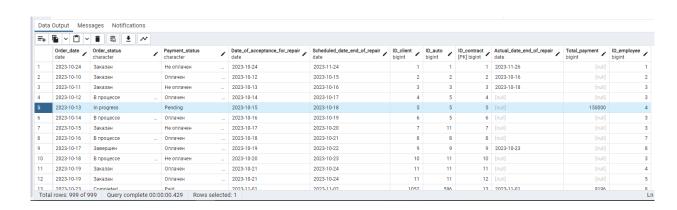
WHERE "ID\_destribution\_of\_work" = NEW."ID\_distribution"

)AND "Order\_status" = 'In progress' AND "Payment\_status" = 'Pending';

RETURN NEW;

END;

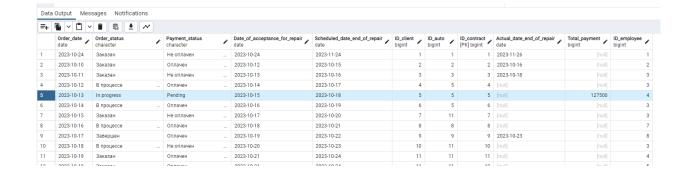
\$\$ LANGUAGE plpgsql;



CREATE TRIGGER update\_total\_payment\_on\_detail\_insert\_trigger

AFTER INSERT ON public."Details\_from\_client"

FOR EACH ROW EXECUTE FUNCTION update total payment on detail insert();



#### Вывод:

В результате выполнения лабораторной работы были закреплены навыки создания и использования процедур, функций и триггеров в PostgreSQL. Освоены принципы их работы, а также применение в конкретных сценариях, связанных с автомастерской и управлением данными в данной предметной области.