

**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»**

**Обучающийся Камалов Руслан Олегович
Факультет прикладной информатики
Группа К3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии
2023 Преподаватель Говорова Марина Михайловна**

**Санкт-Петербург
2024/2025**

1. Цель работы:

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

2.Практическое задание:

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР)
2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
 2. Создать триггеры для индивидуальной БД согласно варианту:
Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

3. Выполнение:

1. Вывести сведения о заказах заданного официанта на заданную дату:

```
restaurant=# create or replace procedure get_waiter_orders_by_date
    IN p_employee_id INTEGER,
    IN p_order_date DATE,
        INOUT result_cursor REFCURSOR DEFAULT 'orders_cursor'
)
language plpgsql
as $$

begin
    open result_cursor for
    select o.order_id, o.table_id, o.order_date, o.comments
    from restaurant.orders o
    join restaurant."tables" t on o.table_id = t.table_id
    where t.employee_id = p_employee_id
    and date(o.order_date) = p_order_date;
end;
$$;
CREATE PROCEDURE
```

```
restaurant=# begin;
call get_waiter_orders_by_date(1, '2025-05-19');
fetch all from orders_cursor;
commit;
BEGIN
result_cursor
-----
orders_cursor
(1 строка)

order_id | table_id |          order_date | comments
-----+-----+-----+-----+
      1 |       1 | 2025-05-19 12:30:00 | Острое не добавлять
      3 |       3 | 2025-05-19 19:00:00 |
      4 |       3 | 2025-05-19 15:58:08.209304 |
      5 |       3 | 2025-05-19 16:02:34.555877 | Без лактозы
      6 |       3 | 2025-05-19 16:02:38.086504 | Без лактозы
      7 |       3 | 2025-05-19 16:18:09.576969 | Без лактозы
      8 |       3 | 2025-05-19 16:18:42.398373 | Без лактозы
      9 |       3 | 2025-05-19 16:21:30.622457 | Без лактозы
     10 |       3 | 2025-05-19 16:21:40.469818 | Без лактозы
     11 |       3 | 2025-05-19 16:24:58.619243 | Без лактозы
     12 |       3 | 2025-05-19 16:25:10.405032 | Без лактозы
     13 |       3 | 2025-05-19 16:25:56.557553 | Без лактозы
     14 |       3 | 2025-05-19 16:26:54.560145 | Без лактозы
     15 |       3 | 2025-05-19 16:27:00.903631 | Без лактозы
     16 |       3 | 2025-05-19 16:32:04.942628 | Без лактозы
     17 |       3 | 2025-05-19 16:34:14.50664 | Без лактозы
     18 |       3 | 2025-05-19 16:34:20.154957 | Без лактозы
(17 строк)

COMMIT
restaurant=#
restaurant=#
restaurant=#
```

2. Выполнить расчет стоимости заданного заказа:

```
restaurant=# create or replace procedure calculate_order_cost
    p_order_id INT
)
as $$

declare
    total_cost DECIMAL(10, 2);
begin
    select sum(od.quantity * (
        select sum(di.quantity * i.price_per_unit)
        from restaurant.dish_ingredients di
        join restaurant.ingredients i on di.ingredient_id = i.ingredient_id
        where di.dish_id = od.dish_id
    )) into total_cost
    from restaurant.order_details od
    where od.order_id = p_order_id;

    if total_cost is null then
        total_cost := 0;
    end if;
    raise notice 'Стоимость заказа: %', total_cost;
end;
$$ language plpgsql;
CREATE PROCEDURE
```

3. Повышение оклада заданного сотрудника на 30 % при повышении его категории:

- #### 4. Создать необходимые триггеры:


```

restaurant=# 
restaurant=# 
restaurant=# 
restaurant=# 
restaurant=# 
restaurant=# create table logs (
    text TEXT NOT NULL,
    added TIMESTAMP WITHOUT TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE
restaurant=# create or replace function add_to_log() returns trigger as $$ 
declare
    mstr varchar(30);
    astr varchar(100);
    retstr varchar(254);
begin
    if TG_OP = 'INSERT' then
        astr = NEW.employee_id;
        mstr := 'Add new employee ';
        retstr := mstr||astr;
        insert into logs(text,added) values (retstr,NOW());
        return new;
    elsif TG_OP = 'UPDATE' then
        astr = NEW.employee_id;
        mstr := 'Update employee ';
        retstr := mstr||astr;
        insert into logs(text,added) values (retstr,NOW());
        return new;
    elsif TG_OP = 'DELETE' then
        astr = OLD.employee_id;
        mstr := 'Remove employee ';
        retstr := mstr || astr;
        insert into logs(text,added) values (retstr,NOW());
        return old;
    end if;
end;
$$ language plpgsql;
CREATE FUNCTION
restaurant=# drop trigger if exists t_employee ON restaurant.employees;
DROP TRIGGER
restaurant=# create trigger t_employee after insert or update or delete on restaurant.employees for each row execute procedure add_to_log();
CREATE TRIGGER
restaurant=# insert into restaurant.employees (employee_id, position_id, full_name, passport_data, category, salary)
values (777, 1, 'Камалов Руслан Олегович', '1111 222333', 'Junior', 50000.00);
INSERT 0 1
restaurant=# select * from logs;
   text    |      added
-----+-----
 Add new employee 777 | 2025-06-15 14:42:05.167322
(1 строка)

restaurant=#

```

5. Выводы:

В ходе работы были успешно освоены практические навыки создания и использования хранимых процедур, функций и триггеров в PostgreSQL. Разработаны процедуры для автоматизации часто выполняемых операций, пользовательские функции для сложных вычислений и триггеры для поддержания целостности данных при изменениях. Все объекты протестированы на корректность выполнения. В результате я научился эффективно применять встроенный функционал PostgreSQL для оптимизации работы с базой данных, что позволяет упростить логику приложений и повысить надежность хранения информации.