

**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: Цыпандин А.П.

Факультет: ИКТ

Группа: K3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

## Оглавление

ЦЕЛЬ РАБОТЫ.....	3
ПРАКТИЧЕСКОЕ ЗАДАНИЕ .....	3
ВАРИАНТ 14. БД «СЛУЖБА ЗАКАЗА ТАКСИ».....	3
ВЫПОЛНЕНИЕ .....	3
Схема логической модели данных.....	5
Процедуры, функции и триггеры.....	5
ВЫВОД .....	7

## Цель работы

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

## Практическое задание

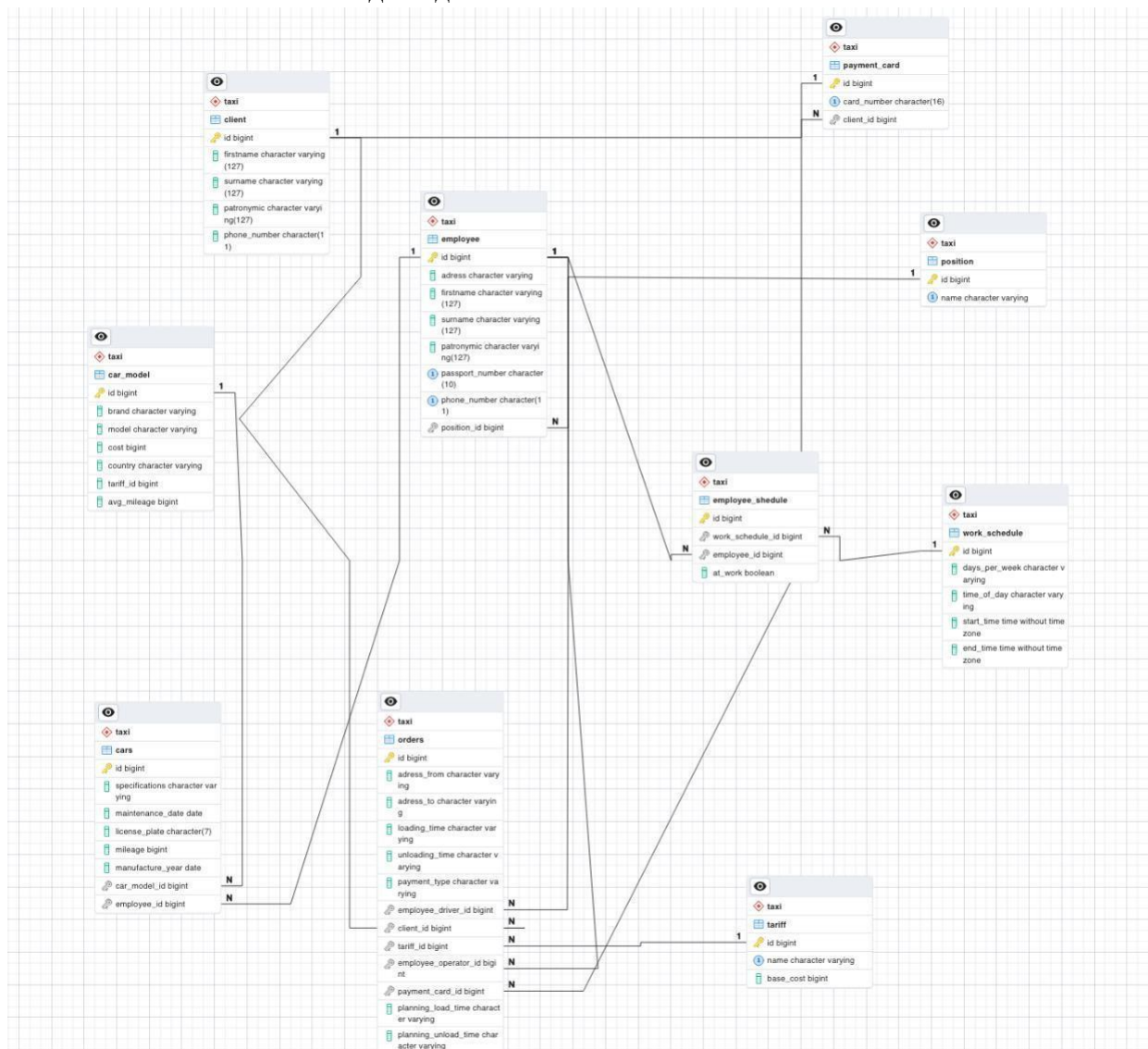
### Вариант 1 (max - 6 баллов)

1. Создать процедуры/функции 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 работы

Название БД: Сервис для заказа такси.

Схема логической модели данных:



## Скрипты создания процедур:

- 1) Для вывода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале.

```
CREATE OR REPLACE FUNCTION get_clients_in_time_interval(start_time
TIMESTAMP, end_time TIMESTAMP)
RETURNS TABLE (
    client_id BIGINT,
    firstname VARCHAR,
    surname VARCHAR,
    phone_number CHAR
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        c.id,
        c.firstname,
        c.surname,
        c.phone_number
    FROM taxi.client c
    JOIN taxi.orders o ON o.client_id = c.id
    WHERE o.planning_load_time BETWEEN start_time AND end_time;
END;
$$ LANGUAGE plpgsql;
```

```
(Taxi_service=# CREATE OR REPLACE FUNCTION get_clients_in_time_interval(start_time TIMEST
AMP)
(Taxi_service=# RETURNS TABLE (
(Taxi_service=# client_id BIGINT,
(Taxi_service=# firstname VARCHAR,
(Taxi_service=# surname VARCHAR,
(Taxi_service=# phone_number CHAR
(Taxi_service=# ) AS $$
(Taxi_service$$ BEGIN
(Taxi_service$$ RETURN QUERY
(Taxi_service$$ SELECT
(Taxi_service$$ c.id,
(Taxi_service$$ c.firstname,
(Taxi_service$$ c.surname,
(Taxi_service$$ c.phone_number
(Taxi_service$$ FROM taxi.client c
(Taxi_service$$ JOIN taxi.orders o ON o.client_id = c.id
(Taxi_service$$ WHERE o.planning_load_time BETWEEN start_time AND end_time;
(Taxi_service$$ END;
(Taxi_service$$ $$ LANGUAGE plpgsql;
CREATE FUNCTION
(Taxi_service=# SELECT * FROM get_clients_in_time_interval('2022-01-01', '2024-01-01');
 client_id | firstname | surname | phone_number
-----+-----+-----+-----
      153 | Иван     | Иванов  | 89124295544
        1 | Богданов | Аверкий | 70545283392
        2 | Силина   | Октябрина | 79858127980
        4 | Марфа    | Аркадьевна | 79123857961
        4 | Марфа    | Аркадьевна | 79123857961
        4 | Марфа    | Аркадьевна | 79123857961
        7 | Ия       | Викторовна | 72312708491
```

- 2) Вывести сведения о том, куда был доставлен пассажир по заданному номеру телефона пассажира.

```
CREATE OR REPLACE FUNCTION get_order_by_mobile_phone(phone VARCHAR)
RETURNS TABLE (
    adress_to VARCHAR,
    loading_time VARCHAR,
    unloading_time VARCHAR,
    employee_driver_id BIGINT,
    client_id BIGINT
) AS $$
DECLARE client_id_d INT;
BEGIN
    SELECT c.id INTO client_id_d
    FROM taxi.client c
    WHERE phone_number = phone;
    IF client_id_d IS NOT NULL THEN
        RETURN QUERY
        SELECT
            o.adress_to AS address_to,
            o.loading_time AS loading_time,
            o.unloading_time AS unloading_time,
            o.employee_driver_id AS driver_id,
            o.client_id AS client_id
        FROM taxi.orders o
        WHERE o.client_id = client_id_d;
    ELSE
        RAISE NOTICE 'Passenger with Phone Number % not found.',
phone;
    END IF;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM get_order_by_mobile_phone('89124295544');
```

```

[Taxi_service=# CREATE OR REPLACE FUNCTION get_order_by_mobile_phone(phone VARCHAR)
[Taxi_service=# RETURNS TABLE(
[Taxi_service(# address_to VARCHAR,
[Taxi_service(# loading_time VARCHAR,
[Taxi_service(# unloading_time VARCHAR,
[Taxi_service(# employee_driver_id BIGINT,
[Taxi_service(# client_id BIGINT
[Taxi_service(# ) AS $$
[Taxi_service$# DECLARE client_id_d INT;
[Taxi_service$# BEGIN
[Taxi_service$# SELECT c.id INTO client_id_d
[Taxi_service$# FROM taxi.client c
[Taxi_service$# WHERE phone_number = phone;
[Taxi_service$# IF client_id_d IS NOT NULL THEN
[Taxi_service$# RETURN QUERY
[Taxi_service$# SELECT
[Taxi_service$# o.address_to AS address_to,
[Taxi_service$# o.loading_time AS loading_time,
[Taxi_service$# o.unloading_time AS unloading_time,
[Taxi_service$# o.employee_driver_id AS driver_id,
[Taxi_service$# o.client_id AS client_id
[Taxi_service$# FROM taxi.orders o
[Taxi_service$# WHERE o.client_id = client_id_d;
[Taxi_service$# ELSE
[Taxi_service$# RAISE NOTICE 'Passenger with Phone Number % not found', phone;
[Taxi_service$# END IF;
[Taxi_service$# END;
[Taxi_service$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
[Taxi_service=# SELECT * FROM get_order_by_mobile_phone('89124295544');

```

address_to	loading_time	unloading_time	employee_driver_id	client_id
Кириллова 19 д.2	2023-03-22 09:03:23+	2023-03-22 10:05:52+ +	13	153

3) Для вычисления суммарного дохода таксопарка за истекший месяц.

```

CREATE OR REPLACE FUNCTION get_last_month_income()
RETURNS TABLE(
    month_income INT
) AS $$
BEGIN
    RETURN QUERY
    SELECT ROUND((SUM(EXTRACT(EPOCH FROM (ews.end_time - ews.start_time)) /
3600 * p.salary_per_hour))::INT
    FROM taxi.employee_working_shift ews
    JOIN taxi.employee e ON ews.driver_id = e.id
    JOIN taxi.position p ON e.position_id = p.id
    WHERE ews.start_time > CURRENT_DATE - INTERVAL '1 month';
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_last_month_income();

```

```

[Taxi_service=# CREATE OR REPLACE FUNCTION get_last_month_income()
Taxi_service=# RETURNS TABLE(
Taxi_service=# month_income INT
Taxi_service=# ) AS $$
Taxi_service$$ BEGIN
Taxi_service$$ RETURN QUERY
Taxi_service$$ SELECT ROUND(SUM(EXTRACT(EPOCH FROM (ews.end_time - ews.start_time)) / 3600 * p.salary_per_hour))::INT
Taxi_service$$ FROM taxi.employee_working_shift ews
Taxi_service$$ JOIN taxi.employee e ON ews.driver_id = e.id
Taxi_service$$ JOIN taxi.position p ON e.position_id = p.id
Taxi_service$$ WHERE ews.start_time > CURRENT_DATE - INTERVAL '1 month';
Taxi_service$$ END;
Taxi_service$$ $$ LANGUAGE plpgsql;
CREATE FUNCTION
[Taxi_service=# SELECT * FROM get_last_month_income();
 month_income
-----
          32000
(1 row)

```

Скрипты создания триггера на логирование данных:

```
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.id;
```

```
        mstr := 'Add new user ';
```

```
        retstr := mstr||astr;
```

```
        INSERT INTO logs(log,added) values (retstr,NOW());
```

```
        RETURN NEW;
```

```
    ELSIF TG_OP = 'UPDATE' THEN
```

```
        astr = NEW.id;
```

```
        mstr := 'Update user ';
```

```
        retstr := mstr||astr;
```

```
        INSERT INTO logs(log,added) values (retstr,NOW());
```

```
        RETURN NEW;
```

```
    ELSIF TG_OP = 'DELETE' THEN
```

```
        astr = OLD.id;
```

```
        mstr := 'Remove user ';
```

```
        retstr := mstr || astr;
```

```
        INSERT INTO logs(log,added) values (retstr,NOW());
```




```
        RETURN OLD;
```

```
    END IF;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER t_client AFTER INSERT OR UPDATE OR DELETE ON taxi.client
FOR EACH ROW EXECUTE PROCEDURE add_to_log();
```

	<b>id</b> [PK] integer 	<b>log</b> text 	<b>added</b> timestamp without time zone 
1	1	Add new user 154	2024-01-25 13:44:56.594936



## **Вывод**

В результате выполнения лабораторной работы мы овладели навыками работы с процедурами, функциями и триггерами в PostgreSQL. Создание хранимых процедур позволяет выполнять сложные операции над данными, функции обеспечивают возможность возврата результатов вычислений, а триггеры автоматизируют обновления в базе данных.