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: К3239

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы

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

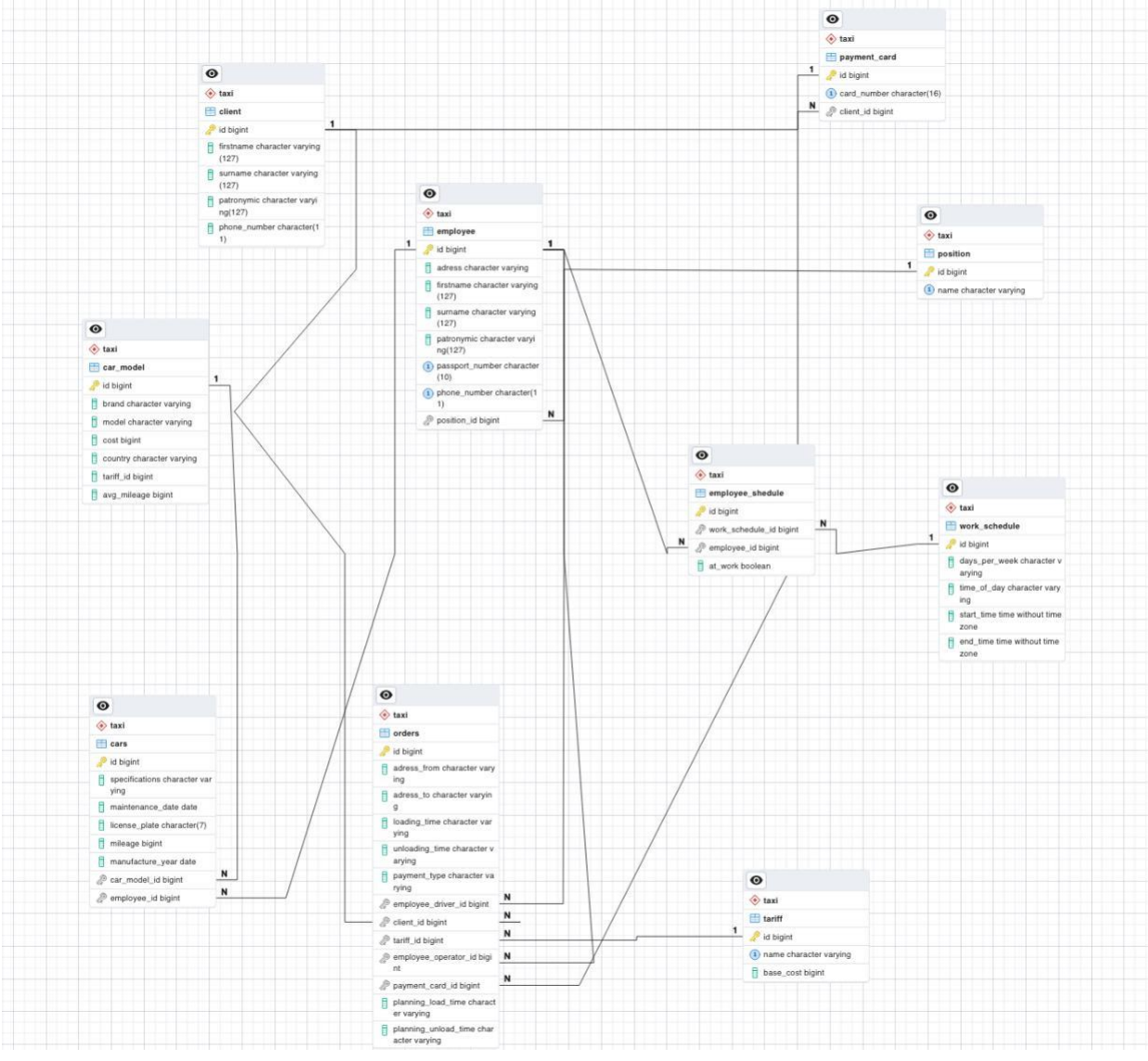
Практическое задание

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 PROCEDURE get_clients_in_time_interval(start_time
TIMESTAMP, end_time TIMESTAMP)
LANGUAGE plpgsql
AS $$
DECLARE
    client_info RECORD;
BEGIN
    FOR client_info IN
        SELECT
            c.id AS client_id,
            c.firstname AS firstname,
            c.surname AS surname,
            c.phone_number AS phone_number
        FROM taxi.client c
        JOIN taxi.orders o ON o.client_id = c.id
        WHERE o.planning_load_time BETWEEN start_time AND end_time
    LOOP
        RAISE NOTICE 'Client ID: %, Firstname: %, Surname: %, Phone Number:
%',
                     client_info.client_id,
                     client_info.firstname,
                     client_info.surname,
                     client_info.phone_number;
    END LOOP;
END;
$$;
```

```
[Taxi_service=# CREATE OR REPLACE PROCEDURE get_clients_in_time_interval(start_time TIMESTAMP, end_time TIMESTAMP)
[Taxi_service-# LANGUAGE plpgsql
[Taxi_service-# AS $$
[Taxi_service$# DECLARE client_info RECORD;
[Taxi_service$# BEGIN
[Taxi_service$# FOR client_info IN
[Taxi_service$# SELECT
[Taxi_service$# c.id AS client_id,
[Taxi_service$# c.firstname AS firstname,
[Taxi_service$# c.surname AS surname,
[Taxi_service$# c.phone_number AS phone_number
[Taxi_service$# FROM taxi.client c
[Taxi_service$# JOIN taxi.orders o ON o.client_id = c.id
[Taxi_service$# WHERE o.planning_load_time BETWEEN start_time AND end_time
[Taxi_service$# LOOP
[Taxi_service$# RAISE NOTICE 'Client id: %, firstname: %, surname: %, phone number: %',
[Taxi_service$# client_info.client_id,
[Taxi_service$# client_info.firstname,
[Taxi_service$# client_info.surname,
[Taxi_service$# client_info.phone_number;
[Taxi_service$# END LOOP;
[Taxi_service$# END;
[Taxi_service$# $$;
CREATE PROCEDURE
```

Вызов:

```
[Taxi_service=# CALL get_clients_in_time_interval('2022-01-11 00:01:01', '2022-01-12 23:59:59');
NOTICE:  Client id: 1, firstname: Богданов, surname: Аверкий, phone number: 70545283392
NOTICE:  Client id: 2, firstname: Силина, surname: Октябрина, phone number: 79858127980
NOTICE:  Client id: 4, firstname: Марфа, surname: Аркадьевна, phone number: 79123857961
CALL
```

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

```
CREATE OR REPLACE PROCEDURE get_order_by_mobile_phone(phone VARCHAR)
LANGUAGE plpgsql
AS $$
DECLARE
    order_info RECORD;
    client_id_d INT;
BEGIN
    SELECT id INTO client_id_d
    FROM taxi.client
    WHERE phone_number = phone;

    IF client_id_d IS NOT NULL THEN
        SELECT
            o.address_to AS address_to,
            o.loading_time AS loading_time,
            o.unloading_time AS unloading_time,
            o.employee_driver_id AS driver_id
        INTO order_info
        FROM taxi.orders o
        WHERE o.client_id = client_id_d;

        RAISE NOTICE 'Arrival Address: %, Loading Time: %, Unloading Time:
%, Driver ID: %',
                     order_info.address_to,
                     order_info.loading_time,
                     order_info.unloading_time,
                     order_info.driver_id;
    ELSE
        RAISE NOTICE 'Passenger with Phone Number % not found.', phone;
    END IF;
END;
$$;
```

```
Taxi_service=# CREATE OR REPLACE PROCEDURE get_order_by_mobile_phone(phone VARCHAR)
Taxi_service-# LANGUAGE plpgsql
Taxi_service-# AS $$
Taxi_service$# DECLARE
Taxi_service$# order_info RECORD;
Taxi_service$# client_id_d INT;
Taxi_service$# BEGIN
Taxi_service$# SELECT id INTO client_id_d
Taxi_service$# FROM taxi.client
Taxi_service$# WHERE phone_number = phone;
Taxi_service$# IF client_id_d IS NOT NULL THEN
Taxi_service$# SELECT
Taxi_service$# o.adress_to AS adress_to,
Taxi_service$# o.loading_time AS loading_time,
Taxi_service$# o.unloading_time AS unloading_time,
Taxi_service$# o.employee_driver_id AS driver_id
Taxi_service$# INTO order_info
Taxi_service$# FROM taxi.orders o
Taxi_service$# WHERE o.client_id = client_id_d;
Taxi_service$# RAISE NOTICE 'arrival adress: %, loading time: %, unloading time: %, driver id: %',
Taxi_service$# order_info.adress_to,
Taxi_service$# order_info.loading_time,
Taxi_service$# order_info.unloading_time,
Taxi_service$# order_info.driver_id;
Taxi_service$# ELSE
Taxi_service$# RAISE NOTICE 'Passenger with Phone Number % not found.', phone_number;
Taxi_service$# END IF;
Taxi_service$# END;
Taxi_service$# $$;
CREATE PROCEDURE
```

Вызов:

```
Taxi_service=# CALL get_order_by_mobile_phone('79123857961');
NOTICE:  arrival adress: бул. Герцена, д. 9 к. 3/5, loading time: 2023-03-11 13:29:24, unloading time: 2023-03-11 13:44:42, driver id: 30
CALL
```

3) Для вычисления суммарного дохода таксопарка за истекший месяц.

```
CREATE OR REPLACE PROCEDURE get_last_month_income()
LANGUAGE plpgsql
DECLARE
    month_income INT;
AS $$
BEGIN
    SELECT SUM(t.base_cost * 8)
    INTO motnh_income
    FROM taxi.tariff t, taxi.orders o
    WHERE o.planning_load_time BETWEEN '2023-12-01' AND '2023-12-31';

    RAISE NOTICE 'Summary: %', month_income;
END;
$$;
```

```
Taxi_service=# CREATE OR REPLACE PROCEDURE get_last_month_income()
Taxi_service-# LANGUAGE plpgsql
Taxi_service-# AS $$
Taxi_service$# DECLARE month_income INT;
Taxi_service$# BEGIN
Taxi_service$# SELECT SUM(t.base_cost * 8)
Taxi_service$# INTO month_income
Taxi_service$# FROM taxi.tariff t, taxi.orders o
Taxi_service$# WHERE o.planning_load_time BETWEEN CURRENT_DATE - INTERVAL '1 month' AND CURRENT_DATE;
Taxi_service$# RAISE NOTICE 'Summary: %', month_income;
Taxi_service$# END;
Taxi_service$# $$;
CREATE PROCEDURE
```

Вызов:

```
[Taxi_service=# CALL get_last_month_income();
NOTICE:  Summary: 56000
CALL
```

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();
```

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.