Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе «процедуры, функции, триггеры в PostgreSQL»

по дисциплине «**Проектирование и реализация баз данных**»

Автор: Коротин А.М.

Факультет: ИКТ

Группа: К32391

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

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).

Практическое задание:

Вариант 1

1. 2.Создать процедуры/функции согласно индивидуальному заданию и (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.

Вариант 2

1. Создать процедуры/функции согласно индивидуальному заданию и (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.

Вариант 14

Задание 4. Создать хранимые процедуры:

- Для вывода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале.
- Вывести сведения о том, куда был доставлен пассажир по заданному номеру телефона пассажира.
- Для вычисления суммарного дохода таксопарка за истекший месяц.

Задание 5. Создать необходимые триггеры.

Ход работы:

1. **Функция для вывода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале**

```
CREATE OR REPLACE FUNCTION get_user_info(interval_start TIMESTAMP, interval_end
TIMESTAMP)

RETURNS TABLE(

  user_id INT,

  first_name VARCHAR(50),

  last_name VARCHAR(50),

  phone_number VARCHAR(12)

)

 as $$ begin return query

SELECT * FROM usr u

    WHERE u.id IN (SELECT DISTINCT o.usr_id FROM ordr o

                                    WHERE o.time_start_fact >= interval_start AND

                                        o.time_end <= interval_end);

END;

$$ LANGUAGE plpgsql;
```

Результат выполнения функции:

```
taxi=# select * from get_user_info('1.1.1999', '1.1.2030');
 user_id | first_name | last_name | phone_number
---------+------------+-----------+--------------
       1 | Вася       | Пупкин    | +79161234567
       2 | 123421     | 412421    | +79007162354
       3 | Alexey     | 4124      | +79007654321
       4 | unique     | user      | +79111111111
(4 rows)
```

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

```
CREATE OR REPLACE FUNCTION get_destination_addr_by_phone_number(phone
VARCHAR(12))

RETURNS SETOF VARCHAR(200)

 as $$ begin return query

SELECT DISTINCT o.address_to FROM ordr o

    JOIN usr u ON u.id = o.usr_id
```

```
        WHERE u.phone_number = phone;
END;
$$ LANGUAGE plpgsql;
```

Результат выполнения функции:

```
taxi=# SELECT * FROM get_destination_addr_by_phone_number('+79161234567');
 get_destination_addr_by_phone_number
--------------------------------------
 251521
 42141241
 Addr to
 Example to
 Hello to
 Домой в кроватку
 ИТМО Гривцова
 Москва
 МСГ
 Ооооооочень длинный адрес
 Придумывать адреса - сложно
(11 rows)
```

3. Для вычисления суммарного дохода таксопарка за истекший месяц.

В связи со спецификой генерации данных доход за прошедший месяц будет = 0. Вместо этого приведу функцию, считающую общий доход таксопарка за все время.

```
CREATE OR REPLACE FUNCTION get_total_income()
RETURNS NUMERIC(10, 2)
as $$
DECLARE function_result NUMERIC(10, 2);
BEGIN
    function_result =
    (SELECT SUM(o.distance_km * r.price_per_km) FROM ordr o
        JOIN car c1 ON o.car_id = c1.id
            JOIN car_type c2 ON c1.type_id = c2.id
                JOIN rate r ON c2.rate_id = r.id);
    RETURN function_result;
END;
$$ LANGUAGE plpgsql;
```

Результат выполнения функции:

```
taxi=# SELECT * FROM get_total_income();
 get_total_income
------------------
           518.00
(1 row)
```

Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL

Создание таблицы логов

```
CREATE TABLE log_table(
        id SERIAL PRIMARY KEY,
        taget TEXT,
        operation TEXT,
        target_record_id INT,
        date_time TIMESTAMP
);
```

Создание и настройка триггерной функции

```
CREATE OR REPLACE FUNCTION log_func()
RETURNS TRIGGER AS $$
BEGIN
 INSERT INTO log_table VALUES (TG_TABLE_NAME, TG_OP, NEW.id, NOW())
 RETURN NEW;
END;

CREATE TRIGGER log_changes_passenger_trigger AFTER
INSERT OR UPDATE OR DELETE
ON usr
FOR EACH ROW EXECUTE FUNCTION log_func();
```

Проверка работы триггера

Вставка

```
taxi=# INSERT INTO usr (first_name, last_name, phone_number) VALUES ('trigger_test', 'last_name', '+79167957092') RETURNING id;
 id
----
 10
(1 row)

INSERT 0 1
taxi=# SELECT * FROM log_table;
 id | taget | operation | target_record_id |         date_time
----+-------+-----------+------------------+----------------------------
  3 | usr   | INSERT    |               10 | 2023-04-26 17:54:34.407101
(1 row)
```

Обновление

```
taxi=# UPDATE usr SET first_name = 'edited_name' WHERE id=10;
UPDATE 1
taxi=# SELECT * FROM log_table;
 id | taget | operation | target_record_id |         date_time
----+-------+-----------+------------------+----------------------------
  3 | usr   | INSERT    |               10 | 2023-04-26 17:54:34.407101
  4 | usr   | UPDATE    |               10 | 2023-04-26 17:55:48.446195
(2 rows)
```

Удаление

```
taxi=# DELETE FROM usr WHERE id=10;
DELETE 1
taxi=# SELECT * FROM log_table;
 id | taget | operation | target_record_id |         date_time
----+-------+-----------+------------------+----------------------------
  3 | usr   | INSERT    |               10 | 2023-04-26 17:54:34.407101
  4 | usr   | UPDATE    |               10 | 2023-04-26 17:55:48.446195
  5 | usr   | DELETE    |                 | 2023-04-26 17:56:34.890761
(3 rows)
```

Вывод

В ходе данной лабораторной работы мной были изучены процедуры и функции в SQL, а также я познакомился с использованием триггеров.

В последнем задании я создал универсальный триггер логирования, который можно использовать с любой с любой таблицей вне зависимости от её структуры (единственное ограничение название столбца “id”).