ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий

Дисциплина:
«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

Выполнил:
студент группы K32392
Байков Иван

_____
(подпись)

Проверил(а):
Говорова Марина Михайловна

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург
2023 г.

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

Практическое задание:

Вариант 2

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. 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.
3. Создать авторский триггер по варианту индивидуального задания.

Выполнение:

Индивидуальное задание БД «Служба заказа такси»

Процедуры\функции:

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

Триггеры:

1.

Процедуры\функции:

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

```
CREATE OR REPLACE FUNCTION get_passenger_data(start_date DATE, end_date DATE)
RETURNS TABLE (phone_number VARCHAR, name VARCHAR)
AS $$
BEGIN
RETURN QUERY
SELECT d.phone_number, p.name
FROM drive d
JOIN passenger p ON d.phone_number = p.phone_number
WHERE d.date_time_drive_start >= start_date AND d.date_time_drive_finish <= end_date;
END;
$$ LANGUAGE plpgsql;
```

Рис. 1 – Функция №1

```
postgres=# CREATE OR REPLACE FUNCTION get_passenger_data(start_date DATE, end_date DATE)
postgres-#   RETURNS TABLE (phone_number integer, name VARCHAR)
postgres-# AS $$
postgres$# BEGIN
postgres$#   RETURN QUERY
postgres$#     SELECT d.phone_number, p.name
postgres$#     FROM drive d
postgres$#     JOIN passenger p ON d.phone_number = p.phone_number
postgres$#     WHERE d.date_time_drive_start >= start_date AND d.date_time_drive_finish <= end_date;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# SELECT * FROM get_passenger_data('2023-01-01', '2023-09-3');
 phone_number |      name
--------------+----------------
      1234567 | Иван Иванов
      1234567 | Иван Иванов
      9876543 | Мария Смирнова
      9876543 | Мария Смирнова
(4 rows)
```

Рис. 2 – Результат выполнения функции №1

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

```
CREATE OR REPLACE FUNCTION get_passenger_destination(phone_number integer)
RETURNS TABLE (destination character varying)
AS $$
BEGIN
RETURN QUERY
SELECT d."where"
FROM drive d
WHERE d.phone_number = get_passenger_destination.phone_number;
END;
$$ LANGUAGE plpgsql;
```

Рис. 3 – Функция №2

```
postgres=# CREATE OR REPLACE FUNCTION get_passenger_destination(phone_number integer)
postgres-#   RETURNS TABLE (destination character varying)
postgres-# AS $$
postgres$# BEGIN
postgres$#   RETURN QUERY
postgres$#     SELECT d."where"
postgres$#     FROM drive d
postgres$#     WHERE d.phone_number = get_passenger_destination.phone_number;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# SELECT * FROM get_passenger_destination('9876543');
 destination
-------------
 Рубинштейна
 Рубинштейна
(2 rows)

postgres=#
```

Рис. 4 – Результат выполнения функции №2

3) Для вычисления суммарного дохода таксопарка за истекший месяц.

```
CREATE OR REPLACE PROCEDURE calculate_total_revenue()
AS $$
DECLARE
start_date DATE;
end_date DATE;
total_revenue DECIMAL := 0;
BEGIN
start_date := date_trunc('month', CURRENT_DATE) - interval '1 month';
end_date := date_trunc('month', CURRENT_DATE) - interval '1 day';

SELECT SUM((d.distance * p.price) + d.waiting_time_penalty)
INTO total_revenue
FROM drive d
JOIN rate r ON d.id_rate = r.id_rate
JOIN price_per_kilometer p ON r.id_rate = p.id_rate
WHERE d.date_time_drive_start >= start_date
AND d.date_time_drive_start <= end_date;

RAISE NOTICE 'Total Revenue: $%', total_revenue;
END;
$$ LANGUAGE plpgsql;
```

Рис. 5 – Процедура №1

```
postgres=# CREATE OR REPLACE PROCEDURE calculate_total_revenue()
postgres-# AS $$
postgres$# DECLARE
postgres$#     start_date DATE;
postgres$#     end_date DATE;
postgres$#     total_revenue DECIMAL := 0;
postgres$# BEGIN
postgres$#     start_date := date_trunc('month', CURRENT_DATE) - interval '1 month';
postgres$#     end_date := date_trunc('month', CURRENT_DATE) - interval '1 day';
postgres$#
postgres$#     SELECT SUM((d.distance * p.price) + d.waiting_time_penalty)
postgres$#     INTO total_revenue
postgres$#     FROM drive d
postgres$#     JOIN rate r ON d.id_rate = r.id_rate
postgres$#     JOIN price_per_kilometer p ON r.id_rate = p.id_rate
postgres$#     WHERE d.date_time_drive_start >= start_date
postgres$#       AND d.date_time_drive_start <= end_date;
postgres$#
postgres$#     RAISE NOTICE 'Total Revenue: $%', total_revenue;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE PROCEDURE
postgres=# CALL calculate_total_revenue();
NOTICE:  Total Revenue: $32000
CALL
```

Рис. 6 – Результат выполнения процедуры №1

Модификация триггера из практической работы:

```
CREATE OR REPLACE FUNCTION fn_check_time_punch() returns trigger as $psql$
DECLARE prev_tp RECORD;
begin
SELECT punch_time,
is_out_punch INTO prev_tp
FROM time_punch
WHERE employee_id = new.employee_id
ORDER BY punch_time DESC
LIMIT 1;
IF prev_tp IS NULL THEN RETURN new;
END IF;
IF prev_tp.is_out_punch = new.is_out_punch
OR prev_tp.punch_time >= new.punch_time
OR new.punch_time > now() THEN RETURN NULL;
END IF;
RETURN new;
end;
$psql$ language plpgsql;
```

Рис. 7 – Триггер из практической работы

Триггеры:

1) Отменяет поездку если время ожидания в денежном эквиваленте превысило 30 у е, либо если в данный момент нет свободных водителей

```
CREATE OR REPLACE FUNCTION cancel_drive_trigger()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.waiting_time_penalty > 30 OR NOT EXISTS (SELECT 1 FROM unoccupied_drivers)
THEN
NEW.status = 2; -- Status 2 => canceled
NEW.date_time_drive_finish = CURRENT_DATE;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;



CREATE TRIGGER cancel_drive
BEFORE UPDATE ON drive
FOR EACH ROW
WHEN (OLD.* IS DISTINCT FROM NEW.*)
EXECUTE FUNCTION cancel_drive_trigger();
```

Рис. 8 – Триггер №1

```
postgres=# SELECT "id_drive", "phone_number", "date_time_drive_start", "date_time_drive_finish", "waiting_time_penalty", "status"
postgres-# FROM drive
postgres-# WHERE "id_drive" = '14214';
 id_drive | phone_number | date_time_drive_start | date_time_drive_finish | waiting_time_penalty | status
----------+--------------+-----------------------+------------------------+----------------------+--------
 14214    |      9876543 | 2023-06-07            | 2023-06-08             |                    0 |      1
(1 row)

postgres=# UPDATE drive
postgres-# SET "waiting_time_penalty" = 31
postgres-# WHERE "id_drive" = '14214';
UPDATE 1
postgres=# SELECT "id_drive", "phone_number", "date_time_drive_start", "date_time_drive_finish", "waiting_time_penalty", "status"
postgres-# FROM drive
postgres-# WHERE "id_drive" = '14214';
 id_drive | phone_number | date_time_drive_start | date_time_drive_finish | waiting_time_penalty | status
----------+--------------+-----------------------+------------------------+----------------------+--------
 14214    |      9876543 | 2023-06-07            | 2023-06-08             |                   31 |      2
(1 row)
```

Рис. 19 – Результат выполнения триггера №1

Выводы:

В процессе выполнения данной лабораторной работы удалось овладеть навыками написания и использования процедур, функций и триггеров в PSQL. Также для себя я рассмотрел использование возможностей функций и процедур как ЯП. И на данный момент для себя могу отметить, что несмотря на схожесть с различными ЯП, SQL не обладает достаточной гибкостью.