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

Отчет

По лабораторной работе №3

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

Вариант 10. БД «Автовокзал»

Автор: Ле Хоанг Чыонг

Факультет: ИКТ

Группа: К32392

Преподаватель: Говорова М. М.

Санкт-Петербург, 2023

1. Описание работы

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. 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.

2. Описание предметной области

Вариант 10. БД «Автовокзал»

Описание предметной области: С автовокзала ежедневно отправляется несколько междугородных/международных автобусных рейсов. Номер рейса определяется маршрутом и временем отправления. По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки. Автобусы курсируют по расписанию, но могут назначаться дополнительные рейсы на заданный период или определенные даты. Билеты могут продаваться предварительно, но не ранее чем за 10 суток. В билете указывается номер места в автобусе. На каждый рейс может продаваться не более 10 билетов без места, цена на которые снижается на 10%. Пунктами отправления и назначения, согласно билету, могут быть промежуточные остановки. Билеты могут продаваться в кассе автовокзала или онлайн. На каждый рейс формируется экипаж из двух водителей. БД должна содержать следующий минимальный набор сведений: Номер рейса. Номер водителя. Номер автобуса. Паспортные данные водителя. Пункт отправления. Пункт назначения. Промежуточные остановки. Дата отправления. Время отправления. Время в пути. Тип автобуса. Количество мест в автобусе. Страна. Производитель. Год выпуска. Номер билета. Номер места в автобусе (при наличии). Цена билета. ФИО пассажира. Паспортные данные пассажира.

3. Выполнение работы

Индивидуальное задание БД «Автовокзал»

Задание 4. Создать хранимые процедуры:

1. Продажи билета.
2. Возврата билета.
3. Добавления нового рейса.

Выполнение:

1. Продажи билета.

```
CREATE OR REPLACE PROCEDURE ticket_sales(
    IN p_passport VARCHAR(30),
    IN p_seat_number INTEGER,
    IN p_price INTEGER,
    IN p_boarding_point VARCHAR(255),
    IN p_alighting_point VARCHAR(255),
    IN p_sales_type VARCHAR(10),
    IN p_trip_id INTEGER)
LANGUAGE PLPGSQL
AS $$
BEGIN
    INSERT INTO public.tickets (seat_number, passport, price, boarding_point, status,
alighting_point, sales_type, trip_id)
    VALUES (p_seat_number, p_passport, p_price, p_boarding_point, 'paid',
p_alighting_point, p_sales_type, p_trip_id);

    UPDATE public.seats
    SET is_booked = true
    WHERE seat_number = p_seat_number;
END;
$$;
```

```
CALL ticket_sales('P123461',12,100000,'New York Central','Los Angeles
Central','online',4);
```

```
Lab_database=# SELECT * FROM tickets;
 ticket_id | seat_number | passport | price  |   boarding_point    |  status   |   alighting_point   | sales_type | trip_id
-----------+-------------+----------+--------+---------------------+-----------+---------------------+------------+---------
         6 |           5 | P123460  | 700000 | Los Angeles Central | paid      | New York Central    | online     |       6
         3 |           2 | P123457  | 550000 | New York Central    | paid      | Los Angeles Central | online     |       3
         4 |           3 | P123458  | 650000 | New York Central    | paid      | Los Angeles Central | online     |       4
         9 |           8 | P123463  | 650000 | Los Angeles Central | unpaid    | New York Central    | online     |       4
         7 |           6 | P123461  | 750000 | Los Angeles Central | unpaid    | New York Central    | online     |       7
         8 |           7 | P123462  | 750000 | Los Angeles Central | cancelled | New York Central    | online     |       7
         5 |           4 | P123459  |  50000 | New York Central    | paid      | Los Angeles Central | online     |       5
        10 |           9 | P123464  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        11 |          10 | P123465  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        15 |           6 | P123465  |  22000 | New York Central    | cancelled | Los Angeles Central | online     |       4
(10 rows)


Lab_database=# CALL ticket_sales('P123461',12,100000,'New York Central','Los Angeles Central','online',4);
CALL
Lab_database=# SELECT * FROM tickets;
 ticket_id | seat_number | passport | price  |   boarding_point    |  status   |   alighting_point   | sales_type | trip_id
-----------+-------------+----------+--------+---------------------+-----------+---------------------+------------+---------
         6 |           5 | P123460  | 700000 | Los Angeles Central | paid      | New York Central    | online     |       6
         3 |           2 | P123457  | 550000 | New York Central    | paid      | Los Angeles Central | online     |       3
         4 |           3 | P123458  | 650000 | New York Central    | paid      | Los Angeles Central | online     |       4
         9 |           8 | P123463  | 650000 | Los Angeles Central | unpaid    | New York Central    | online     |       4
         7 |           6 | P123461  | 750000 | Los Angeles Central | unpaid    | New York Central    | online     |       7
         8 |           7 | P123462  | 750000 | Los Angeles Central | cancelled | New York Central    | online     |       7
         5 |           4 | P123459  |  50000 | New York Central    | paid      | Los Angeles Central | online     |       5
        10 |           9 | P123464  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        11 |          10 | P123465  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        15 |           6 | P123465  |  22000 | New York Central    | cancelled | Los Angeles Central | online     |       4
        22 |          12 | P123461  | 100000 | New York Central    | paid      | Los Angeles Central | online     |       4
(11 rows)
```

Рис. 1 – Функция №1

2. Возврата билета.

```
CREATE OR REPLACE PROCEDURE ticket_refunds(
    IN p_ticket_id INTEGER
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE public.tickets
    SET status = 'cancelled'
    WHERE ticket_id = p_ticket_id;

    UPDATE public.seats
    SET is_booked = false
    WHERE seat_number IN (
        SELECT seat_number
        FROM public.tickets
        WHERE ticket_id = p_ticket_id
    );
END;
$$;
```

```
CALL ticket_refunds(15)
```

```
Lab_database=# SELECT * FROM tickets;
 ticket_id | seat_number | passport | price  |   boarding_point    |  status   |   alighting_point   | sales_type | trip_id
-----------+-------------+----------+--------+---------------------+-----------+---------------------+------------+---------
         6 |           5 | P123460  | 700000 | Los Angeles Central | paid      | New York Central    | online     |       6
         3 |           2 | P123457  | 550000 | New York Central    | paid      | Los Angeles Central | online     |       3
         4 |           3 | P123458  | 650000 | New York Central    | paid      | Los Angeles Central | online     |       4
         9 |           8 | P123463  | 650000 | Los Angeles Central | unpaid    | New York Central    | online     |       4
         7 |           6 | P123461  | 750000 | Los Angeles Central | unpaid    | New York Central    | online     |       7
         8 |           7 | P123462  | 750000 | Los Angeles Central | cancelled | New York Central    | online     |       7
         5 |           4 | P123459  |  50000 | New York Central    | paid      | Los Angeles Central | online     |       5
        10 |           9 | P123464  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        11 |          10 | P123465  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        15 |           6 | P123465  |  22000 | New York Central    | cancelled | Los Angeles Central | online     |       4
        22 |          12 | P123461  | 100000 | New York Central    | paid      | Los Angeles Central | online     |       4
(11 rows)


Lab_database=# CALL ticket_refunds(22);
CALL
Lab_database=# SELECT * FROM tickets;
 ticket_id | seat_number | passport | price  |   boarding_point    |  status   |   alighting_point   | sales_type | trip_id
-----------+-------------+----------+--------+---------------------+-----------+---------------------+------------+---------
         6 |           5 | P123460  | 700000 | Los Angeles Central | paid      | New York Central    | online     |       6
         3 |           2 | P123457  | 550000 | New York Central    | paid      | Los Angeles Central | online     |       3
         4 |           3 | P123458  | 650000 | New York Central    | paid      | Los Angeles Central | online     |       4
         9 |           8 | P123463  | 650000 | Los Angeles Central | unpaid    | New York Central    | online     |       4
         7 |           6 | P123461  | 750000 | Los Angeles Central | unpaid    | New York Central    | online     |       7
         8 |           7 | P123462  | 750000 | Los Angeles Central | cancelled | New York Central    | online     |       7
         5 |           4 | P123459  |  50000 | New York Central    | paid      | Los Angeles Central | online     |       5
        10 |           9 | P123464  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        11 |          10 | P123465  |  50000 | New York Central    | cancelled | Los Angeles Central | offline    |       5
        15 |           6 | P123465  |  22000 | New York Central    | cancelled | Los Angeles Central | online     |       4
        22 |          12 | P123461  | 100000 | New York Central    | cancelled | Los Angeles Central | online     |       4
(11 rows)
```

Рис. 2 – Функция №2

3. Добавления нового рейса.

```
CREATE OR REPLACE PROCEDURE add_trip(
    IN p_departure_point VARCHAR(255),
    IN p_arrival_point VARCHAR(255),
    IN p_actual_departure_time TIMESTAMP,
    IN p_actual_arrival_time TIMESTAMP,
    IN p_schedule_id INTEGER,
    IN p_bus_id INTEGER,
    IN p_status VARCHAR(20)
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO public.trips (departure_point, arrival_point, actual_departure_time,
actual_arrival_time, schedule_id, bus_id, status)
    VALUES (p_departure_point, p_arrival_point, p_actual_departure_time,
p_actual_arrival_time, p_schedule_id, p_bus_id, p_status);
END;
$$;
```

```
CALL add_trip('New York Central','Los Angeles Central','2023-05-05 08:00:00','2023-05-05
16:00:00',1,2,'Pending')
```

```
Lab_database=# SELECT * FROM trips;
 trip_id |   departure_point    |    arrival_point     | actual_departure_time | actual_arrival_time | schedule_id | bus_id |   status
---------+----------------------+----------------------+-----------------------+---------------------+-------------+--------+-------------
       5 | Chicago Central      | Houston Central      | 2023-05-07 10:15:00   | 2023-05-08 16:25:00 |           3 |      3 | Pending
       6 | Houston Central      | Phoenix Central      | 2023-05-10 08:25:00   | 2023-05-10 14:50:00 |           4 |      4 | Completed
       7 | Phoenix Central      | Philadelphia Central | 2023-05-12 08:30:00   | 2023-05-12 14:15:00 |           5 |      5 | In progress
       8 | New York Central     | Los Angeles Central  | 2023-06-01 08:00:00   | 2023-06-01 16:00:00 |           1 |      1 | Pending
       3 | New York Central     | Los Angeles Central  | 2023-05-01 08:00:00   | 2023-05-01 16:45:00 |           1 |      1 | Completed
       4 | Philadelphia Central | San Antonio Central  | 2023-05-03 08:00:00   | 2023-05-03 16:45:00 |           6 |      6 | Completed
      10 | New York             | Boston               | 2023-05-04 08:00:00   | 2023-05-04 16:45:00 |           1 |      1 | Pending
      11 | Boston               | Philadelphia         | 2023-05-04 09:00:00   | 2023-05-04 17:45:00 |           1 |      1 | Pending
      12 | New York             | Boston               | 2023-05-05 08:00:00   | 2023-05-05 12:00:00 |          11 |      1 | Pending
      13 | Boston               | Philadelphia         | 2023-05-05 09:30:00   | 2023-05-05 12:30:00 |          12 |      2 | Pending
      14 | New York             | Washington           | 2023-05-05 10:00:00   | 2023-05-05 15:00:00 |          13 |      3 | Pending
(11 rows)


Lab_database=# CALL add_trip('New York Central','Los Angeles Central','2023-05-05 08:00:00','2023-05-05 16:00:00',1,2,'Pending');
CALL
Lab_database=# SELECT * FROM trips;
 trip_id |   departure_point    |    arrival_point     | actual_departure_time | actual_arrival_time | schedule_id | bus_id |   status
---------+----------------------+----------------------+-----------------------+---------------------+-------------+--------+-------------
       5 | Chicago Central      | Houston Central      | 2023-05-07 10:15:00   | 2023-05-08 16:25:00 |           3 |      3 | Pending
       6 | Houston Central      | Phoenix Central      | 2023-05-10 08:25:00   | 2023-05-10 14:50:00 |           4 |      4 | Completed
       7 | Phoenix Central      | Philadelphia Central | 2023-05-12 08:30:00   | 2023-05-12 14:15:00 |           5 |      5 | In progress
       8 | New York Central     | Los Angeles Central  | 2023-06-01 08:00:00   | 2023-06-01 16:00:00 |           1 |      1 | Pending
       3 | New York Central     | Los Angeles Central  | 2023-05-01 08:00:00   | 2023-05-01 16:45:00 |           1 |      1 | Completed
       4 | Philadelphia Central | San Antonio Central  | 2023-05-03 08:00:00   | 2023-05-03 16:45:00 |           6 |      6 | Completed
      10 | New York             | Boston               | 2023-05-04 08:00:00   | 2023-05-04 16:45:00 |           1 |      1 | Pending
      11 | Boston               | Philadelphia         | 2023-05-04 09:00:00   | 2023-05-04 17:45:00 |           1 |      1 | Pending
      12 | New York             | Boston               | 2023-05-05 08:00:00   | 2023-05-05 12:00:00 |          11 |      1 | Pending
      13 | Boston               | Philadelphia         | 2023-05-05 09:30:00   | 2023-05-05 12:30:00 |          12 |      2 | Pending
      14 | New York             | Washington           | 2023-05-05 10:00:00   | 2023-05-05 15:00:00 |          13 |      3 | Pending
      17 | New York Central     | Los Angeles Central  | 2023-05-05 08:00:00   | 2023-05-05 16:00:00 |           1 |      2 | Pending
(12 rows)
```

Рис. 3 – Функция №3

Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5).

Задание 5. Создать необходимые триггеры.

Создание таблицы логов

```
CREATE TABLE IF NOT EXISTS log_table (
  id SERIAL PRIMARY KEY,
  table_name TEXT,
  operation TEXT,
  old_data TEXT,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Создание и настройка триггерной функции

```
CREATE OR REPLACE FUNCTION create_trigger_function() RETURNS TRIGGER AS $$
DECLARE
  table_name TEXT;
  operation TEXT;
  old_data TEXT;
BEGIN
  IF (TG_OP = 'DELETE') THEN
    old_data = OLD::text;
    operation = 'DELETE';
  ELSIF (TG_OP = 'UPDATE') THEN
    old_data = OLD::text;
```

```
    operation = 'UPDATE';
  ELSE
    old_data = null;
    operation = 'INSERT';
  END IF;

  table_name = TG_TABLE_NAME;

  INSERT INTO log_table (table_name, operation, old_data)
  VALUES (table_name, operation, old_data);
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER tickets_log_trigger
AFTER INSERT OR UPDATE OR DELETE ON tickets
FOR EACH ROW
EXECUTE FUNCTION create_trigger_function();
```

Проверка работы триггера

```
CALL ticket_sales('P123463',14,110000,'New York Central','Los Angeles
Central','online',5);
```

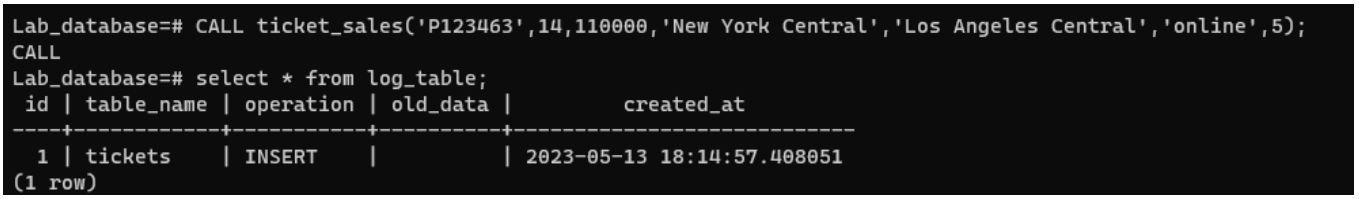

Рис. 4 – Тест №1

```
CALL ticket_refunds(35);
```

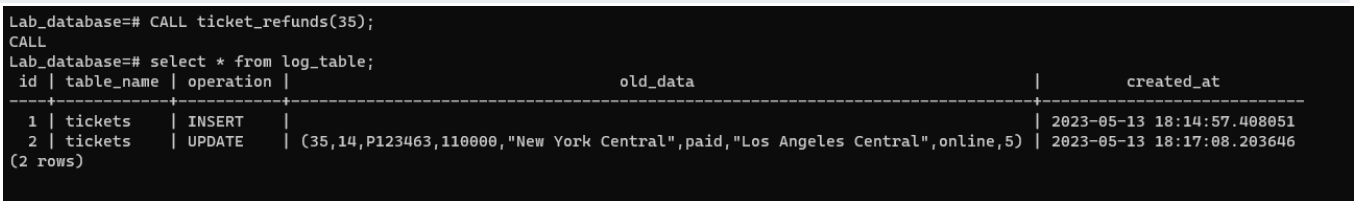

Рис. 5 – Тест №2

```
DELETE FROM tickets WHERE ticket_id = 35;
```

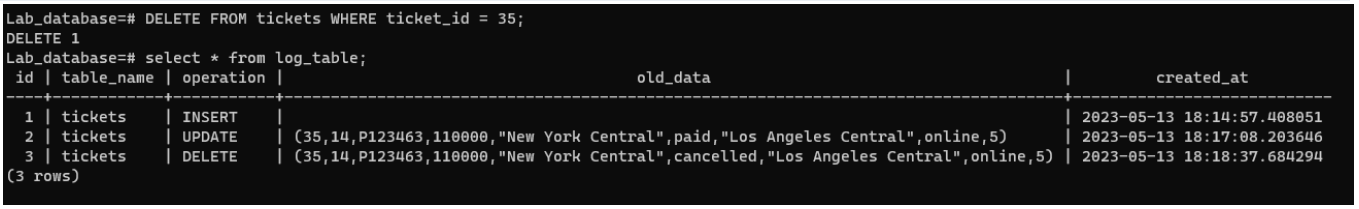

Рис. 6 – Тест №3

4. Вывод

В этой лаборатории я изучил процедуры и функции SQL, а также познакомился с использованием триггеров. Я считаю, что это действительно полезное знание, и оно обязательно поможет будущим программистам.