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

О Т Ч Е Т

по лабораторной работе «Процедуры, функции и триггеры в POSTGRESQL»

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

Автор: Бармина Диана Андреевна

Факультет: ИКТ

Группа: К32421

Преподаватель: Говорова М.М.

Дата сдачи: 24.04.23

Санкт-Петербург 2022

СОДЕРЖАНИЕ

1 Описание работы 3
2 Индивидуальное задание 3
3 Создание процедур/функций 4
3.1 Функция №1 4
3.2 Функция №2 5
3.3 Функция №3 6
4 Модификация триггера с практики 8
5 Создание авторского триггера 10
ЗАКЛЮЧЕНИЕ 12

1 Описание работы

Цель работы: овладеть практическими 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.

Указание. Работа выполняется в консоли SQL Shell (psql).

2 Индивидуальное задание

Вариант 13 – БД «Ресторан»

Задание 4. Создать хранимые процедуры:

- Вывести сведения о заказах заданного официанта на заданную дату.
- Выполнить расчет стоимости заданного заказа.
- Повышения оклада заданного сотрудника на 30 % при повышении его категории.

3 Выполнение работы

3.1 Функция №1

Задача: вывести сведения о заказах заданного официанта на заданную дату.

Код SQL:

```
CREATE OR REPLACE FUNCTION get_all_about_order(employee_id int, order_date date)
RETURNS TABLE(
                    id_waiter int,  date_order date,
                    id_order int, price_order numeric(9,2),
                    table_number int, id_dish int,
                    id_employee int, order_notes text)
as $$ begin return query
SELECT t1.id_employee,
      t1.date_order,
      t1.id_order,
      t1.price_order,
      t1.table_number,
      t2.id_dish,
      t2.id_employee,
      t2.order_notes
FROM "Restaurant_schema"."Order" t1,
    "Restaurant_schema"."Order_fulfillment" t2
WHERE t1.id_employee = employee_id
  AND t1.date_order = order_date
  AND t2.id_order = t1.id_order;
END; $$ LANGUAGE  plpgsql;
```

Выполнение функции:

```
Restaurant=# select * from get_all_about_order(4, '2023-04-08');
 id_waiter | date_order | id_order | price_order | table_number | id_dish | id_employee |           order_notes
-----------+------------+----------+-------------+--------------+---------+-------------+---------------------------------
         4 | 2023-04-08 |        2 |     1950.00 |            1 |       2 |           8 |
         4 | 2023-04-08 |        4 |     1950.00 |            1 |       1 |          10 | Без оливок, без лимона - аллергия
         4 | 2023-04-08 |        4 |     1950.00 |            1 |       6 |           1 |
         4 | 2023-04-08 |        6 |     1950.00 |            1 |       2 |           6 | Без корицы
         4 | 2023-04-08 |        7 |     2500.00 |            2 |       2 |           2 |
         4 | 2023-04-08 |        7 |     2500.00 |            2 |       1 |           2 |
(6 строк)
```

Рисунок 1 – Выполнение функции №1

3.2 Функция №2

Задача: выполнить расчет стоимости заданного заказа.

При создании функции я воспользовалась представлением №1 из прошлой лабораторной работы.

Код SQL:

```
CREATE OR REPLACE FUNCTION get_orders_cost(order_id int)
RETURNS numeric(9,2) as $$
DECLARE res numeric(9,2);
BEGIN
        res =
          (SELECT round(sum(round)*1.4, 2)
           FROM dish_price_ing
           WHERE id_dish in
              (SELECT id_dish
               FROM "Restaurant_schema"."Order_fulfillment"
               WHERE id_order = order_id));
RETURN res;
END;$$ LANGUAGE plpgsql;
```

Выполнение функции:

```
Restaurant=# select * from get_orders_cost(4);
 get_orders_cost
-----------------
           14.31
(1 строка)
```

Рисунок 2 – Выполнение функции №2

3.3 Функция №3

Задача: Повышения оклада заданного сотрудника на 30 % при повышении его категории. Реализация данной задачи невозможна, так как оклад в базе данных привязан не к таблице «Сотрудники», а к таблице «Должность». Была придумана новая задача: повышение категории повара, когда он приготовит определенное количество блюд (с 1 на 8 при 4, с 8 до 3 при 5 – цифры маленькие, чтобы не забивать таблицы большим количеством данных).

Код SQL:

```
CREATE OR REPLACE FUNCTION change_post()
RETURNS TRIGGER AS $$
BEGIN
UPDATE "Restaurant_schema"."Employee"
SET id_post =
 (SELECT t2.new_post
  FROM
   (SELECT CASE
           WHEN t1.id_post = 1
               AND
                (SELECT COUNT(id_oreder_fulfillment)
                 FROM "Restaurant_schema"."Order_fulfillment"
                 WHERE id_employee=NEW.id_employee) = 4 THEN 8
           WHEN t1.id_post = 8
               AND
                (SELECT COUNT(id_oreder_fulfillment)
                 FROM "Restaurant_schema"."Order_fulfillment"
                 WHERE id_employee=NEW.id_employee) = 5 THEN 3
       END AS new_post,
       t1.id_employee
    FROM "Restaurant_schema"."Employee" t1) AS t2
  WHERE t2.id_employee=1)
WHERE id_employee=NEW.id_employee;
RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER change_post_trg AFTER INSERT ON
"Restaurant_schema"."Order_fulfillment" FOR EACH ROW EXECUTE procedure
change_post();
```

```
Restaurant=# select count(id_oreder_fulfillment), id_employee from "Restaurant_schema"."O
rder_fulfillment" group by id_employee;
 count | id_employee
-------+-------------
     1 |           8
     1 |           9
     1 |          10
     3 |           1
     2 |           2
     1 |           6
Restaurant=# select id_employee, id_post from "Restaurant_schema"."Employee";
 id_employee | id_post
-------------+---------
           6 |       3
          10 |       3
          11 |       3
           2 |       3
           3 |       4
           5 |       5
           8 |       1
           9 |       1
          12 |       1
           4 |       5
           1 |       1
Restaurant=# insert into "Restaurant_schema"."Order_fulfillment" (id_dish, id_order, id_e
mployee, ready_status) values (1, 31, 1, 'в процессе');
INSERT 0 1
Restaurant=# select count(id_oreder_fulfillment), id_employee from "Restaurant_schema"."O
rder_fulfillment" group by id_employee;
 count | id_employee
-------+-------------
     1 |           8
     1 |           9
     1 |          10
     4 |           1
     2 |           2
     1 |           6
Restaurant=# select id_employee, id_post from "Restaurant_schema"."Employee";
 id_employee | id_post
-------------+---------
           6 |       3
          10 |       3
          11 |       3
           2 |       3
           3 |       4
           5 |       5
           8 |       1
           9 |       1
          12 |       1
           4 |       5
           1 |       8
```

Рисунок 3 – Работа триггерной функции

4 Модификация триггера с практической работы

Задача: модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника. Слабые места изначального триггера:

Код SQL создания изначального триггера:

```
CREATE OR REPLACE FUNCTION fn_check_time_punch() RETURNS TRIGGER AS
$PSQL$
   BEGIN
      IF new.is_out_punch = (
        SELECT tps.is_out_punch
        FROM time_punch tps
        WHERE tps.employee_id = new.employee_id
        ORDER BY tps.id desc limit 1
      ) THEN
         RETURN NULL;
      END IF;
      RETURN new;
   END;
$PSQL$ LANGUAGE plpgsql;
```

Код SQL улучшенного триггера:

```
CREATE OR REPLACE FUNCTION fn_check_time_punch()
RETURNS TRIGGER AS $psql$ BEGIN
IF (new.punch_time >
     (SELECT max(punch_time)
      FROM time_punch
      WHERE employee_id = new.employee_id)
   AND new.punch_time <= now()
   AND new.is_out_punch !=
    (SELECT is_out_punch
     FROM
      (SELECT *
       FROM time_punch
       WHERE employee_id = new.employee_id
       ORDER BY punch_time DESC
       LIMIT 1) AS t1))
OR (NOT EXISTS
     (SELECT max(punch_time)
      FROM time_punch
      WHERE employee_id = new.employee_id)
   AND new.punch_time <= now()
   AND NOT new.is_out_punch)
THEN RETURN NEW;
END IF; RETURN NULL;
END; $psql$ LANGUAGE plpgsql;
```

```
emp_time=# select * from time_punch
emp_time-# ;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  4 |           1 | f            | 2023-04-17 11:13:22.299114
  5 |           1 | t            | 2023-04-17 11:00:00
  7 |           1 | f            | 2023-04-17 11:16:05.839156
  8 |           1 | t            | 2023-04-17 11:16:15.225425
  9 |           1 | f            | 2023-04-17 11:16:10
 10 |           2 | t            | 2023-04-17 11:18:13.714063
 12 |           2 | f            | 2023-04-17 11:23:15.065526
 14 |           2 | t            | 2023-04-17 11:24:28.261359
```

```
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, fals
e, '2023-04-17 11:23:30');
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, fals
e, '2023-04-17 11:20:30');
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, true
, now());
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, fals
e, '2023-04-28 10:00:00');
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, true
, '2023-04-28 10:00:00');
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, true
, '2023-04-17 11:23:30');
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, fals
e, now());
INSERT 0 1
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, fals
e, now());
INSERT 0 0
emp_time=# insert into time_punch (employee_id, is_out_punch, punch_time) values (2, true
, now());
INSERT 0 1
```

Рисунок 4 – Работа улучшенного триггера

5 Создание авторского триггера

Задача: нельзя добавлять в таблицу «Заказы» заказ, который обслуживается официантом, которого нет в смене на день заказа; нельзя изменять номер официанта в таблице заказы, если в тот день его не было в смене.

Код SQL:

```
CREATE OR REPLACE FUNCTION add_new_order()
RETURNS TRIGGER AS $psql$
BEGIN
        IF new.id_employee not in
          (SELECT id_employee
           FROM "Restaurant_schema"."Shift_allocation" t1
           JOIN "Restaurant_schema"."Shift" t2
                ON t1.id_shift=t2.id_shift
           WHERE shift_date = new.date_order);
        THEN RETURN NULL;
        END IF;
RETURN NEW;
END;
$PSQL$ LANGUAGE plpgsql;

CREATE TRIGGER check_employee_in_shift
BEFORE INSERT ON "Restaurant_schema"."Order"
FOR EACH ROW EXECUTE PROCEDURE add_new_order();
```

Работа триггера:

```
Restaurant=# update "Restaurant_schema"."Order" set id_employee=4 where date_order=date(n
ow()) and table_number=3;
UPDATE 0
```

```
Restaurant=# insert into "Restaurant_schema"."Order" (id_employee, date_order, status_ord
er, price_order, table_number) values (4, '2023-04-22', 'accepted', 2000.00, 3);
INSERT 0 0
Restaurant=# insert into "Restaurant_schema"."Order" (id_employee, date_order, status_ord
er, price_order, table_number) values (5, '2023-04-22', 'accepted', 2000.00, 3);
INSERT 0 1
```

```
Restaurant=# select * from "Restaurant_schema"."Shift_allocation";
 id_shift | id_employee | table_number | id_shift_allocation
----------+-------------+--------------+---------------------
        7 |           4 |            3 |                   2
        7 |           4 |            5 |                   3
        8 |           1 |            1 |                   4
        8 |           1 |            3 |                   5
        7 |           3 |            8 |                   1
       21 |           3 |            1 |                   6
       21 |           3 |            8 |                   7
       21 |           3 |            1 |                   9
       21 |           3 |            8 |                  10
       22 |           5 |            5 |                  13
       22 |           5 |            3 |                  14
(11 строк)

Restaurant=# select * from "Restaurant_schema"."Shift";
 id_shift | shift_date | start_time | end_time
----------+------------+------------+----------
        6 | 2023-03-19 | 09:00:00   | 17:00:00
        7 | 2023-03-19 | 17:00:00   | 23:00:00
        8 | 2023-03-20 | 09:00:00   | 17:00:00
        9 | 2023-03-20 | 16:00:00   | 23:00:00
       10 | 2023-03-21 | 09:00:00   | 17:00:00
       11 | 2023-03-21 | 17:00:00   | 23:00:00
       12 | 2023-03-22 | 09:00:00   | 17:00:00
       13 | 2023-03-22 | 15:00:00   | 23:00:00
       15 | 2023-04-08 | 15:00:00   | 23:00:00
       17 | 2023-04-10 | 09:00:00   | 17:00:00
       18 | 2023-04-10 | 17:00:00   | 23:00:00
       19 | 2023-04-10 | 09:00:00   | 17:00:00
       20 | 2023-04-10 | 09:00:00   | 17:00:00
       21 | 2023-04-22 | 09:00:00   | 17:00:00
       22 | 2023-04-22 | 17:00:00   | 23:00:00
(15 строк)
```

Рисунок 5 – Работа триггера

ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы были приобретены навыки создания процедур, функций и триггеров.

В результате данной лабораторной работы были созданы две процедуры из индивидуального задания, а также авторские функция и триггер. Модификация триггера из практической работы помогла лучше понять принцип работы триггера и его возможности.