

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет инфокоммуникационных технологий

Лабораторная работа №5
«Работа с триггерами и функциями в
POSTGRESQL»
по дисциплине:
«Базы данных»

Выполнила:

студентка II курса ИКТ

группы К3242

Липина Ольга Андреевна

Проверил:

Говорова Марина Михайловна

Санкт-Петербург

2022

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).

Задания

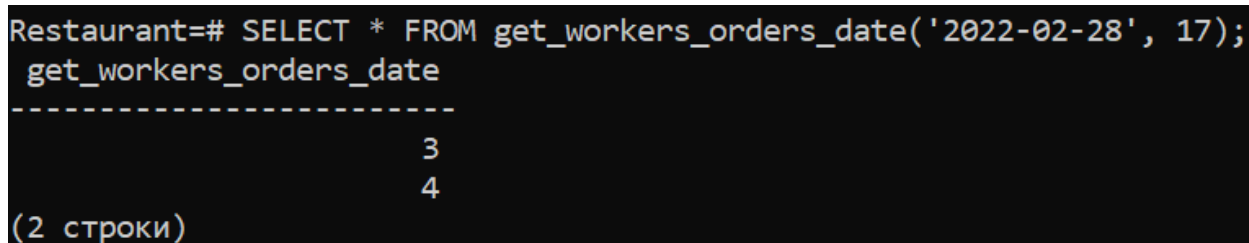
Задание: вывести сведения о заказах заданного официанта на заданную дату.

Я решила, что здесь уместнее использовать функцию, так как в процедурах непосредственный возврат данных происходит не как результат выполнения функции, и для получения данных нужны дополнительные строчки кода и усилия, а сведения о заказах официанта на определённую дату мы, вероятно, будем использовать как аргументы в последующих запросах баз данных, и поэтому нам удобнее именно возвращать эти значения, а не выводить в поток вывода.

Код функции:

```
CREATE OR REPLACE FUNCTION get_workers_orders_date("Date" date,
"Worker_id" integer)
  RETURNS SETOF integer AS $$
SELECT orders.id
  FROM "Restaraunt_schema".orders
  JOIN "Restaraunt_schema".workers ON workers.id=orders.worker_id
  WHERE workers.id="Worker_id" AND orders.date="Date";
$$
language sql;
```

Пример выполнения:



```
Restaurant=# SELECT * FROM get_workers_orders_date('2022-02-28', 17);
get_workers_orders_date
-----
                3
                4
(2 строки)
```

Рисунок 1 – Получение заказов официанта на заданную дату

Задание: выполнить расчет стоимости заданного заказа.

По упомянутым выше причинам здесь тоже показалось, что уместнее использовать функцию.

Пример выполнения:

```
Restaurant=# SELECT * FROM get_order_bill(11);
get_order_bill
-----
              3800.0
(1 строка)
```

Рисунок 2 – Расчёт стоимости заданного заказа

Код функции:

```
CREATE OR REPLACE FUNCTION get_order_bill("order_id" integer)
RETURNS numeric AS $$
    SELECT SUM(order_content.quantity * price * ingr_quantity) AS bill
    FROM "Restaraunt_schema".orders
    INNER JOIN "Restaraunt_schema".order_content
    ON "Restaraunt_schema".orders.id="Restaraunt_schema".order_content.id_order
    INNER JOIN "Restaraunt_schema".dishes
    ON "Restaraunt_schema".order_content.id_dish="Restaraunt_schema".dishes.id
    INNER JOIN "Restaraunt_schema".ingr_quan
    ON "Restaraunt_schema".dishes.id="Restaraunt_schema".ingr_quan.id_dish
    INNER JOIN "Restaraunt_schema".purch_ingr
    ON "Restaraunt_schema".ingr_quan.id_ingr="Restaraunt_schema".purch_ingr.id_ingr
    GROUP BY id_order
    HAVING id_order="order_id";
$$
language sql;
```

Задание: создать процедуру для повышения оклада заданного сотрудника на 30 % при повышении его категории. Создать необходимый триггер.

Пример выполнения:

```
Restaurant=# SELECT category, salary
Restaurant=# FROM "Restaraunt_schema".workers WHERE id=3;
  category |      salary
-----+-----
team_leader | 1990.482000000000000000
(1 строка)

Restaurant=# UPDATE "Restaraunt_schema".workers
Restaurant=# SET category='partner'
Restaurant=# WHERE id=3;
ЗАМЕЧАНИЕ: old_pos old "team_leader" supplied by "2587.6266000000000000"
UPDATE 1
Restaurant=# SELECT category, salary
Restaurant=# FROM "Restaraunt_schema".workers WHERE id=3;
  category |      salary
-----+-----
partner   | 2587.626600000000000000
(1 строка)
```

Рисунок 3 – Срабатывание триггера на повышение категории сотрудника – изменение зарплаты

Код функции и триггера:

```
CREATE OR REPLACE FUNCTION new_salary() RETURNS trigger AS $$
DECLARE
    new_pos integer = (SELECT int_pos FROM
    "Restaraunt_schema".position_int
    WHERE (string_pos=NEW.category));

    old_pos integer = (SELECT int_pos FROM
    "Restaraunt_schema".position_int
    WHERE (string_pos=OLD.category));
BEGIN
    IF new_pos > old_pos THEN
        NEW.salary := NEW.salary * 13 / 10;
        RETURN NEW;
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER
    new_salary_1
BEFORE UPDATE ON
    "Restaraunt_schema".workers
FOR EACH ROW EXECUTE PROCEDURE
    new_salary();
```

Задание: создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL

Код функции и триггера:

```
CREATE OR REPLACE FUNCTION logging_events () RETURNS
TRIGGER AS $$
DECLARE
    mstr varchar(30);
    astr varchar(100);
    retstr varchar(254);
BEGIN
IF TG_OP = 'INSERT' THEN
    astr = NEW;
    mstr := 'Add new';
    retstr := mstr || astr;
    INSERT INTO
    "Restaraunt_schema".logs(text,added,table_name) values
    (retstr, NOW(), TG_TABLE_NAME);
    RETURN NEW;
ELSIF TG_OP = 'UPDATE' THEN
    astr = NEW;
    mstr := 'Update';
    retstr := mstr || astr;
    INSERT INTO
    "Restaraunt_schema".logs(text,added,table_name) values
    (retstr,NOW(), TG_TABLE_NAME);
    RETURN NEW;
ELSIF TG_OP = 'DELETE' THEN
    astr = OLD;
    mstr := 'Remove';
    retstr := mstr || astr;
    INSERT INTO
    "Restaraunt_schema".logs(text,added,table_name) values
    (retstr,NOW(), TG_TABLE_NAME);
    RETURN OLD;
END IF;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER log_workers AFTER INSERT OR UPDATE OR DELETE ON
"Restaraunt_schema".workers FOR EACH ROW EXECUTE PROCEDURE logging_events ();
```

```
Restaurant=# UPDATE "Restaraunt_schema".workers
Restaurant=# SET category='middle'
Restaurant=# WHERE id=3;
UPDATE 1
Restaurant=# select * from "Restaraunt_schema".logs;
 id |          text          |          added          | table_name
-----+-----+-----+-----
  1 | Update(3,"Вольфган Джон Марьянович",middle,"1005 125757",cooker,2587.6266000000000000) | 23:51:51.715107+03 | workers
(1 строка)
```

Рисунок 4 – Срабатывание триггера на изменение данных

Выводы:

С помощью инструментов командной строки `psql` я научилась создавать хранимые функции для таблиц базы данных, а также создавать триггерные функции и триггеры, срабатывающие во время наступления определённых событий в данных.