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 K32402
Пластун Елизавета Олеговна

_____
(подпись)

Проверил(а):
Говорова Марина Михайловна

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург
2023 г.

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 13. БД «Ресторан»

Описание предметной области: Необходимо создать систему для обслуживания заказов клиентов в ресторане.

Сотрудники ресторана – повара и официанты.

За каждым официантом закреплены определенные столы за смену. Клиенты могут бронировать столы заранее.

Каждый повар может готовить определенный набор блюд.

Официант принимает заказ от стола и передает его на кухню. Шеф-повар распределяет блюда для приготовления между поварами. В одном заказе может быть несколько одинаковых или разных блюд.

Запас продуктов на складе не должен быть ниже заданного значения.

Цена заказа складывается из стоимости ингредиентов и наценки, которая составляет 40% стоимости ингредиентов.

БД должна содержать следующий минимальный набор сведений: Табельный номер сотрудника. ФИО сотрудника. Паспортные данные сотрудника. Категория сотрудника. Должность сотрудника. Оклад сотрудника. Наименование ингредиента. Код ингредиента. Дата закупки. Объем закупки. Количество продукта на складе. Необходимый запас продукта. Срок годности. Цена ингредиента. Калорийность (на 100г продукта). Поставщик. Наименование блюда. Код блюда. Объем ингредиента. Номер стола. Дата заказа. Код заказа. Количество. Название блюда. Ингредиенты, входящие в блюдо. Тип ингредиента.

Задание 1.1 (ЛР 1 БД). Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

Задание 1.2. Создайте логическую модель БД, используя ИЛМ (задание 1.1). Используйте необходимые средства поддержки целостности данных в СУБД.

Задание 2. Создать запросы:

- Вывести данные официанта, принявшего заказы на максимальную сумму за истекший месяц.
- Рассчитать премию каждого официанта за последние 10 дней (5% от стоимости каждого заказа).
- Подсчитать, сколько ингредиентов содержит каждое блюдо.
- Вывести название блюда, содержащее максимальное число ингредиентов.
- Какой повар может приготовить максимальное число видов блюд?
- Сколько закреплено столов за каждым из официантов?
- Какой из ингредиентов используется в максимальном количестве блюд?

Задание 3. Создать представление:

- для расчета стоимости ингредиентов для заданного блюда;
- количество приготовленных блюд по каждому блюду за определенную дату.

Задание 4. Создать хранимые процедуры:

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

Задание 5. Создать необходимые триггеры.

Выполнение:
Создать хранимые процедуры:

Вывести сведения о заказах заданного официанта на заданную дату.

```
postgres=# create or replace function orders_employee_for_date(date DATE,id_person INTEGER)
postgres-# returns table(id_order INTEGER, price DOUBLE PRECISION, id_table INTEGER, date_order DATE) as $$ BEGIN
```

```
postgres$# return query
postgres$# select "lr1.2".order.id_order,"lr1.2".order.price, "lr1.2".order.id_table,
"lr1.2".order.date_order from  "lr1.2".order where "lr1.2".order.date_order = date and
"lr1.2".order.id_employee = id_person;
postgres$# END;
postgres$# $$ language plpgsql;
```

```
postgres=# create or replace function orders_employee_for_date(date DATE,id_person INTEGER)
postgres-# returns table(id_order INTEGER, price DOUBLE PRECISION, id_table INTEGER, date_order DATE) as $$ BEGIN
postgres$# return query
postgres$# select "lr1.2".order.id_order,"lr1.2".order.price, "lr1.2".order.id_table, "lr1.2".order.date_order from  "lr1.2".order where "lr1.2".order.date_or
der = date and  "lr1.2".order.id_employee = id_person;
postgres$# END;
postgres$# $$ language plpgsql;
CREATE FUNCTION
```

```
postgres=# select * from orders_employee_for_date('2023-04-05', 8);
 id_order | price  | id_table | date_order
----------+--------+----------+------------
        9 | 450000 |        1 | 2023-04-05
        6 |  40000 |        1 | 2023-04-05
(2 rows)
```

Выполнить расчет стоимости заданного заказа.

```
postgres=# create or replace function total_price_for_order(id INTEGER)
postgres-# returns INTEGER as $$ BEGIN
postgres$# return
postgres$# (select sum(total_sum_for_dish) from (select
"lr1.2".order_list.id_order, "lr1.2".order_list.id_dish, "lr1.2".order_list.amount *
"lr1.2".dish.price as total_sum_for_dish from "lr1.2".order_list join "lr1.2".dish
using(id_dish) where "lr1.2".order_list.id_order = id ) as h);
postgres$# END;
postgres$# $$ language plpgsql;
```

```
postgres=# create or replace function total_price_for_order(id INTEGER)
postgres-# returns INTEGER as $$ BEGIN
postgres$# return
postgres$# (select sum(total_sum_for_dish) from (select "lr1.2".order_list.id_order, "lr1.2".order_list.id_dish, "lr1.2".order_list.amount * "lr1.2".dish.pric
e as total_sum_for_dish from "lr1.2".order_list join "lr1.2".dish using(id_dish) where "lr1.2".order_list.id_order = id ) as h);
postgres$# END;
postgres$# $$ language plpgsql;
CREATE FUNCTION
```

```
postgres=# select * from total_price_for_order(1);
 total_price_for_order
-----------------------
                 45694
(1 row)

postgres=#
```

Повышение оклада сотрудников

```
create procedure update_salariess() language sql as $$
 update "lr1.2".position set salary = case when category = 1 then
(select 1.3 * salary
from "lr1.2".position where category = 1)
when category = 2 then
(select 1.6 * salary
from "lr1.2".position where category = 2)
 when category = 3 then
(select 1.9 * salary
from "lr1.2".position where category = 3)
 when category = 4 then
(select 2 * salary
from "lr1.2".position where category = 4)
 when category = 5 then
(select 2.1 * salary
from "lr1.2".position where category = 5)
else 0
end $$;
```

```
postgres=# create procedure update_salary() language sql as $$
postgres$# update "lr1.2".position set salary = case when category = 1 then
postgres$# (select 1.3 * salary
postgres$# from "lr1.2".position where category = 1)
postgres$#  when category = 2 then
postgres$#  (select 1.6 * salary
postgres$# from "lr1.2".position where category = 2)
postgres$#  when category = 3 then
postgres$# (select 1.9 * salary
postgres$# from "lr1.2".position where category = 3)
postgres$# else 0
postgres$# end $$;
CREATE PROCEDURE
postgres=#
```

```
postgres=# create procedure update_salariess() language sql as $$
 update "lr1.2".position set salary = case when category = 1 then
(select 1.3 * salary
from "lr1.2".position where category = 1)
when category = 2 then
(select 1.6 * salary
from "lr1.2".position where category = 2)
 when category = 3 then
(select 1.9 * salary
from "lr1.2".position where category = 3)
 when category = 4 then
(select 2 * salary
from "lr1.2".position where category = 4)
 when category = 5 then
(select 2.1 * salary
from "lr1.2".position where category = 5)
else 0
end $$;
CREATE PROCEDURE
postgres=# select * from "lr1.2".position
ORDER BY id_position ASC
;
 id_position | name_position | salary  | category
-------------+---------------+---------+----------
           1 | о ф и ц и а н т       |  113230 |        1
           2 | м е н е д ж е р       |  128000 |        2
           3 | п о в а р         |  108300 |        3
           4 | ш е в           |     100 |        4
           5 | п и а н и с т       | 1000000 |        5
(5 rows)

postgres=# call update_salariess();
CALL
postgres=# select * from "lr1.2".position
ORDER BY id_position ASC
;
 id_position | name_position | salary  | category
-------------+---------------+---------+----------
           1 | о ф и ц и а н т       |  147199 |        1
           2 | м е н е д ж е р       |  204800 |        2
           3 | п о в а р         |  205770 |        3
           4 | ш е в           |     200 |        4
           5 | п и а н и с т       | 2100000 |        5
(5 rows)

postgres=#
```

Триггеры

Записываем логи

```
CREATE OR REPLACE FUNCTION do_log()
        RETURNS TRIGGER AS $$
        DECLARE
        mstr varchar(30);
        astr varchar(100);
        retstr varchar(254);
        BEGIN
        IF TG_OP = 'INSERT' THEN
        astr = NEW;
        mstr := 'Add data ';
        retstr := mstr||astr;
        INSERT INTO "lr1.2".logs(text, added, table_name) values
        (retstr,NOW(), TG_TABLE_NAME);
        RETURN NEW;
        ELSIF TG_OP = 'UPDATE' THEN
        astr = NEW;
        mstr := 'Update data ';
        retstr := mstr||astr;
        INSERT INTO "lr1.2".logs(text, added, table_name) values
        (retstr,NOW(), TG_TABLE_NAME);
        RETURN NEW;
        ELSIF TG_OP = 'DELETE' THEN
        astr = OLD;
        mstr := 'Remove data ';
        retstr := mstr || astr;
        INSERT INTO "lr1.2".logs(text, added, table_name) values
        (retstr,NOW(), TG_TABLE_NAME);
        RETURN OLD;
        END IF;
        END;
        $$ LANGUAGE plpgsql;



        CREATE TRIGGER t_employee AFTER INSERT OR UPDATE OR
DELETE ON
        "lr1.2".employee FOR EACH ROW EXECUTE PROCEDURE
        do_log();
```

```
                    ^
postgres=# CREATE OR REPLACE FUNCTION do_log()
                RETURNS TRIGGER AS $$
                DECLARE
                mstr varchar(30);
                astr varchar(100);
                retstr varchar(254);
                BEGIN
                IF TG_OP = 'INSERT' THEN
                astr = NEW;
                mstr := 'Add data ';
                retstr := mstr||astr;
                INSERT INTO "lr1.2".logs(text, added, table_name) values
                (retstr,NOW(), TG_TABLE_NAME);
                RETURN NEW;
                ELSIF TG_OP = 'UPDATE' THEN
                astr = NEW;
                mstr := 'Update data ';
                retstr := mstr||astr;
                INSERT INTO "lr1.2".logs(text, added, table_name) values
                (retstr,NOW(), TG_TABLE_NAME);
                RETURN NEW;
                ELSIF TG_OP = 'DELETE' THEN
                astr = OLD;
                mstr := 'Remove data ';
                retstr := mstr || astr;
                INSERT INTO "lr1.2".logs(text, added, table_name) values
                (retstr,NOW(), TG_TABLE_NAME);
                RETURN OLD;
                END IF;
                END;
                $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
postgres=# CREATE TRIGGER t_employee AFTER INSERT OR UPDATE OR DELETE ON
                "lr1.2".employee FOR EACH ROW EXECUTE PROCEDURE
                do_log();
CREATE TRIGGER
```

```
postgres=# insert into "lr1.2".employee (id_employee, full_name, passport_datas, id_position) values(13,'шмель шамиль и баба клава','5 55
3535',2);
INSERT 0 1
postgres=# update "lr1.2".employee set full_name= 'hjhffhgghgrb' where id_employee = 1;
UPDATE 1
```

```
postgres=# delete from "lr1.2".employee where id_employee= 13;
DELETE 1
postgres=#
```

update "lr1.2".employee set full_name= 'hjhffhgghgrb' where id_employee = 1;
delete from "lr1.2".employee where id_employee= 13;
insert into "lr1.2".employee (id_employee, full_name, passport_datas, id_position) values(13,'шмель шамиль и баба клава','5 553535',2);

```
postgres=# select * from "lr1.2".logs
postgres-# ;
                        text                         |   added    | table_name
-----------------------------------------------------+------------+------------
 Add data (13,"шмель шамиль и баба клава ","5 553535",2)    | 2023-05-31 | employee
 Update data (1,hjhffhgghgrb,д ц г р а п о а п в ж д у р д к и п л о ц ,1)      | 2023-05-31 | employee
 Remove data (13,"шмель шамиль и баба клава ","5 553535",2) | 2023-05-31 | employee
(3 rows)

postgres=#
```