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

Пономарев Константин Витальеви

(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.

```
create or replace function ordersEmployeeForDate(date DATE, personId
INTEGER)
 returns table (orderId INTEGER, totalPrice INTEGER, tableId
INTEGER, orderDate DATE) as $$ BEGIN
 return query
 select public."orderTable"."orderId",
public."orderTable"."totalPrice", public."orderTable"."tableId",
public."orderTable"."orderDate" from public."orderTable" where
public."orderTable"."orderDate" = date and
public."orderTable"."employeeId" = personId;
 END;
 $$ language plpgsql;
```

```
postgres=# create or replace function ordersEmployeeForDate(date DATE, personId INTEGER)
postgres-#  returns table (orderId INTEGER, totalPrice INTEGER, tableId INTEGER, orderDate DATE) as $$ BEGIN
postgres$#  return query
postgres$#  select public."orderTable"."orderId", public."orderTable"."totalPrice", public."orderTable"."tableId", public."ord
erTable"."orderDate" from public."orderTable" where public."orderTable"."orderDate" = date and public."orderTable"."employeeId
" = personId;
postgres$#  END;
postgres$#  $$ language plpgsql;
CREATE FUNCTION
postgres=# select * from ordersEmployeeForDate('2023-08-28',1);
 orderid | totalprice | tableid | orderdate
---------+------------+---------+------------
       2 |      12122 |       0 | 2023-08-28
(1 row)
```

Выполнить расчет стоимости заданного заказа

```
create or replace function totalPriceForOrder(id INTEGER)  returns INTEGER as $$ BEGIN
return  (select sum(totalSumForDish) from (select public.order_list.id_order,
public.order_list.id_dish, public.order_list.amount * public.dish.price as totalSumForDish
from public.order_list join public.dish using(id_dish) where public.order_list.id_order = id )
as h);  END;  $$ language plpgsql;
```

```
postgres=#  create or replace function totalPriceForOrder(id INTEGER)
postgres-#  returns INTEGER as $$ BEGIN
postgres$#  return  (select sum("totalPrice") from (select ("order_details"."count" * "dish"."price") as "totalPrice",
postgres$# public."order_details"."dishId" from public."order_details" join
postgres$# public."dish" using("dishId") where "orderId" = id) as f);
postgres$#  END;  $$ language plpgsql;
CREATE FUNCTION
postgres=# select * from totalPriceForOrder(1)
postgres-# ;
 totalpricefororder
--------------------
               1480
(1 row)

postgres=#
```

Повышение оклада сотрудников

```
postgres=# create procedure updateSalaries() language sql as $$
postgres$# update public."position" set "salary" = case when "category" = 1 then
postgres$# (select 1.5 * "salary" from public."position" where "category" = 1)
postgres$# when "category" = 2 then
postgres$# (select 1.8 * "salary" from public."position" where "category" = 2)
postgres$# when "category" = 3 then
postgres$# (select 5 * "salary" from public."position" where "category" = 3)
postgres$# when "category" = 4 then
postgres$# (select 2.5 * "salary" from public."position" where "category" = 10)
postgres$# else 0
postgres$# end $$;
CREATE PROCEDURE
```

```
postgres=# select * from public."position";
 positionId |  salary  |   name   | category
------------+----------+----------+----------
          0 |   126563 | Официант |        1
          1 |   629856 | Менеджер |        2
          2 | 93750000 | Директор |        3
          3 |        0 | Повар    |       10
postgres=# select * from public."position";
 positionId |  salary  |   name   | category
------------+----------+----------+----------
          0 |    84375 | Официант |        1
          1 |   349920 | Менеджер |        2
          2 | 18750000 | Директор |        3
```

Триггеры

Логирование таблицы с сотрудниками

```
CREATE OR REPLACE FUNCTION logData() RETURNS TRIGGER AS $$
DECLARE
prefix varchar(30);
columnStr varchar(100);
mappedString varchar(254);
BEGIN
IF TG_OP = 'INSERT' THEN
columnStr = NEW;
prefix := 'Add data ';
mappedString := prefix||columnStr;
INSERT INTO public."logs"("text", "addDate", "tableName") values
(mappedString,NOW(), TG_TABLE_NAME);
RETURN NEW;
ELSIF TG_OP = 'UPDATE' THEN
columnStr = NEW;
prefix := 'Update data:';
mappedString := prefix||columnStr;
INSERT INTO public."logs"("text", "addDate", "tableName") values
(mappedString,NOW(), TG_TABLE_NAME);
RETURN NEW;
ELSIF TG_OP = 'DELETE' THEN
columnStr = OLD;
prefix := 'Remove data:';
mappedString := prefix || columnStr;
INSERT INTO public."logs"("text", "addDate", "tableName") values
(mappedString,NOW(), TG_TABLE_NAME);
RETURN OLD;
END IF;
END;
$$ LANGUAGE plpgsql;
```

```
postgres=# select * from public."logs";
                 text                  |  addDate   | tableName
---------------------------------------+------------+-----------
 Add data (7,0,PG,"11 123 123")        | 2023-09-12 | employee
 Update data:(3,1,aaa,"13 13 333777") | 2023-09-12 | employee
 Remove data:(3,1,aaa,"13 13 333777") | 2023-09-12 | employee
(3 rows)

postgres=#
```

Выводы

В ходе выполнения лабораторной работы, я научился создавать триггеры, которые могут быть полезным при работе с большим количеством данных, которые необходимо отслеживать (например, если бэкенд отвечает долго на запрос и необходимо разбираться в действиях пользователя)