

Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Лабораторная работа № 3
«Процедуры, функции, триггеры в PostgreSQL»

Выполнил: Галиновский Роман Андреевич
Группа: К3240
Преподаватель: 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).

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

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.

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

ТЕХНОЛОГИЯ ВЫПОЛНЕНИЯ РАБОТЫ:

1. Название БД

Вариант 13. «Ресторан»

Описание предметной области: Сотрудники ресторана – повара и официанты. За каждым официантом закреплены определенные столы. Каждый повар готовит определенный набор 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: 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. Поставщик. Наименование блюда. Код блюда. Объем ингредиента. Номер стола. Дата заказа. Код заказа. Количество. Название блюда. Ингредиенты, входящие в блюдо. Тип ингредиента.

Состав реквизитов сущностей:

- a) **Ингредиент** (ID ингредиента, цена ингредиента, тип ингредиента, название ингредиента, объём ингредиента, количество на складе, срок годности, калорийность)
- b) **Поставщик** (ID поставщика, имя поставщика, город поставщика, страна поставщика)
- c) **Стол** (ID стола, размещение стола, занятость стола, вместимость стола)
- d) **Должность** (ID должности, оклад, категория сотрудника)
- e) **Закупки** (ID закупки, ID поставщика, дата закупки, объём закупки)
- f) **Состав закупки** (ID состава закупки, ID закупки, ID ингредиента, стоимость, нужный запас)
- g) **Сотрудники** (ID сотрудника, ID должности, Фамилия, имя, отчество, паспорт сотрудника, *категория*, *зарплата*)
- h) **Официант** (ID Официанта, ID сотрудника)
- i) **Заказы** (ID заказа, ID стола, ID Официанта, Дата заказа)
- j) **Состав Заказа**(ID заказа, ID блюда, Количество блюда, примечание)

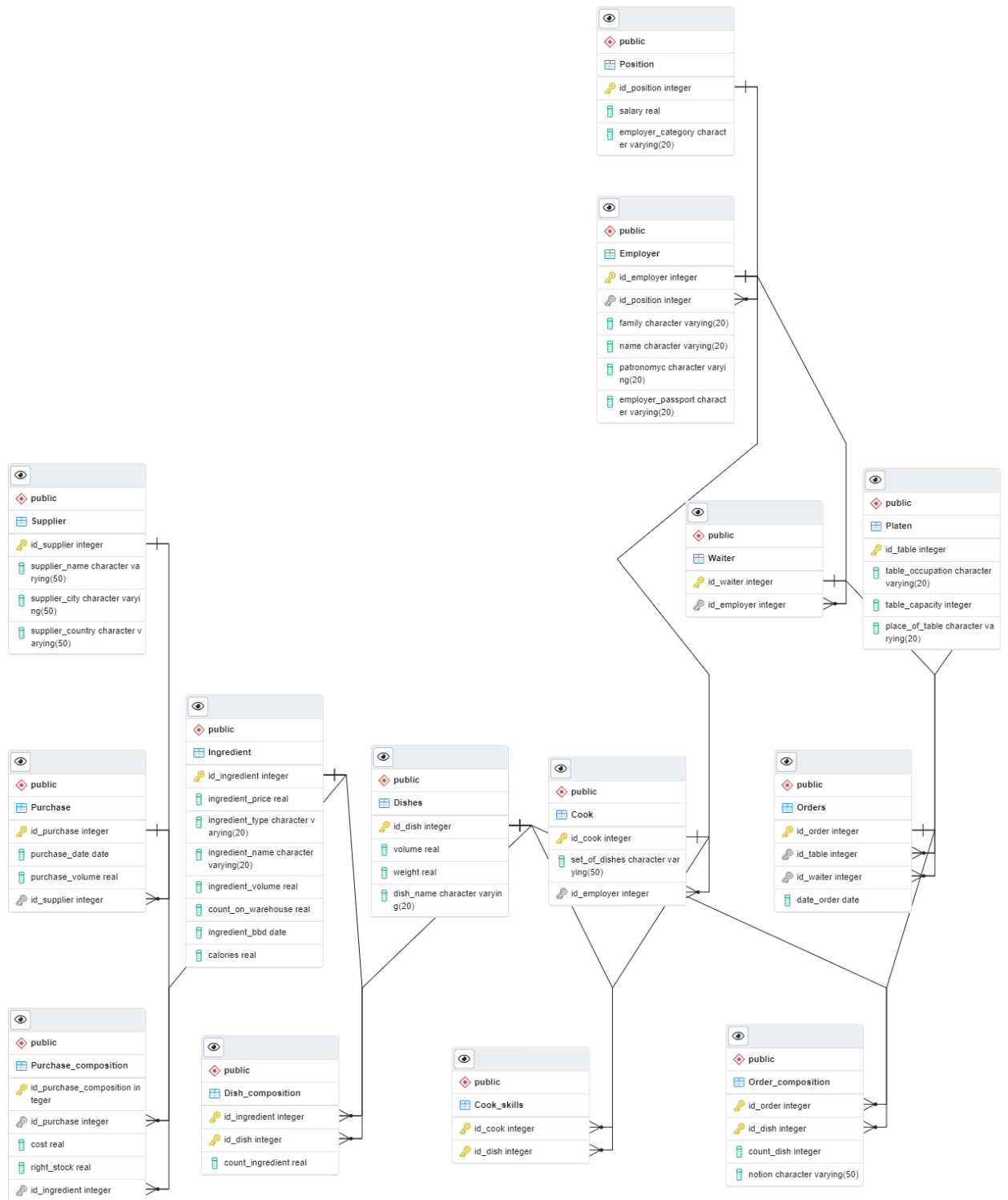
k) Повара (ID Повара, ID сотрудника, набор блюд)

l) Умение повара(ID Повара, ID Блюда)

m) Блюдо (ID блюда, объём, вес, название блюда)

n) Состав Блюда (ID ингредиента, ID блюда, количество ингредиентов)

Схема базы данных:



Выполнение:

1. Создание хранимых процедур:

1.1. Вывести сведения о заказах заданного официанта на заданную дату:

```
CREATE OR REPLACE FUNCTION orders_for_waiters (order_date date, waiter_id integer)
```

```
RETURNS TABLE (family VARCHAR, name VARCHAR,  
order_id integer)
```

```
AS $$
```

```
SELECT public."Employer".family, public."Employer".name,  
public."Orders".id_order
```

```
FROM public."Waiter", public."Employer", public."Orders"
```

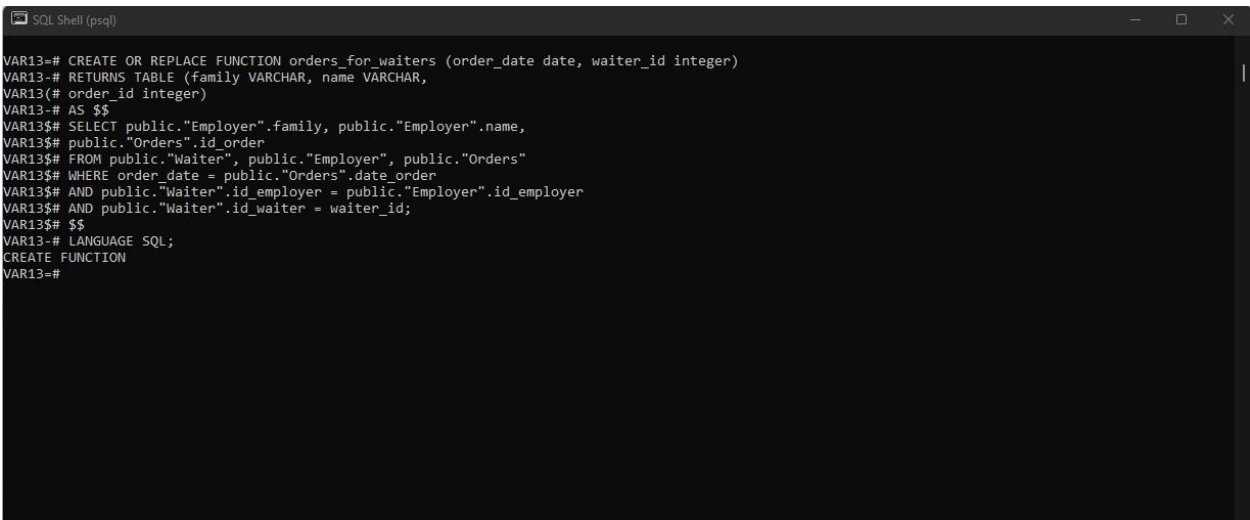
```
WHERE order_date = public."Orders".date_order
```

```
AND public."Waiter".id_employer = public."Employer".id_employer
```

```
AND public."Waiter".id_waiter = waiter_id;
```

```
$$
```

```
LANGUAGE SQL;
```



```
SQL Shell (psql)  
VAR13=# CREATE OR REPLACE FUNCTION orders_for_waiters (order_date date, waiter_id integer)  
VAR13=# RETURNS TABLE (family VARCHAR, name VARCHAR,  
VAR13=# order_id integer)  
VAR13=# AS $$  
VAR13$# SELECT public."Employer".family, public."Employer".name,  
VAR13$# public."Orders".id_order  
VAR13$# FROM public."Waiter", public."Employer", public."Orders"  
VAR13$# WHERE order_date = public."Orders".date_order  
VAR13$# AND public."Waiter".id_employer = public."Employer".id_employer  
VAR13$# AND public."Waiter".id_waiter = waiter_id;  
VAR13$# $$  
VAR13=# LANGUAGE SQL;  
CREATE FUNCTION  
VAR13=#
```

Результат:

При проверке `SELECT * FROM orders_for_waiters('2022-06-10', 111132);`

```
SQL Shell (psql)
VAR13=# LANGUAGE SQL;
CREATE FUNCTION
VAR13=# SELECT * FROM orders_for_waiters('2022-06-10',111132);
 family | name  | order_id
-----+-----+-----
 Romanov | Mikhail | 111700
 Romanov | Mikhail | 111701
 Romanov | Mikhail | 111702
 Romanov | Mikhail | 111703
(4 строки)

VAR13=#
```

1.2 Выполнить расчет стоимости заданного заказа:

CREATE OR REPLACE FUNCTION price_of_current_order (order_id integer)

RETURNS INTEGER

AS \$\$

*SELECT (SUM(public."Ingredient".ingredient_price) *
public."Order_composition".count_dish) AS RES*

*FROM public."Ingredient", public."Order_composition",
public."Dish_composition",public."Dishes"*

*WHERE public."Ingredient".id_ingredient =
public."Dish_composition".id_ingredient*

AND public."Dish_composition".id_dish = public."Dishes".id_dish

AND public."Dishes".id_dish = public."Order_composition".id_dish

AND order_id = public."Order_composition".id_order

GROUP BY public."Order_composition".count_dish

\$\$

LANGUAGE SQL;

```
SQL Shell (psql)
VAR13=# CREATE OR REPLACE FUNCTION price_of_current_order (order_id integer)
VAR13=# RETURNS INTEGER
VAR13=# AS $$
VAR13$# SELECT (SUM(public."Ingredient".ingredient_price) * public."Order_composition".count_dish) AS RES
VAR13$# FROM public."Ingredient", public."Order_composition", public."Dish_composition",public."Dishes"
VAR13$# WHERE public."Ingredient".id_ingredient = public."Dish_composition".id_ingredient
VAR13$# AND public."Dish_composition".id_dish = public."Dishes".id_dish
VAR13$# AND public."Dishes".id_dish = public."Order_composition".id_dish
VAR13$# AND order_id = public."Order_composition".id_order
VAR13$# GROUP BY public."Order_composition".count_dish
VAR13$# $$
VAR13=# LANGUAGE SQL;
CREATE FUNCTION
VAR13=#
```

Результат:

При проверке SELECT price_of_current_order(111702);

SELECT price_of_current_order(111165);

```
SQL Shell (psql)
VAR13=# SELECT price_of_current_order(111702);
price_of_current_order
-----
2400
(1 строка)

VAR13=# SELECT price_of_current_order(111165);
price_of_current_order
-----
2450
(1 строка)

VAR13=#
```

1.3. Повышения оклада заданного сотрудника на 30 % при повышении его категории:

CREATE OR REPLACE FUNCTION salary_increase (employer_id integer)

RETURNS TABLE(family VARCHAR, name VARCHAR, salary real, category integer)

AS \$\$

UPDATE public."Employer"

SET salary = salary * 1.3,

category = category+1

WHERE id_employer IN(SELECT id_employer FROM public."Employer"

WHERE employer_id = public."Employer".id_employer);

SELECT public."Employer".family, public."Employer".name,
public."Employer".salary, public."Employer".category

FROM public."Employer"

WHERE employer_id = public."Employer".id_employer

\$\$

LANGUAGE SQL;

р.с в запросе было сказано сделать исключительно процедуру по UPDATE сотрудника, но я решил помимо самого UPDATE категории и ЗП, сделать так, чтоб функция возвращала таблицу, где измененные данные непосредственно будут видны.

```
SQL Shell (psql)
VAR13=# CREATE OR REPLACE FUNCTION salary_increase (employer_id integer)
VAR13=# RETURNS TABLE(family VARCHAR, name VARCHAR, salary real, category integer)
VAR13=# AS $$
VAR13$# UPDATE public."Employer"
VAR13$# SET salary = salary * 1.3,
VAR13$# category = category+1
VAR13$# WHERE id_employer IN(SELECT id_employer FROM public."Employer"
VAR13$# WHERE employer_id = public."Employer".id_employer);
VAR13$# SELECT public."Employer".family, public."Employer".name, public."Employer".salary, public."Employer".category
VAR13$# FROM public."Employer"
VAR13$# WHERE employer_id = public."Employer".id_employer
VAR13$# $$
VAR13=# LANGUAGE SQL;
CREATE FUNCTION
VAR13=#
```

Результат:

До:

```
VAR13=# SELECT * FROM public.\"Employer\";
```

id_employer	id_position	family	name	patronymc	employer_passport	salary	category
111121	111114	Nagiev	Dmitriy	Vladimirovich	6716 53355	75000	1
111122	111111	Victoryia	Lazareva	Sergeevna	6728 22837	30000	1
111123	111113	Dobroslave	Shiryaev	Nurzupaevicg	6111 11111	60000	1
111124	111113	Ismail	Darzaev	Umarshapaevich	6111 195851	60000	1
111126	111112	Bezrukova	Kseniya	Sergeevna	11 124566	27000	1
111127	111112	Romanov	Mikhail	Yanovich	6666 133728	27000	1
111128	111112	Roseman	Eugeny	Yanovich	6636 135628	27000	1
111129	111115	Davidov	Vladislav	Sergeevich	1122 2222	17000	1
111125	111113	Maxim	Prihodko	Tatianovich	6111 133337	60000	1

(9 строк)

VAR13=#

После:

При проверке SELECT * FROM salary_increase(111125);

```
VAR13=# SELECT * FROM salary_increase(111125);
```

family	name	salary	category
Maxim	Prihodko	78000	2

(1 строка)

VAR13=#

```
VAR13=# SELECT * FROM public.\"Employer\";
```

id_employer	id_position	family	name	patronymc	employer_passport	salary	category
111121	111114	Nagiev	Dmitriy	Vladimirovich	6716 53355	75000	1
111122	111111	Victoryia	Lazareva	Sergeevna	6728 22837	30000	1
111123	111113	Dobroslave	Shiryaev	Nurzupaevicg	6111 11111	60000	1
111124	111113	Ismail	Darzaev	Umarshapaevich	6111 195851	60000	1
111126	111112	Bezrukova	Kseniya	Sergeevna	11 124566	27000	1
111127	111112	Romanov	Mikhail	Yanovich	6666 133728	27000	1
111128	111112	Roseman	Eugeny	Yanovich	6636 135628	27000	1
111129	111115	Davidov	Vladislav	Sergeevich	1122 2222	17000	1
111125	111113	Maxim	Prihodko	Tatianovich	6111 133337	78000	2

(9 строк)

VAR13=#

2. Создание триггеров:

2.1. Создадим универсальный триггер для логирования действий добавления, обновления и удаления данных. Предварительно создадим таблицу «logs» для этого:

```
SQL Shell (psql)
^
VAR13=# CREATE OR REPLACE FUNCTION iud()
VAR13=# RETURNS TRIGGER AS $$
VAR13$# DECLARE
VAR13$# astr varchar(30);
VAR13$# bstr varchar(100);
VAR13$# cstr varchar(254);
VAR13$# BEGIN
VAR13$# IF TG_OP = 'INSERT' THEN
VAR13$# bstr = NEW;
VAR13$# astr:= 'INSERTION';
VAR13$# cstr:= astr||bstr;
VAR13$# INSERT INTO logs(state_of, name_of_table, changes_done_to, time_of_changes) values (astr,TG_TABLE_NAME,cstr,NOW());
VAR13$# RETURN NEW;
VAR13$# ELSIF TG_OP = 'UPDATE'
VAR13$# THEN bstr = NEW;
VAR13$# astr:= 'Update user ';
VAR13$# cstr:= astr||bstr;
VAR13$# INSERT INTO logs(state_of, name_of_table, changes_done_to, time_of_changes) values (astr,TG_TABLE_NAME,cstr,NOW());
VAR13$# RETURN NEW;
VAR13$# ELSIF TG_OP = 'DELETE' THEN
VAR13$# bstr = OLD;
VAR13$# astr:='REMOVAL';
VAR13$# cstr:=astr||bstr;
VAR13$# INSERT INTO logs(state_of, name_of_table, changes_done_to, time_of_changes) values (astr,TG_TABLE_NAME,cstr,NOW());
VAR13$# RETURN OLD;
VAR13$# END IF;
VAR13$# END;
VAR13$# $$
VAR13=# LANGUAGE plpgsql;
CREATE FUNCTION
VAR13=#
```

Создадим триггер и сделаем запросы:

```
SQL Shell (psql)
VAR13=# CREATE TRIGGER tg AFTER INSERT OR UPDATE OR DELETE ON public."Employer" FOR EACH ROW EXECUTE PROCEDURE iud();
CREATE TRIGGER
VAR13=# UPDATE public."Employer" set "category" = '4' where "id_employer" = '111123';
UPDATE 1
VAR13=# INSERT INTO public."Employer"(id_employer, id_position, family, name, patronomyc, employer_passport, salary, category)
VAR13=# values ('111130', '111115', 'Vauebekov', 'Zakid', 'Victorovich', '222 8888', '16000', '1');
INSERT 0 1
VAR13=# DELETE FROM public."Employer" WHERE id_employer = '111130';
DELETE 1
VAR13=#
```

Результаты:

```
SQL Shell (psql)
VAR13=# SELECT * FROM logs;
state_of | name_of_table | changes_done_to | time_of_changes
-----|-----|-----|-----
Update user | Employer | Update user (111123,111113,Dobroslyay,Shiryaev,Nurzupaevicg,"6111 1111",60000,4) | 2022-06-16 14:42:54.856656+03
INSERTION | Employer | INSERTION(111130,111115,Vauebekov,Zakid,Victorovich,"222 8888",16000,1) | 2022-06-16 14:47:54.265647+03
REMOVAL | Employer | REMOVAL(111130,111115,Vauebekov,Zakid,Victorovich,"222 8888",16000,1) | 2022-06-16 14:48:55.993991+03
(3 rows)

VAR13=#
```

Выводы:

В ходе выполнения лабораторной работы я узнал принцип создания хранимых процедур и триггеров.