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

Отчет

по лабораторной работе «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине «**Базы данных**»

Автор: Скороходова Е.С.

Факультет: ИКТ

Группа: К32392

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Цель работы: овладеть практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

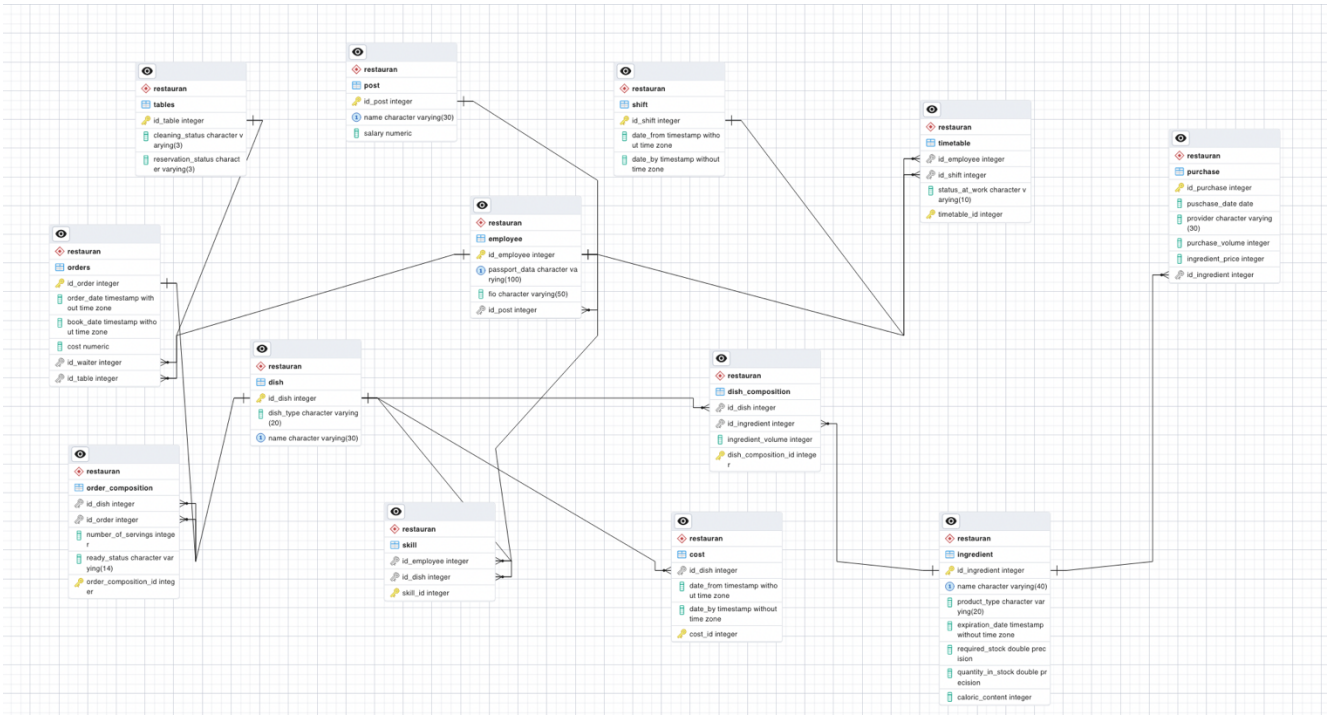
Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:
Вариант 1

- скрипты кода разработанных объектов (процедур/функций и триггера на логирование действий) и подтверждающие скриншоты работы и результатов в psql согласно индивидуальному заданию (часть 4 и 5).

Ход работы:



- Вывести сведения о заказах заданного официанта на заданную дату.

```
rest_back=#  CREATE OR REPLACE FUNCTION restauran.get_waiter_orders (emp_id INT,
rest_back(#  date_order DATE) RETURNS TABLE (id_order INT,
rest_back(#  id_waiter INT,
rest_back(#  id_table INT,
rest_back(#  cost DECIMAL) AS $$ begin return querySELECT o.id_order,
rest_back$#  o.id_waiter,
rest_back$#  o.id_table,
rest_back$#  o.cost
rest_back$# FROM restauran.orders AS o
rest_back$# WHERE o.id_waiter = emp_id
rest_back$# AND DATE(o.order_date) = date_order; end; $$ LANGUAGE 'plpgsql';
```

Вывод:

```
rest_back=# select restauran.get_waiter_orders(3, '2023-04-30');
 get_waiter_orders
-------------------
 (4,3,3,1043.85)
 (6,3,3,1542.4)
(2 rows)
```

- Выполнить расчет стоимости заданного заказа.

```
rest_back=# CREATE PROCEDURE restauran.CalculateOrderCost (order_id INT) AS $$ DECLARE order_Cost DECIMAL;
rest_back$#     BEGIN order_Cost =
rest_back$# (SELECT SUM(oc.number_of_servings * o.cost)
rest_back$# FROM restauran.order_composition oc
rest_back$# JOIN restauran.orders AS o
rest_back$# ON o.id_order = oc.id_order
rest_back$# JOIN restauran.dish d
rest_back$# ON oc.id_dish = d.id_dish
rest_back$# WHERE oc.id_order = order_id); RETURN order_Cost; END; $$ language 'plpgsql';
```

Вывод:

```
rest_back=# select restauran.calculateordercost(3);
 calculateordercost
--------------------
               1626
(1 row)
```

- Повышения оклада заданного сотрудника на 30 %.

До

```
rest_back=# SELECT *
rest_back-# FROM restauran.employee
rest_back-# JOIN restauran.post ON employee.id_post = post.id_post;
 id_employee | passport_data |            fio             | id_post | id_post |   name    | salary
-------------+---------------+----------------------------+---------+---------+-----------+--------
           1 | 4018 998445   | Иванов Иван Иванович       |       3 |       3 | Официант  |  45000
           2 | 5334 584036   | Лопаткина Инга Юрьевна     |       1 |       1 | Повар     |  60000
           3 | 3804 638490   | Петров Юрим Максимович     |       3 |       3 | Официант  |  45000
           4 | 3948 859483   | Старикова Евгения Юрьевна  |       2 |       2 | Шеф-повар | 100000
(4 rows)
```

После создания функции

```
rest_back=# CREATE OR REPLACE FUNCTION restauran.increase_salary_category(employee_id INT) RETURNS VOID AS $$
rest_back$# DECLARE
rest_back$#     current_salary DECIMAL;
rest_back$#     new_salary DECIMAL;
rest_back$# BEGIN
rest_back$#     SELECT p.salary INTO current_salary
rest_back$#     FROM restauran.employee AS e
rest_back$#     JOIN restauran.post AS p ON e.id_post = p.id_post
rest_back$#     WHERE e.id_employee = employee_id;
rest_back$#     new_salary := current_salary * 1.3;
rest_back$#     UPDATE restauran.post
rest_back$#     SET salary = new_salary
rest_back$#     WHERE id_post = (
rest_back$#         SELECT id_post
rest_back$#         FROM restauran.employee
rest_back$#         WHERE id_employee = employee_id
rest_back$#     );
rest_back$# END;
rest_back$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
rest_back=# call restauran.increase_salary_category(2);
ERROR:  restauran.increase_salary_category(integer) is not a procedure
LINE 1: call restauran.increase_salary_category(2);
             ^
HINT:  To call a function, use SELECT.
rest_back=# select restauran.increase_salary_category(2);
 increase_salary_category
--------------------------

(1 row)

rest_back=# SELECT *
FROM restauran.employee
JOIN restauran.post ON employee.id_post = post.id_post;
 id_employee | passport_data |            fio             | id_post | id_post |   name    | salary
-------------+---------------+----------------------------+---------+---------+-----------+---------
           1 | 4018 998445   | Иванов Иван Иванович       |       3 |       3 | Официант  |   45000
           2 | 5334 584036   | Лопаткина Инга Юрьевна     |       1 |       1 | Повар     | 78000.0
           3 | 3804 638490   | Петров Юрим Максимович     |       3 |       3 | Официант  |   45000
           4 | 3948 859483   | Старикова Евгения Юрьевна  |       2 |       2 | Шеф-повар |  100000
(4 rows)
```

Триггер на проверку того, что нельзя в заказ добавить неубранный стол:

```
rest_back=# CREATE FUNCTION restauran.check_clean_table() RETURNS TRIGGER AS $$
rest_back$# BEGIN
rest_back$#     IF (SELECT cleaning_status FROM restauran.tables WHERE id_table
= NEW.id_table) != 'Да' THEN
rest_back$#         RAISE EXCEPTION 'Нельзя предложить грязный стол';
rest_back$#     END IF;
rest_back$#     RETURN NEW;
rest_back$# END;
rest_back$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
rest_back=#
```

```
rest_back=# CREATE TRIGGER check_clean_table_trigger
rest_back-# BEFORE INSERT ON restauran.orders
rest_back-# FOR EACH ROW
rest_back-# EXECUTE FUNCTION restauran.check_clean_table();
CREATE TRIGGER
rest_back=#
```

```
rest_back=# insert into restauran.orders (id_order, order_date, book_date, cost,
id_waiter, id_table) values (7, '2023-05-10', null, 4024, 1, 2);
ERROR:  Нельзя предложить грязный стол
CONTEXT:  PL/pgSQL function restauran.check_clean_table() line 4 at RAISE
rest_back=#
```

Вывод

В ходе лабораторной работы я научилась создавать и использовать процедуры, функции и триггеры в базе данных PostgreSQL. Также, я поняла, что функции и процедуры в SQL недостаточно гибкие, как в ЯП.