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

Отчет

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

по дисциплине «**Проектирование и реализация баз данных**»

Автор: Нестеренко Ю. А.

Факультет: ИКТ

Группа: К32422

Преподаватель: Говорова М. М.

Дата: 20.04.2023

Санкт-Петербург 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

Практическое задание (вариант 2):

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).

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.

1 Описание предметной области

Вариант 13. БД «Ресторан»

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.

2 Схема логической модели базы данных

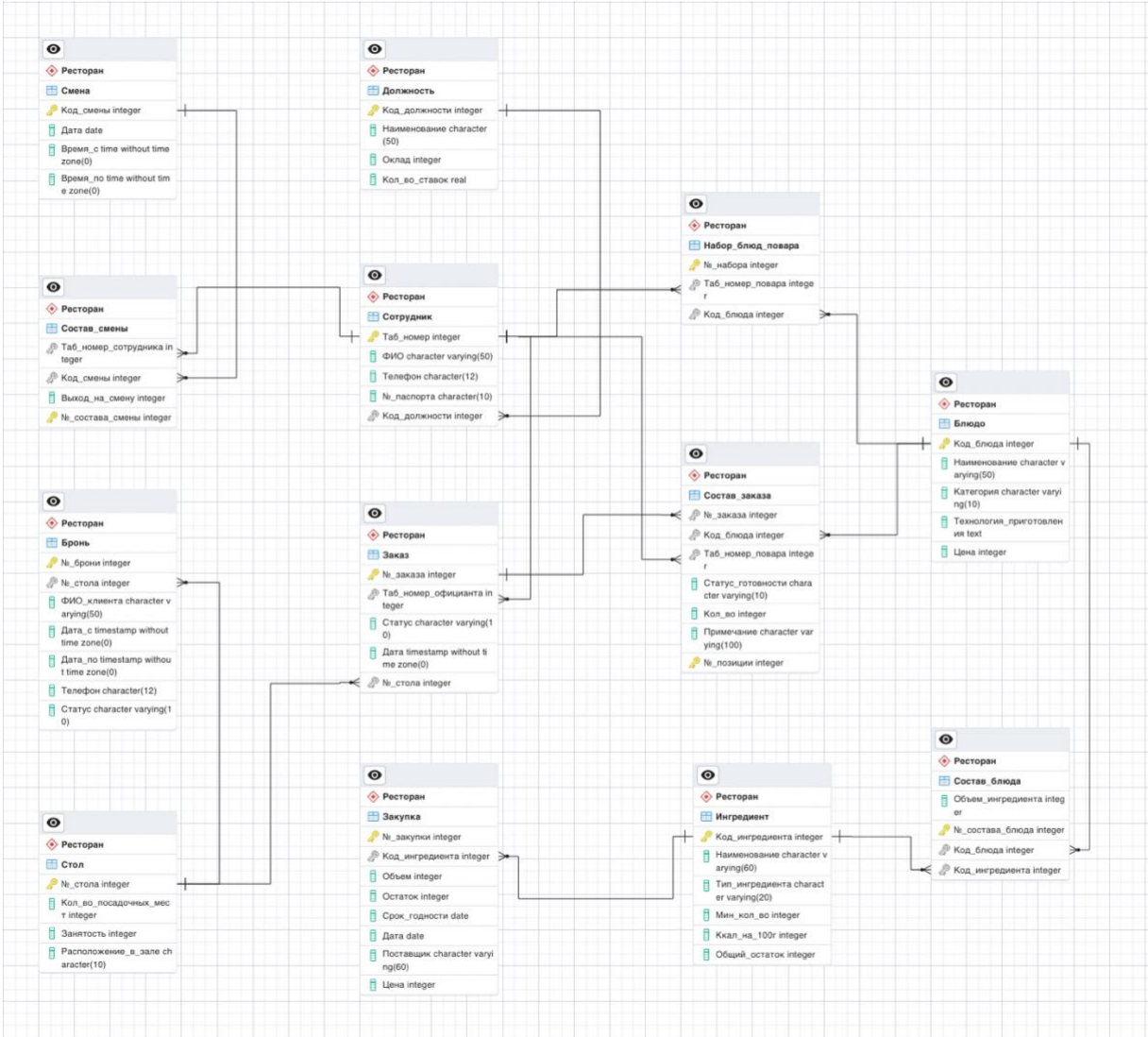

Рисунок 1 – Схема ЛМ БД в Generate ERD

3 Процедуры и функции

1. Вывести сведения о заказах заданного официанта на заданную дату.

```
CREATE FUNCTION orders_by_waiter_and_date(waiter_id integer, order_date date)
RETURNS SETOF "Ресторан"."Заказ" AS
$$
    SELECT * FROM "Ресторан"."Заказ"
    WHERE ("Таб_номер_официанта" = waiter_id)
    AND ("Дата"::date = order_date)
$$ LANGUAGE SQL;
```

```
[Restaurant=# CREATE FUNCTION orders_by_waiter_and_date(waiter_id integer, order_date date)
[Restaurant-# RETURNS SETOF "Ресторан"."Заказ" AS
[Restaurant-# $$
[Restaurant$# SELECT * FROM "Ресторан"."Заказ"
 Restaurant$# WHERE ("Таб_номер_официанта" = waiter_id) AND ("Дата"::date = order_date)
[Restaurant$# $$ LANGUAGE SQL;
 CREATE FUNCTION
[Restaurant=# SELECT * FROM orders_by_waiter_and_date(289731, '2023-03-15');
  №_заказа | Таб_номер_официанта |  Статус  |        Дата         | №_стола
----------+---------------------+----------+---------------------+---------
         6 |              289731 | Оплачен  | 2023-03-15 12:58:00 |       3
         8 |              289731 | В работе | 2023-03-15 16:48:00 |       7
(2 rows)
```

Рисунок 2 – Использование функции 1

2. Выполнить расчет стоимости заданного заказа.

```
CREATE FUNCTION order_total_price(order_num integer)
RETURNS table("№_заказа" int, "Стоимость" int) AS
$$
    SELECT order_num, SUM("Кол_во" * "Цена") AS "Цена_заказа"
    FROM "Ресторан"."Состав_заказа" JOIN "Ресторан"."Блюдо"
    ON "Состав_заказа"."Код_блюда" = "Блюдо"."Код_блюда" WHERE
    "№_заказа" = order_num
$$ LANGUAGE SQL;
```

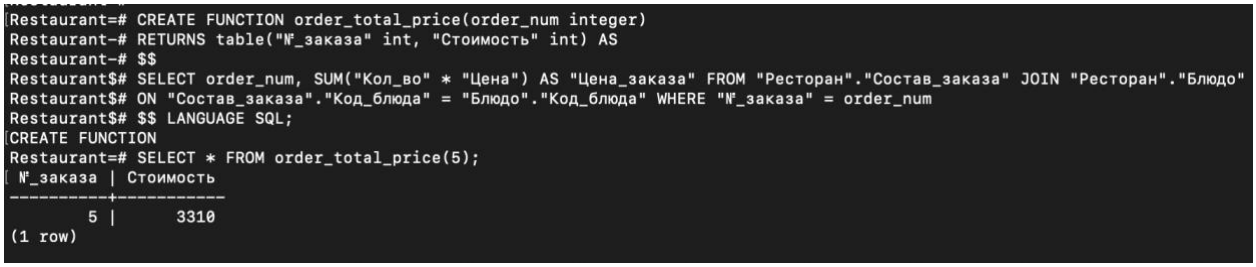

Рисунок 3 – Использование функции 2

3. Повышение оклада заданного сотрудника на 30 % при повышении его категории.

Модификация: у меня нет отдельной зарплаты каждого сотрудника - оклад относится к должности. Поэтому поменяю формулировку: пересчитать зарплаты поваров (только у них есть категории) как +30% к предыдущей по порядку категории.

```
CREATE PROCEDURE update_cook_salary_by_category()
LANGUAGE SQL
AS
$$
	UPDATE "Ресторан"."Должность" SET "Оклад" = CASE
	WHEN "Наименование" LIKE '%1 категории%' THEN
		(SELECT ROUND(1.3*"Оклад" / "Кол_во_ставок")
		FROM "Ресторан"."Должность"
		WHERE "Наименование" LIKE '%стажер%')
		* "Кол_во_ставок"
	WHEN "Наименование" LIKE '%1 категории%' THEN
		(SELECT ROUND(1.3*"Оклад")
		FROM "Ресторан"."Должность"
		WHERE "Наименование" LIKE '%1 категории%'
		AND "Кол_во_ставок" = 1) * "Кол_во_ставок"
	WHEN "Наименование" = 'Су-шеф' THEN
		(SELECT ROUND(1.3*"Оклад")
		FROM "Ресторан"."Должность"
		WHERE "Наименование" LIKE '%2 категории%'
		AND "Кол_во_ставок" = 1) * "Кол_во_ставок"
	WHEN "Наименование" = 'Шеф-повар' THEN
		(SELECT ROUND(1.3*"Оклад")
		FROM "Ресторан"."Должность"
		WHERE "Наименование" = 'Су-шеф'
		AND "Кол_во_ставок" = 1) * "Кол_во_ставок"
	ELSE "Оклад"
	END
$$;
```

```
[Restaurant=#
[Restaurant=# SELECT * FROM "Ресторан"."Должность";
[ Код_должности |                    Наименование                    | Оклад  | Кол_во_ставок
----------------+----------------------------------------------------+--------+---------------
            101 | Шеф-повар                                          | 384300 |             1
            102 | Су-шеф                                             | 219600 |             1
            103 | Повар 2 категории                                  | 136500 |             1
            104 | Повар 1 категории                                  | 100800 |             1
            105 | Повар 1 категории                                  |  50400 |           0.5
            201 | Администратор зала                                 | 117600 |             1
            202 | Хостес                                             |  88200 |             1
            203 | Бармен                                             |  82950 |             1
            204 | Официант                                           |  67410 |             1
            106 | Повар-стажер                                       |  36750 |           0.5
            205 | Официант                                           |  33705 |           0.5
(11 rows)

Restaurant=#
[Restaurant=#
[Restaurant=#
[Restaurant=# CREATE PROCEDURE update_cook_salary_by_category()
Restaurant-# LANGUAGE SQL
Restaurant-# AS
Restaurant-# $$
Restaurant$# UPDATE "Ресторан"."Должность" SET "Оклад" = CASE
Restaurant$# WHEN "Наименование" LIKE '%1 категории%' THEN
Restaurant$# (SELECT ROUND(1.3*"Оклад" / "Кол_во_ставок") FROM "Ресторан"."Должность"
Restaurant$#  WHERE "Наименование" LIKE '%стажер%') * "Кол_во_ставок"
Restaurant$# WHEN "Наименование" LIKE '%1 категории%' THEN
Restaurant$# (SELECT ROUND(1.3*"Оклад") FROM "Ресторан"."Должность"
Restaurant$#  WHERE "Наименование" LIKE '%1 категории%' AND "Кол_во_ставок" = 1) * "Кол_во_ставок"
Restaurant$# WHEN "Наименование" = 'Су-шеф' THEN
Restaurant$# (SELECT ROUND(1.3*"Оклад") FROM "Ресторан"."Должность"
Restaurant$#  WHERE "Наименование" LIKE '%2 категории%' AND "Кол_во_ставок" = 1) * "Кол_во_ставок"
Restaurant$# WHEN "Наименование" = 'Шеф-повар' THEN
Restaurant$# (SELECT ROUND(1.3*"Оклад") FROM "Ресторан"."Должность"
Restaurant$#  WHERE "Наименование" = 'Су-шеф' AND "Кол_во_ставок" = 1) * "Кол_во_ставок"
Restaurant$# ELSE "Оклад"
Restaurant$# END
Restaurant$# $$;
[CREATE PROCEDURE
Restaurant=#
[Restaurant=# CALL update_cook_salary_by_category();
[CALL
Restaurant=# SELECT * FROM "Ресторан"."Должность";
[ Код_должности |                    Наименование                    | Оклад  | Кол_во_ставок
----------------+----------------------------------------------------+--------+---------------
            101 | Шеф-повар                                          | 285480 |             1
            102 | Су-шеф                                             | 177450 |             1
            103 | Повар 2 категории                                  | 136500 |             1
            104 | Повар 1 категории                                  |  95550 |             1
            105 | Повар 1 категории                                  |  47775 |           0.5
            201 | Администратор зала                                 | 117600 |             1
            202 | Хостес                                             |  88200 |             1
            203 | Бармен                                             |  82950 |             1
            204 | Официант                                           |  67410 |             1
            106 | Повар-стажер                                       |  36750 |           0.5
            205 | Официант                                           |  33705 |           0.5
(11 rows)
```

Рисунок 4 – Использование процедуры 3

4 Модификация триггера на проверку корректности входа и выхода сотрудника из Практического задания 1 Лабораторного практикума.

```
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |     punch_time
----+-------------+--------------+--------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
(3 rows)

emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, false, now());
INSERT 0 0
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |     punch_time
----+-------------+--------------+--------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
(3 rows)

emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, now());
INSERT 0 1
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
  7 |           1 | t            | 2023-04-17 16:13:03.086169
(4 rows)

emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, '2023-04-17 16:23:03.086169');
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, '2022-04-17 16:23:03.086169');
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, false, '2022-04-17 16:23:03.086169');
INSERT 0 1
emp_time=# insert into employee(username) values ('Олег');
ERROR:  duplicate key value violates unique constraint "employee_pkey"
DETAIL:  Key (id)=(1) already exists.
emp_time=# insert into employee(username) values ('Олег');
INSERT 0 1
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (2, true, '2023-04-17 16:23:03.086169');
INSERT 0 1
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
  7 |           1 | t            | 2023-04-17 16:13:03.086169
 10 |           1 | f            | 2022-04-17 16:23:03.086169
 11 |           2 | t            | 2023-04-17 16:23:03.086169
(6 rows)

emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, '2023-04-17 16:23:03.086169');
INSERT 0 1
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
  7 |           1 | t            | 2023-04-17 16:13:03.086169
 10 |           1 | f            | 2022-04-17 16:23:03.086169
 11 |           2 | t            | 2023-04-17 16:23:03.086169
 12 |           1 | t            | 2023-04-17 16:23:03.086169
(7 rows)

emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (2, true, '2023-04-17 16:33:03.086169');
INSERT 0 0
```

Рисунок 5 – Тестирование изначальной версии триггера

«Узкие» места некорректных данных по входу и выходу:

1. Триггер не ограничивает вставку данных «задним числом».

2. Для новых сотрудников выход может записываться даже без предшествующего ему входа

3. Внутри триггерной функции сортировка по порядку вставки данных, а не по дате.

Модифицированная версия триггерной функции:

```
create or replace function fn_check_time_punch() returns trigger as $psql$
   begin
      if new.is_out_punch = (
        select tps.is_out_punch
        from time_punch tps
        where tps.employee_id = new.employee_id
        order by tps.punch_time desc limit 1
      ) or new.punch_time < (
                  select tps.punch_time
        from time_punch tps
        where tps.employee_id = new.employee_id
        order by tps.punch_time desc limit 1
                ) or (new.is_out_punch = true and not exists (
                        select * from time_punch tps
                        where tps.employee_id = new.employee_id
                      )
                )
                then
          return null;
      end if;
      return new;
   end;
$psql$ language plpgsql;
```

```
emp_time=# create or replace function fn_check_time_punch() returns trigger as $psql$
emp_time$#     begin
emp_time$#         if new.is_out_punch = (
emp_time$#           select tps.is_out_punch
emp_time$#           from time_punch tps
emp_time$#           where tps.employee_id = new.employee_id
emp_time$#           order by tps.punch_time desc limit 1
emp_time$#         ) or new.punch_time < (
emp_time$#   select tps.punch_time
emp_time$#           from time_punch tps
emp_time$#           where tps.employee_id = new.employee_id
emp_time$#           order by tps.punch_time desc limit 1
emp_time$# ) or (new.is_out_punch = true and not exists (
emp_time$#   select * from time_punch tps
emp_time$#   where tps.employee_id = new.employee_id
emp_time$# )
emp_time$# )
emp_time$# then
emp_time$#             return null;
emp_time$#         end if;
emp_time$#         return new;
emp_time$#     end;
emp_time$# $psql$ language plpgsql;
CREATE FUNCTION
```

Рисунок 6 – Модифицированная триггерная функция

```
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |     punch_time
----+-------------+--------------+---------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
(3 rows)

emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, now());
INSERT 0 1
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, '2023-04-20 16:00:00');
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, false, '2022-01-01 16:00:00');
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, '2023-04-20 16:00:00');
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, true, '2022-01-01 16:00:00');
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (2, true, now());
INSERT 0 0
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (2, false, now());
INSERT 0 1
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, false, now());
INSERT 0 1
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  5 |           1 | f            | 2021-01-01 11:50:00
 15 |           1 | t            | 2023-04-20 12:47:32.57192
 22 |           2 | f            | 2023-04-20 12:54:40.75612
 23 |           1 | f            | 2023-04-20 12:55:33.027043
(6 rows)
```

Рисунок 7 – Тестирование обновленной версии триггера

ВЫВОДЫ

В рамках данной лабораторной работы получены практические навыки создания и использования процедур, функций и триггеров в базе данных PostgreSQL. Реализованы 2 функции и 1 процедура согласно индивидуальному заданию - варианту 13 БД «Ресторан», пункт 4. Проанализирована работа триггера на проверку корректности входа и выхода сотрудника из Практического задания 1 Лабораторного практикума. Найдены «узкие» места некорректных данных по входу и выходу. Согласно выявленным недочетам, триггерная функция была модифицирована и проверена на корректных и некорректных данных. Дополнительно получен опыт взаимодействия с БД через консольный клиент SQL SHELL (psql).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лекция 2.4 Процедуры. Функции. Триггеры. 2023. (дата обращения 18.04.2023)

2. Документация PostgreSQL [Электронный ресурс] // Официальный сайт PostgreSQL. 1996–2023. URL: https://www.postgresql.org/docs/13/index.html (дата обращения: 18.04.2023).

3. Видеокурс PostgreSQL #6 | Хранимые процедуры (функции) [Электронный ресурс] // YouTube. URL: https://www.youtube.com/watch?v=8y_BFAIkvZk (дата обращения 18.04.2023)

4. Проектирование и реализация баз данных. Работа в SQL SHELL (psql). 2023. (дата обращения 19.04.2023)

5. Проектирование и реализация баз данных. Практическое занятие. Работа с триггерами и функциями. 2023. (дата обращения 19.04.2023)