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

Отчет

по лабораторной работе №5 «ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

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

Автор: Дерещук Т. Е.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание

Вариант 1 (max - 6 баллов)

1. Создать процедуры/функции 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 (max - 8 баллов)

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

2.1. 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).

2.2. Создать авторский триггер по варианту индивидуального задания.

Выполнение: Вариант 20 «Автозаправки»

Процедуры/функции согласно индивидуальному заданию (часть 4).

- *Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма).*

Query | Query History

```
1  CREATE OR REPLACE FUNCTION schema_1.GetClientPurchases(
2      client_code INTEGER,
3      purchase_date_input TIMESTAMP WITH TIME ZONE
4  )
5  RETURNS TABLE (
6      client_full_name CHARACTER VARYING(100),
7      purchase_date DATE,
8      fuel_quantity NUMERIC,
9      paid_amount NUMERIC
10 ) AS $$
11 BEGIN
12     RETURN QUERY
13     SELECT c.full_name AS client_full_name,
14            DATE(s.date_sold) AS purchase_date,
15            s.amount_sold_fuel AS fuel_quantity,
16            s.amount_sold_fuel * 2 AS paid_amount
17     FROM schema_1.customer c
18     JOIN schema_1.card ca ON c.customer_id = ca.customer_id
19     JOIN schema_1.service_1 s ON ca.card_number = s.card_number
20    WHERE c.customer_id = client_code
21      AND DATE_TRUNC('day', s.date_sold) = DATE_TRUNC('day', purchase_date_input);
22 END;
23 $$ LANGUAGE plpgsql;
24
```

Messages | Notifications | Data Output

```
CREATE FUNCTION

Query returned successfully in 80 msec.
```

Query | Query History

```
1  select * from schema_1.GetClientPurchases(7, '2023-07-21')
```

Messages | Notifications | Data Output

	client_full_name character varying	purchase_date date	fuel_quantity numeric	paid_amount numeric
1	Дмитрий Исаев	2023-07-21	4	8
2	Дмитрий Исаев	2023-07-21	5	10

- *Самый непопулярный вид топлива за прошедшую неделю.*

Query | Query History

```
1  CREATE OR REPLACE FUNCTION schema_1.LeastPopularFuelLastWeek() RETURNS TABLE (
2      popular_grade_fuel VARCHAR(50),
3      sales_count NUMERIC
4  ) AS $$
5  BEGIN
6      RETURN QUERY
7      SELECT brand, sales_sum FROM
8   (SELECT f.brand, SUM(s.amount_sold_fuel) AS sales_sum
9   FROM schema_1.fuel f
10  RIGHT JOIN schema_1.service_1 s ON f.fuel_code = s.sold_fuel_code
11  WHERE s.date_sold >= (CURRENT_DATE - INTERVAL '1 year')
12  GROUP BY f.brand) inn
13  ORDER BY sales_sum LIMIT 1;
14 END;
15 $$ LANGUAGE plpgsql;
16
```

Messages | Notifications | Data Output

```
CREATE FUNCTION

Query returned successfully in 491 msec.
```

Query | Query History

```
1  select * from schema_1.LeastPopularFuelLastWeek();
```

Messages | Notifications | Data Output

	popular_grade_fuel character varying	sales_count numeric
1	Газ	10

- *Количество видов топлива, поставляемых каждой фирмой-поставщиком.*

Query | Query History

```
1  CREATE OR REPLACE FUNCTION schema_1.CountFuelTypesPerSupplier()
2  RETURNS TABLE (
3      supplier_company_id bigint,
4      supplier_company_title character varying(100),
5      fuel_types_count integer
6  ) AS $$
7  BEGIN
8      RETURN QUERY
9      SELECT fsc.company_code AS supplier_company_id,
10            fsc.company_title AS supplier_company_title,
11            COUNT(DISTINCT fsc.fuel_type)::integer AS fuel_types_count
12     FROM schema_1.fuel_supplier_company fsc
13     GROUP BY fsc.company_code, fsc.company_title;
14 END;
15 $$ LANGUAGE plpgsql;
16
```

Messages | Notifications | Data Output

```
CREATE FUNCTION

Query returned successfully in 47 msec.
```

Query | Query History

```
1  select * from schema_1.CountFuelTypesPerSupplier();
```

Messages | Notifications | Data Output

	supplier_company_id bigint	supplier_company_title character varying	fuel_types_count integer
1	101	ОАО "ГазНефть"	1
2	123	ОАО "Нефтегаз"	1
3	210	ОАО "Нефть-Газ"	1
4	222	ОАО "Нефтепродукты-Газ"	1
5	333	ОАО "Топливо"	1
6	444	ОАО "Нефть-Продукты"	1
7	456	ОАО "Нефтепродукт"	1
8	555	ОАО "Топливо-Продукт"	1
9	777	ОАО "Продукты-Нефть"	1
10	789	ОАО "Нефть-Газ-Продукт"	1

Создать тригер

Для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер

Создать авторский триггер по варианту индивидуального задания

Вывод

Мы овладели практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL, научились писать и тестировать их в консоли SQL Shell.