**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: Сачук А.А.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**Оглавление**

[Цель работы 3](#_Toc153720875)

[Практическое задание 3](#_Toc153720876)

[Выполнение: Вариант 20 «Автозаправки» 3](#_Toc153720877)

[Процедуры/функции согласно индивидуальному заданию (часть 4). 3](#_Toc153720878)

[ Создать триггер 6](#_Toc153720879)

[ Создать авторский триггер по варианту индивидуального задания. 7](#_Toc153720880)

[Вывод : 8](#_Toc153720881)

# 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).

CREATE OR REPLACE FUNCTION GetClientPurchases(

client\_code INTEGER,

purchase\_date\_input TIMESTAMP WITH TIME ZONE

)

RETURNS TABLE (

client\_full\_name CHARACTER VARYING(500),

purchase\_date DATE,

fuel\_quantity DOUBLE PRECISION,

paid\_amount DOUBLE PRECISION

) AS $$

BEGIN

RETURN QUERY

SELECT c.full\_name AS client\_full\_name,

DATE(b.date\_of\_sale) AS purchase\_date,

b.quantity AS fuel\_quantity,

b.price AS paid\_amount

FROM "Refill Schema".client c

JOIN "Refill Schema".card\_account ca ON c.code\_client = ca.code\_client

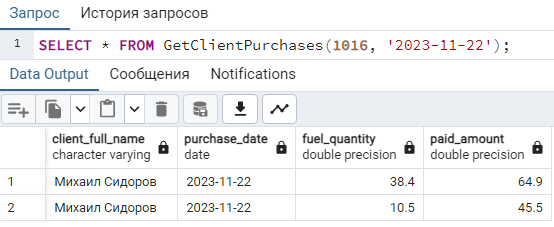
JOIN "Refill Schema".buy b ON ca.code\_card = b.code\_card

WHERE c.code\_client = client\_code

AND DATE\_TRUNC('day', b.date\_of\_sale) = DATE\_TRUNC('day', purchase\_date\_input);

END;

$$ LANGUAGE plpgsql;



* Количество видов топлива, поставляемых каждой фирмой-поставщиком.

CREATE OR REPLACE FUNCTION CountFuelTypesPerSupplier()

RETURNS TABLE (

supplier\_company\_id integer,

supplier\_company\_title character varying(250),

fuel\_types\_count integer

) AS $$

BEGIN

RETURN QUERY

SELECT ps.code\_supplier\_company AS supplier\_company\_id,

fsc.title AS supplier\_company\_title,

COUNT(DISTINCT ps.code\_fuel)::integer AS fuel\_types\_count

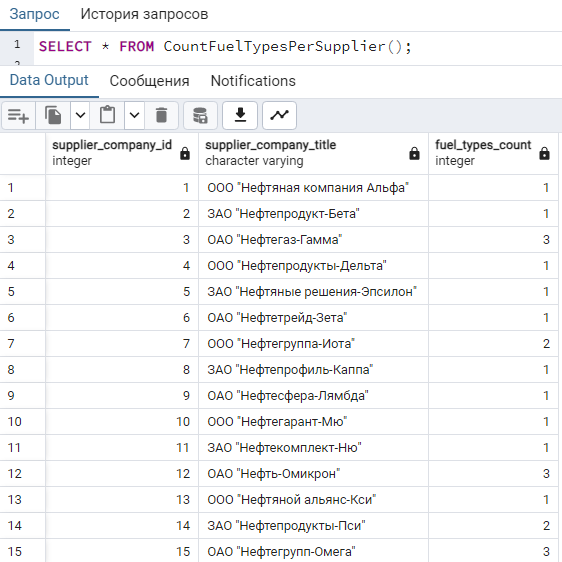
FROM "Refill Schema".produces ps

JOIN "Refill Schema".fuel\_supplier\_company fsc ON ps.code\_supplier\_company = fsc.code\_supplier\_company

GROUP BY ps.code\_supplier\_company, fsc.title;

END;

$$ LANGUAGE plpgsql;



* Самый непопулярный вид топлива за прошедшую неделю.

CREATE OR REPLACE FUNCTION LeastPopularFuelLastWeek() RETURNS TABLE (

popular\_grade\_fuel VARCHAR(50),

sales\_count DOUBLE PRECISION

) AS $$

BEGIN

RETURN QUERY

SELECT grade\_fuel, COALESCE(sales\_sum, 0.0) AS sales\_count

FROM (

SELECT f.grade\_fuel, SUM(COALESCE(b.quantity, 0.0)) AS sales\_sum,

RANK() OVER (ORDER BY SUM(COALESCE(b.quantity, 0.0)) DESC) AS sales\_rank

FROM "Refill Schema".fuel f

LEFT JOIN "Refill Schema".buy b ON f.code\_fuel = b.code\_fuel\_sold

AND b.date\_of\_sale >= (CURRENT\_DATE - INTERVAL '1 week')

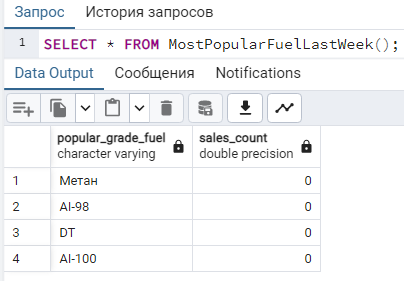
GROUP BY f.grade\_fuel

) ranked\_sales

WHERE sales\_sum = 0.0;

END;

$$ LANGUAGE plpgsql;



## Создать триггер

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

CREATE TABLE "Refill Schema".buy\_log (

log\_id SERIAL PRIMARY KEY,

action\_type VARCHAR(10),

action\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

table\_name VARCHAR(100),

key\_column\_name VARCHAR(100),

key\_column\_value TEXT,

user\_name VARCHAR(100)

);

CREATE OR REPLACE FUNCTION log\_refill\_schema\_buy()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

INSERT INTO "Refill Schema".buy\_log (action\_type, table\_name, key\_column\_name, key\_column\_value, user\_name)

VALUES ('INSERT', TG\_TABLE\_NAME, 'code\_buy', NEW.code\_buy::TEXT, current\_user);

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO "Refill Schema".buy\_log (action\_type, table\_name, key\_column\_name, key\_column\_value, user\_name)

VALUES ('UPDATE', TG\_TABLE\_NAME, 'code\_buy', NEW.code\_buy::TEXT, current\_user);

ELSIF TG\_OP = 'DELETE' THEN

INSERT INTO "Refill Schema".buy\_log (action\_type, table\_name, key\_column\_name, key\_column\_value, user\_name)

VALUES ('DELETE', TG\_TABLE\_NAME, 'code\_buy', OLD.code\_buy::TEXT, current\_user);

END IF;

RETURN NULL;

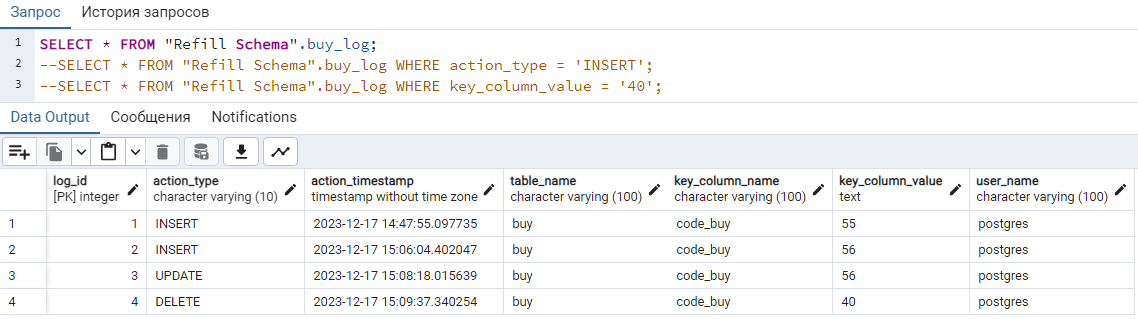
END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_log\_refill\_schema\_buy

AFTER INSERT OR UPDATE OR DELETE ON "Refill Schema".buy

FOR EACH ROW EXECUTE FUNCTION log\_refill\_schema\_buy();



## Создать авторский триггер по варианту индивидуального задания.

триггер, который будет автоматически обновлять дату продажи (date\_of\_sale) на текущую дату при вставке новой записи в таблицу.

CREATE OR REPLACE FUNCTION update\_date\_of\_sale()

RETURNS TRIGGER AS $$

BEGIN

NEW.date\_of\_sale = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

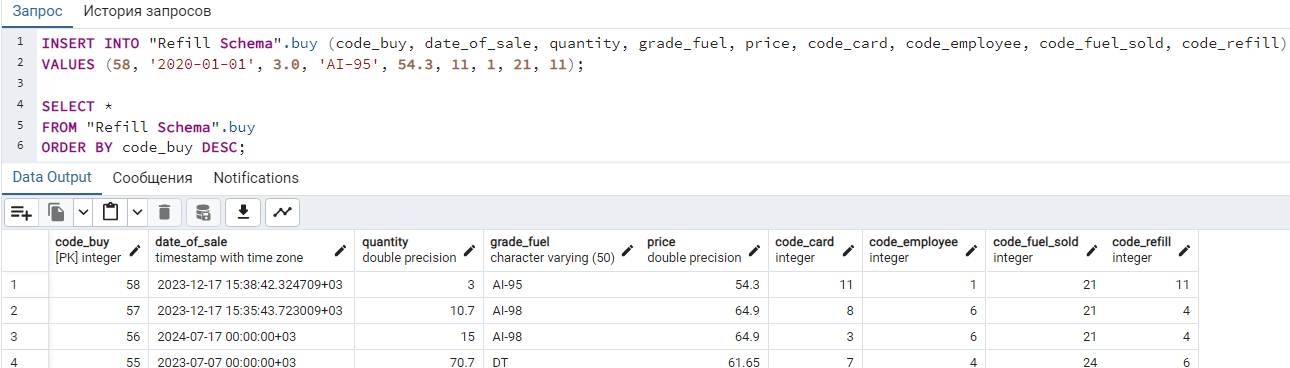
$$ LANGUAGE plpgsql;

у

CREATE TRIGGER trg\_update\_date\_of\_sale

BEFORE INSERT ON "Refill Schema".buy

FOR EACH ROW EXECUTE FUNCTION update\_date\_of\_sale();



# 

# 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.