

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»**

Факультет инфокоммуникационных технологий

Дисциплина:

«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

Выполнил:

студент группы К32392

Тишалович Леонид

Михайлович

(подпись)

Проверил(а):

Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург

2023 г.

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 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.

Выполнение:

Индивидуальное задание БД «Оптовая база»

Процедуры\функции:

1. для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.
2. для расчета стоимости всех партий товаров, проданных за прошедшие сутки.

Процедуры\функции:

1) Функция для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.

```
CREATE OR REPLACE FUNCTION reduce_price(days_limit INT,
reduction_percent REAL)
RETURNS VOID AS $$
BEGIN
    UPDATE shipment_content
    SET price = price * (1 - reduction_percent / 100)
    FROM shipment
    WHERE shipment.shipment_id = shipment_content.shipment_id AND
    current_date - shipment.invoice_creation_date > days_limit;
END;
$$ LANGUAGE plpgsql;
```

Рис. 1 – Функция №1

	shipment_content_id [PK] bigint	shipment_id bigint	product_id bigint	remains integer	amount_of_goods integer	expiration_date date	price integer
1	1	1	1	10	20	2023-06-17	100

Рис. 2 – До выполнения функции №1

	shipment_content_id [PK] bigint	shipment_id bigint	product_id bigint	remains integer	amount_of_goods integer	expiration_date date	price integer
1	1	1	1	10	20	2023-06-17	90

Рис. 3 - После выполнения функции №1

2) Функция для расчета стоимости всех партий товаров, проданных за прошедшие сутки.

```
CREATE OR REPLACE FUNCTION total_sales_last_day()
RETURNS DECIMAL AS $$
DECLARE
    total DECIMAL;
BEGIN
    SELECT SUM(purchase_content.price *
purchase_content.amount_of_goods) INTO total
    FROM purchase_content
    JOIN purchase ON purchase.purchase_id =
purchase_content.purchase_id
    WHERE purchase.invoice_creation_date = current_date - interval '1
day';
    RETURN total;
END;
$$ LANGUAGE plpgsql;
```

Рис. 4 – Функция №2


	total_sales_last_day 
	numeric
1	450

Рис. 5 – Результат выполнения функции №2

Создаем триггер для логирования событий INSERT DELETE UPDATE

Создадим табличку для записи логирования:

```
CREATE TABLE log_table (
    log_id BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    table_name VARCHAR(100),
    operation VARCHAR(10),
    old_data TEXT,
    new_data TEXT,
    log_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Создание функций и триггеров:

```
CREATE OR REPLACE FUNCTION product_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log_table (table_name, operation, new_data)
    VALUES ('product', 'INSERT', row_to_json(NEW)::text);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER product_insert_log
AFTER INSERT ON product
FOR EACH ROW EXECUTE FUNCTION product_insert_trigger();

CREATE OR REPLACE FUNCTION product_update_trigger()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log_table (table_name, operation, old_data, new_data)
    VALUES ('product', 'UPDATE', row_to_json(OLD)::text,
row_to_json(NEW)::text);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER product_update_log
AFTER UPDATE ON product
FOR EACH ROW EXECUTE FUNCTION product_update_trigger();

CREATE OR REPLACE FUNCTION product_delete_trigger()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log_table (table_name, operation, old_data)
    VALUES ('product', 'DELETE', row_to_json(OLD)::text);
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER product_delete_log
AFTER DELETE ON product
FOR EACH ROW EXECUTE FUNCTION product_delete_trigger();
```

log_id [PK] bigint	table_name character varying (100)	operation character varying (10)	old_data text	new_data text	log_date timestamp without time zone
1	product	INSERT	[null]	("product_id":2;"product_type":"Test";"product_name":"Test product";"unit_of_measurement_id":1)	2023-05-28 16:55:12.89545
2	product	UPDATE	("product_id":2;"product_type":"Test";"product_name":"Test product";"unit_of_measurement_id":1)	("product_id":2;"product_type":"Test";"product_name":"Updated product";"unit_of_measurement_id":1)	2023-05-28 16:55:22.642038
3	product	DELETE	("product_id":2;"product_type":"Test";"product_name":"Updated product";"unit_of_measurement_id":1)	[null]	2023-05-28 16:55:31.398585

Рис. 6 - Таблица с логами

Новый триггер:

```
CREATE OR REPLACE FUNCTION log_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO log_table (table_name, operation, old_data)
        VALUES (TG_TABLE_NAME, TG_OP, row_to_json(OLD)::text);
        RETURN OLD;
    ELIF (TG_OP = 'UPDATE') THEN
        INSERT INTO log_table (table_name, operation, old_data,
new_data)
        VALUES (TG_TABLE_NAME, TG_OP, row_to_json(OLD)::text,
row_to_json(NEW)::text);
        RETURN NEW;
    ELIF (TG_OP = 'INSERT') THEN
        INSERT INTO log_table (table_name, operation, new_data)
        VALUES (TG_TABLE_NAME, TG_OP, row_to_json(NEW)::text);
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_changes
AFTER INSERT OR UPDATE OR DELETE ON product
FOR EACH ROW EXECUTE FUNCTION log_trigger();
```

Выводы:

В результате работы были изучены и применены различные функции и процедуры в PostgreSQL, а также разработан триггер для логирования операций INSERT, UPDATE и DELETE.