

Лабораторна робота №4

Виконав студент групи МІТ-31
Тимохін Роман Миколайович

Тема: Розширення можливостей PostgreSQL: користувацькі типи, функції та тригери

Мета роботи: Закріпити знання з розширюваності PostgreSQL. Навчитися створювати користувацькі типи даних. Реалізувати власну користувацьку функцію або агрегат. Створити тригери для логування змін у базі даних. Автоматично оновлювати пов'язані таблиці чи заповнювати значення. Оновити діаграму бази даних відповідно до виконаних завдань. Перевірити коректність роботи реалізованих об'єктів через виконання тестових SQL-запитів.

Користувацький тип даних

створюємо ENUM тип для статусу замовлення

```
-- Користувацький тип статусу замовлення
CREATE TYPE order_status AS ENUM ('Очікується', 'Оплачено',
'Скасовано', 'Виконано');

-- Зміна поля у таблиці orders
ALTER TABLE orders
ALTER COLUMN status DROP DEFAULT,
ALTER COLUMN status TYPE order_status USING status::order_status,
ALTER COLUMN status SET DEFAULT 'Очікується';
```

♦ Це дозволяє жорстко обмежити список допустимих статусів та уникнути помилок введення.

Користувачька функція або агрегат

Функція: підрахунок середнього рейтингу товару

```
CREATE OR REPLACE FUNCTION get_average_rating(prod_id INT)
RETURNS NUMERIC AS $$
DECLARE
    avg_rating NUMERIC;
BEGIN
    SELECT AVG(rating) INTO avg_rating
    FROM reviews
    WHERE product_id = prod_id;

    RETURN COALESCE(avg_rating, 0);
END;
$$ LANGUAGE plpgsql;
```

Тестовий виклик:

```
SELECT product_name, get_average_rating(product_id) AS avg_rating
FROM products;
```

Запрос История запросов

```
1 SELECT product_name, get_average_rating(product_id) AS avg_rating
2 FROM products;
3
```

Data Output Сообщения Notifications

	product_name character varying (100)	avg_rating numeric
1	Наушники	5.0000000000000000
2	Ноутбук	4.0000000000000000
3	Футболка	0
4	Книга "Clean Code"	3.0000000000000000
5	Наушники	5.0000000000000000
6	Ноутбук	4.0000000000000000
7	Футболка	0
8	Книга "Clean Code"	0
9	Смартфон Samsung A55	0
10	Наушники JBL	0
11	Футболка Nike	0
12	Книга "Мистецтво війни"	0
13	Книга "Clean Code"	0
14	Штани Adidas	0

Тригери для логування та оновлення

Логування змін у таблиці products

```
-- Створення таблиці логів
CREATE TABLE product_log (
    log_id SERIAL PRIMARY KEY,
    action_type VARCHAR(10),
    product_id INT,
    old_price NUMERIC,
    new_price NUMERIC,
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Функція тригера:

```
CREATE OR REPLACE FUNCTION log_product_update()  
RETURNS TRIGGER AS $$  
BEGIN  
    IF TG_OP = 'UPDATE' THEN  
        INSERT INTO product_log(action_type, product_id, old_price,  
new_price)  
        VALUES ('UPDATE', NEW.product_id, OLD.price, NEW.price);  
    ELSIF TG_OP = 'DELETE' THEN  
        INSERT INTO product_log(action_type, product_id, old_price)  
        VALUES ('DELETE', OLD.product_id, OLD.price);  
    END IF;  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

Тригер:

```
CREATE TRIGGER trg_product_update  
AFTER UPDATE OR DELETE ON products  
FOR EACH ROW  
EXECUTE FUNCTION log_product_update();
```

Автоматичне оновлення: сума замовлення

```
-- Додамо стовпець суми  
ALTER TABLE orders ADD COLUMN total_amount NUMERIC DEFAULT 0;  
  
-- Функція, яка оновлює суму  
CREATE OR REPLACE FUNCTION update_order_total()  
RETURNS TRIGGER AS $$  
BEGIN  
    UPDATE orders  
    SET total_amount = (  
        SELECT SUM(p.price * oi.quantity)  
        FROM order_items oi  
        JOIN products p ON oi.product_id = p.product_id  
        WHERE oi.order_id = NEW.order_id  
    );  
END;
```

```
)  
WHERE order_id = NEW.order_id;  
  
RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

♦ Тригер:

```
CREATE TRIGGER trg_update_total  
AFTER INSERT OR UPDATE ON order_items  
FOR EACH ROW  
EXECUTE FUNCTION update_order_total();
```

Перевірка роботи (Тестові запити)

```
-- 1. Вивести всі замовлення з підрахованою сумою  
SELECT order_id, total_amount FROM orders;  
  
-- 2. Вивести лог змін по товарах  
SELECT * FROM product_log ORDER BY changed_at DESC;  
  
-- 3. Змінити ціну товару для тесту  
UPDATE products SET price = price + 100 WHERE product_id = 1;  
  
-- 4. Перевірити середній рейтинг  
SELECT product_name, get_average_rating(product_id) FROM products;
```

Наприклад:

запрос история запросов

```
1  -- 1. Вивести всі замовлення з підрахованою сумою
2  SELECT order_id, total_amount FROM orders;
```

Data Output Сообщения Notifications

	order_id [PK] integer	total_amount numeric
1	1	0
2	2	0
3	3	0
4	4	0

Оновлена діаграма буде файлом pdf на github.

Висновок:

У ході виконання лабораторної роботи №4 було поглиблено розуміння принципів побудови розширених структур бази даних на прикладі інформаційної системи інтернет-магазину. Реалізовано користувацький тип даних (ENUM) для зберігання статусів замовлень, що забезпечує цілісність і передбачуваність даних. Створено користувацьку функцію для розрахунку середнього рейтингу товарів, яка підвищує гнучкість у вибірках. Запроваджено механізм логування змін у таблиці товарів за допомогою тригерів, що покращує контроль над змінами. Додано автоматичне оновлення загальної суми замовлення при зміні його вмісту. Оновлена ER-діаграма відображає всі нові сутності та зв'язки. Застосовані підходи підвищують продуктивність, надійність та масштабованість бази даних.