Лабораторна робота №4

З дисципліни: Бази даних та інформаційні системи
Студента групи МІТ-31: Полюховича А.І

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.

Завдання:

1. Створення користувацького типу даних
   - Для початку потрібно створити новий тип order_status для статусів замовлень та додати новий стовпець status у таблицю Orders.

```
-- Створення користувацького типу ENUM
CREATE TYPE order_status AS ENUM ('pending', 'shipped', 'delivered', 'canceled');

-- Додавання нового стовпця в таблицю Orders
ALTER TABLE Orders ADD COLUMN status order_status DEFAULT 'pending';
```

Він нам дозволяє нам працювати зі статусами замовлень, використовуючи строго визначені значення.
Результат:

	orderid [PK] integer	customerid integer	orderdate date	totalamount numeric	status order_status
1	38	8	2024-02-01	25500	pending
2	39	9	2024-02-02	15000	pending
3	40	5	2024-02-03	8000	pending
4	41	4	2024-02-04	5000	pending
5	42	5	2024-02-05	12000	pending
6	43	6	2024-02-06	3500	pending
7	44	7	2024-02-07	9000	pending

2. Створення користувацької функції або агрегату
   - Наступним кроком ми реалізуємо функцію calculate_average_order_amount(), яка обчислює середню суму замовлення у таблиці Orders.

```sql
-- Створення функції для обчислення середньої суми замовлення
CREATE OR REPLACE FUNCTION calculate_average_order_amount()
RETURNS NUMERIC AS
'
    SELECT AVG(TotalAmount) FROM Orders;
'
LANGUAGE SQL;

SELECT calculate_average_order_amount();
```

Данна функція повертає середнє значення поля TotalAmount у таблиці Orders.

Результат:

	calculate_average_order_amount numeric
1	11142.8571428571428571

3. Створення тригерів для логування змін та автоматичного оновлення пов’язаних таблиць
   - Ми створюємо таблицю Order_Log для збереження змін у замовленнях, реалізуємо тригерну функцію, яка буде фіксувати операції INSERT, UPDATE, DELETE та прив’язуємо тригер до таблиці Orders.

```sql
-- Створення таблиці для логування змін у замовленнях (змінений тип operation)
CREATE TABLE Order_Log (
    log_id SERIAL PRIMARY KEY,
    order_id INT,
    operation VARCHAR(10), -- Тепер зберігає 'INSERT', 'UPDATE' або 'DELETE'
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Створення тригерної функції для логування змін у Orders
CREATE OR REPLACE FUNCTION log_order_changes() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO Order_Log (order_id, operation)
    VALUES (NEW.OrderID, TG_OP);  -- TG_OP містить 'INSERT', 'UPDATE' або 'DELETE
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Створення тригера для відстеження змін у Orders
CREATE TRIGGER track_order_changes
AFTER INSERT OR UPDATE OR DELETE ON Orders
FOR EACH ROW
EXECUTE FUNCTION log_order_changes();


-- Перевірка результату
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)
VALUES (9, '2024-03-12', 5000);

SELECT * FROM Order_Log;
```

Результат:

	log_id [PK] integer	order_id integer	operation character varying (10)	changed_at timestamp without time zone
1	1	47	INSERT	2025-03-12 20:31:23.620739

4. Оновлення діаграми бази даних
   - Коли додається новий запис у OrderDetails, потрібно зменшити кількість товару на складі у таблиці Product, тому ми створимо тригерну функцію update_stock_quantity(), яка зменшуватиме stockquantity у Product та додамо тригер update_stock, що викликатиме цю функцію після додавання нового замовлення.

```
-- Створення тригерної функції для оновлення залишку товарів
CREATE OR REPLACE FUNCTION update_stock_quantity() RETURNS TRIGGER AS $$
BEGIN
    UPDATE Product
    SET stockquantity = stockquantity - NEW.Quantity
    WHERE ProductID = NEW.ProductID;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Створення тригера, що зменшує залишок товару після додавання запису в OrderDetails
CREATE TRIGGER update_stock
AFTER INSERT ON OrderDetails
FOR EACH ROW
EXECUTE FUNCTION update_stock_quantity();
--Перевірка чи працює тригер
INSERT INTO OrderDetails (OrderID, ProductID, Quantity, Price)
VALUES (41, 5, 2, 1500); -- Купили 2 одиниці товару з ProductID = 5
-- перевірка чи оновилася кількість товару
SELECT * FROM Product WHERE ProductID = 5;
```

   Коли додається новий запис у OrderDetails, тригер автоматично зменшує stockquantity у Product, наприклад, якщо було 50 товарів і хтось купив 2, то після замовлення залишиться 48.
   Результат:

	productid [PK] integer	name character varying (255)	category character varying (100)	price numeric	stockquantity integer
1	5	Мишка	Комп`ютерні аксесуари	500	48

5. Перевірка роботи

```
--Перевірка статусів замовлень:
SELECT OrderID, status FROM Orders;
```

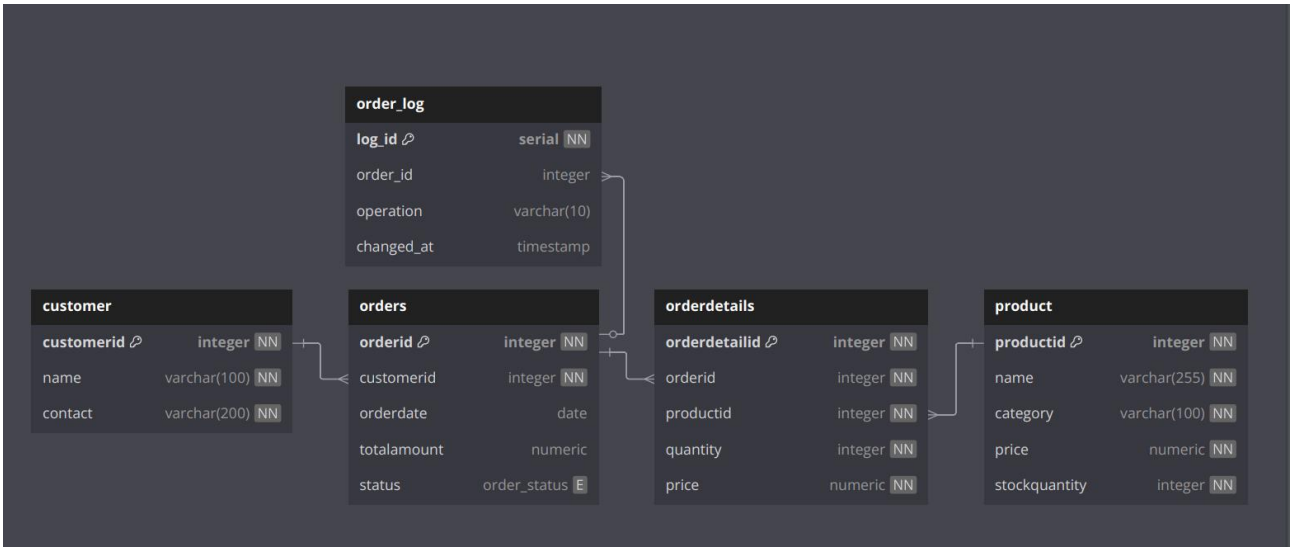
	orderid [PK] integer	status order_status
1	38	pending
2	39	pending
3	40	pending
4	41	pending
5	42	pending
6	43	pending
7	44	pending
8	47	pending

```
--Перевірка лог змін у Order_Log:
SELECT * FROM Order_Log;
```

	log_id [PK] integer	order_id integer	operation character varying (10)	changed_at timestamp without time zone
1	1	47	INSERT	2025-03-12 20:31:23.620739

```
--Перевірка залишку продукту товарів:
SELECT * FROM Product;
```

	productid [PK] integer	name character varying (255)	category character varying (100)	price numeric	stockquantity integer
1	4	Ноутбук	Електроніка	25000	10
2	6	Смартфон	Електроніка	15000	20
3	7	Клавіатура	Комп`ютерні аксесуари	1200	30
4	8	Монітор	Електроніка	8000	15
5	9	Навушники	Аудіотехніка	2000	25
6	10	Принтер	Офісна техніка	7000	10
7	11	Флеш-накопичувач	Накопичувачі	600	40
8	12	Веб-камера	Комп`ютерні аксесуари	1500	20
9	13	Геймпад	Комп`ютерні аксесуари	2500	15
10	14	Мікрофон	Аудіотехніка	3000	12
11	15	Зовнішній жорсткий диск	Накопичувачі	5000	18
12	16	Маршрутизатор	Мережеве обладнання	3500	22
13	17	Смарт-годинник	Електроніка	9000	8
14	5	Мишка	Комп`ютерні аксесуари	500	48



Висновок:

Під час лабораторної роботи було реалізовано користувацькі типи даних, користувацьку функцію та тригер для відстеження змін у таблиці. Використання цих засобів SQL допомагає пришвидшити роботу з базами даних та дозволяє краще контролювати які саме дані зберігаються та як вони змінюються для запобігання помилок та підтримки стабільної роботи системи.