ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №4

Тема: «Вивчення поняття запити мови SQL»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-23

2025

Тема: «Вивчення поняття запити мови SQL».

Мета роботи: Вивчення створення і використання запитів мови SQL.

Завдання лабораторної роботи (Варіант 14):

1. Написати запит, який виводить перелік запчастин до певного механізму.

2. Написати запит, який виводить перелік механізмів та загальну кількість запчастин до кожного з них.

3. Написати запит, який виводить назву(-ви) механізму та кількість запчастин для механізму(-мів) з найбільшою кількістю запчастин.

Хід роботи

У рамках виконання лабораторної роботи було створено та виконано три SQL-запити для бази даних PartsTradeDB, яка моделює торгівлю запчастинами. Запити були спрямовані на аналіз даних у таблицях Mechanism і Part. Для тестування запитів використано тестові дані, додані раніше. Зокрема, таблиця Mechanism містить 5 механізмів (наприклад, "Engine", "Brakes"), а таблиця Part — 5 запчастин, кожна з яких пов'язана з одним механізмом через масив Mechanism_IDs. Нижче наведено детальний опис виконання кожного запиту.

1. Перший запит: Перелік запчастин до певного механізму

Перший запит мав на меті вивести перелік запчастин для механізму з ID_Mechanism = 1 ("Engine"). Для цього використано з'єднання таблиць Part і Mechanism за допомогою оператора ANY для порівняння ID_Mechanism із масивом Mechanism_IDs, а також фільтрацію за ID_Mechanism = 1. Результати відсортовано за ціною у спадному порядку.

Query | Query History

```sql
SELECT
    p.Name AS Назва_Запчастини,
    p.Serial_Number AS Серійний_Номер,
    p.Price AS Ціна,
    m.Name AS Назва_Механізму
FROM Part p
JOIN Mechanism m ON m.ID_Mechanism = ANY(p.Mechanism_IDs)
WHERE m.ID_Mechanism = 1
ORDER BY p.Price DESC;
```

Data Output | Messages | Notifications

	Назва_Запчастини character varying (100)	Серійний_Номер character varying (50)	Ціна numeric (10,2)	Назва_Механізму character varying (100)
1	Hybrid Battery	PART101	820.00	Engine
2	Brake Pads	PART001	150.50	Engine

2. Другий запит: Перелік механізмів та загальна кількість запчастин

Другий запит мав вивести всі механізми та кількість пов’язаних із ними запчастин. Використано LEFT JOIN для включення всіх механізмів, агрегатну функцію COUNT для підрахунку запчастин і групування за назвою механізму. Результати відсортовано за кількістю запчастин у спадному порядку.

Query | Query History

```
SELECT
    m.Name AS Назва_Механізму,
    COUNT(p.ID_Part) AS Кількість_Запчастин
FROM Mechanism m
LEFT JOIN Part p ON m.ID_Mechanism = ANY(p.Mechanism_IDs)
GROUP BY m.Name
ORDER BY Кількість_Запчастин DESC;
```

Data Output | Messages | Notifications

	Назва_Механізму character varying (100)	Кількість_Запчастин bigint
1	Suspension	2
2	Engine	2
3	Brake System	2
4	Transmission	2
5	Steering	1
6	ABS Module	0
7	Hybrid Engine	0
8	Rack Steering	0
9	Hydraulic Suspension	0
10	Drum Brakes	0
11	Turbo System	0
12	CVT Transmission	0

3. Третій запит: Механізми з найбільшою кількістю запчастин

Третій запит мав визначити механізми з найбільшою кількістю запчастин. Для цього використано CTE (WITH) для обчислення кількості запчастин, віконну функцію RANK для визначення максимуму та фільтрацію за рангом 1. Результати відсортовано за назвою механізму.

Для аналізу продуктивності першого запиту виконано EXPLAIN ANALYZE. Результати запитів і знімки екрану додано до звіту. Усі запити виконано успішно, що дозволило проаналізувати зв’язки між механізмами та запчастинами.

Query | Query History

```sql
WITH MechanismCounts AS (
    SELECT
        m.Name AS Назва_Механізму,
        COUNT(p.ID_Part) AS Кількість_Запчастин,
        RANK() OVER (ORDER BY COUNT(p.ID_Part) DESC) AS Ранг
    FROM Mechanism m
    LEFT JOIN Part p ON m.ID_Mechanism = ANY(p.Mechanism_IDs)
    GROUP BY m.Name
)
SELECT
    Назва_Механізму,
    Кількість_Запчастин
FROM MechanismCounts
WHERE Ранг = 1
ORDER BY Назва_Механізму;
```

Data Output | Messages | Notifications

	Назва_Механізму character varying (100)	Кількість_Запчастин bigint
1	Brake System	2
2	Engine	2
3	Suspension	2
4	Transmission	2

Висновок: У рамках виконання лабораторної роботи було створено та виконано три SQL-запити для бази даних `PartsTradeDB`, що моделює торгівлю запчастинами. Запити дозволили проаналізувати зв’язки між таблицями `Mechanism` і `Part`, зокрема: виведено перелік запчастин для механізму з `ID_Mechanism = 1` ("Engine"), визначено кількість запчастин для кожного механізму та ідентифіковано механізми з найбільшою кількістю запчастин. Для реалізації використано `JOIN`, оператор `ANY` для роботи з масивами, агрегатну функцію `COUNT`, віконну функцію `RANK` та CTE. Усі запити виконано успішно, результати підтвердили коректність зв’язків між даними. Аналіз продуктивності за допомогою `EXPLAIN ANALYZE` показав ефективність запитів для невеликого обсягу даних, але для масштабування рекомендовано додати індекси на `Mechanism_IDs`. Результати та знімки екрану додано до звіту.

Відповіді на контрольні роботи

1. SQL розшифровується як Structured Query Language (структурована мова запитів).
2. Основні підмови SQL:
   - DDL (Data Definition Language) – мова визначення даних (CREATE, ALTER, DROP).
   - DML (Data Manipulation Language) – мова маніпулювання даними (SELECT, INSERT, UPDATE, DELETE).
   - DCL (Data Control Language) – мова контролю доступу (GRANT, REVOKE).
   - TCL (Transaction Control Language) – мова керування транзакціями (COMMIT, ROLLBACK, SAVEPOINT).
3. SQL традиційно використовується для роботи зі структурованими даними, але сучасні СУБД (наприклад, PostgreSQL, MySQL, SQL Server) підтримують JSON, XML, масиви та інші типи напівструктурованих і об'єктно-орієнтованих даних.
4. Для вибору певних рядків таблиці використовують оператор SELECT разом із WHERE, наприклад:

   ```
   SELECT * FROM users WHERE age > 18;
   ```
5. Для об'єднання таблиць використовують команду JOIN, наприклад:

   ```
   SELECT orders.id, customers.name
   FROM orders
   JOIN customers ON orders.customer_id = customers.id;
   ```
6. Умови, зазначені в ON, виконуються перед фільтрацією в WHERE. Це означає, що:
   - ON визначає, як поєднуються записи таблиць.
   - WHERE відфільтровує вже об'єднані результати.

   Наприклад:

   ```
   SELECT * FROM orders
   LEFT JOIN customers ON orders.customer_id = customers.id
   WHERE customers.country = 'Ukraine';
   ```