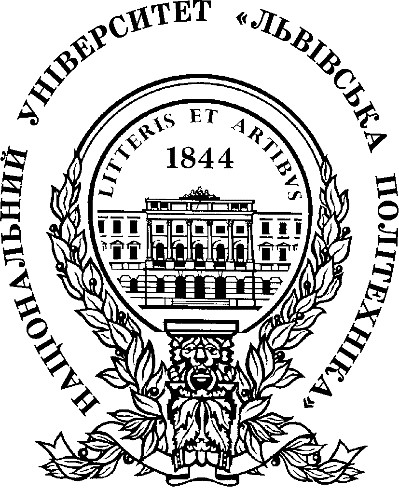
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра систем штучного інтелекту



ЗВІТ № 13 **з курсу “ОБДЗ”** на тему:

# «Аналіз та оптимізація запитів»

**Виконав:**

студент групи КН-208

Фіняк М.В.

**Викладач:**

Якимишин Х.М.

# Лабораторна робота № 13

**Мета роботи:** навчитися аналізувати роботу СУБД та оптимізовувати виконання складних запитів на вибірку даних. Виконати аналіз складних запитів за допомогою директиви EXPLAIN, модифікувати найповільніші запити з метою їх пришвидчення.

# Короткі теоретичні відомості

Для аналізу виконання запитів в MySQL існує декілька спеціальних директив. Основна з них – EXPLAIN.

Директива EXPLAIN дозволяє визначити поля таблиці, для яких варто створити додаткові індекси, щоб пришвидшити вибірку даних. Індекс – це механізм, який підвищує швидкість пошуку та доступу до записів за індексованими полями. Загалом, варто створювати індекси для тих полів, за якими відбувається з’єднання таблиць, перевірка умови чи пошук.

За допомогою директиви EXPLAIN також можна визначити послідовність, в якій відбувається з’єднання таблиць при вибірці даних. Якщо оптимізатор вибирає не найкращу послідовність з’єднання таблиць, потрібно використати опцію STRAIGHT\_JOIN директиви SELECT. Тоді з’єднання таблиць буде відбуватись в тому порядку, в якому перераховані таблиці у запиті. Також, за допомогою опцій FORCE INDEX, USE INDEX та IGNORE INDEX можна керувати використанням індексів у випадку їх неправильного вибору оптимізатором, тобто, якщо вони не підвищують ефективність вибірки рядків.

***Опис директив***

* SELECT BENCHMARK(кількість\_циклів, вираз)

Виконує вираз вказану кількість разів, і повертає загальний час виконання.

* EXPLAIN SELECT …

Використовується разом із запитом SELECT. Виводить інформацію про план обробки і виконання запиту, включно з інформацією про те, як і в якому порядку з’єднувались таблиці. EXPLAIN EXTENDED виводить розширену інформацію.

Результати директиви виводяться у вигляді рядків з такими полями:

* id – порядковий номер директиви SELECT у запиті;
* select\_type – тип вибірки (simple, primary, union, subquery, derived, uncachable subquery тощо);
* table – назва таблиці, для якої виводиться інформація;
* type – тип з’єднання (system, const, eq\_ref, ref, fulltext, range тощо);
* possible\_keys – індекси, які наявні у таблиці, і можуть бути використані;
* key – назва індексу, який було обрано для виконання запиту;
* key\_len – довжина індекса, який був використаний при виконанні запиту;
* ref – вказує, які рядки чи константи порівнюються зі значенням індекса при відборі;
* rows – (прогнозована) кількість рядків, потрібних для виконання запиту;
* Extra – додаткові дані про хід виконання запиту.

* ANALYZE TABLE

Оновлює статистичну інформацію про таблицю (наприклад, поточний розмір ключових полів). Ця інформація впливає на роботу оптимізатора запитів, і може вплинути на вибір індексів при виконанні запитів.

* SHOW INDEX FROM ім’я\_таблиці Виводить інформацію про індекси таблиці.
* CREATE [UNIQUE | FULLTEXT] INDEX назва

ON ім’я\_таблиці (перелік\_полів)

Створює індекс для одного або декількох полів таблиці. Одне поле може входити до кількох індексів. Якщо індекс оголошено як UNIQUE, то значення відповідних полів таблиці повинні бути унікальними. Таблиці MyISAM підтримують створення повнотекстових індексів (FULLTEXT) для полів типу TEXT, CHAR, VARCHAR.

# Хід роботи

1. За допомогою директиви SHOW INDEX визначимо наявні індекси для таблиці Dish.

**SHOW INDEX FROM dish;**

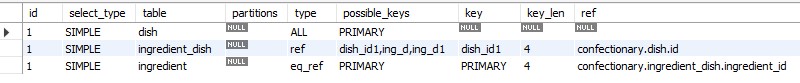


1. Виконаємо аналіз виконання складного запиту з однієї з попередніх робіт використовуючи EXPLAIN.

**EXPLAIN SELECT dish.name, ingredient.name from dish, ingredient, ingredient\_dish**

**WHERE dish.kitchen\_name='French' AND dish.id = ingredient\_dish.dish\_id1**

**AND ingredient.id = ingredient\_dish.ingredient\_id;**



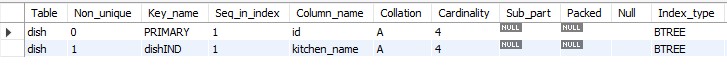
Для таблиці dish використовується тип з’єднання – ALL. Це найгірший тип,який свідчить про те, що буде відбуватись сканування усієї таблиці. Отже - потрібна оптимізація. Для цього створимо новий індекс, що пришвидшить пошук потрібної інформації.

Команда для створення індексу :

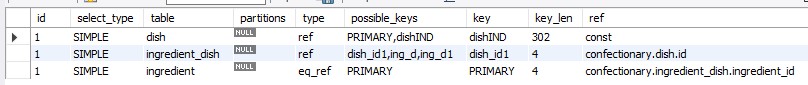
**CREATE INDEX dishIND ON dish(kitchen\_name);**

Перевіримо:

**SHOW INDEX FROM dish;**



Ще раз проведемо аналіз запиту:



Запит оптимізовано.

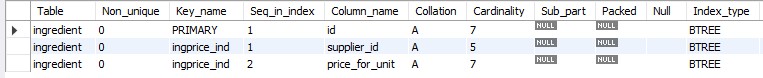
3. Проведемо аналіз виконання ще одного запиту з однієї з попередніх робіт використовуючи EXPLAIN :

Створимо унікальний індекс для таблиці ingredient:

**CREATE UNIQUE INDEX ingprice\_ind ON ingredient (supplier\_id, price\_for\_unit);**

Перевіримо:

**SHOW INDEX FROM ingredient;**



Проведемо аналіз запиту:

**EXPLAIN SELECT dish.id, dish.name AS dish\_name,**

**COUNT(ingredient.id) AS amount FROM (ingredient\_dish INNER JOIN**

**(supplier INNER JOIN ingredient)) INNER JOIN dish**

**ON supplier.id=ingredient.supplier\_id**

**AND ingredient.id=ingredient\_dish.ingredient\_id**

**AND ingredient\_dish.dish\_id1=dish.id**

**WHERE ingredient.price\_for\_unit BETWEEN 200 AND 400 GROUP BY dish\_name;**



Для пошуку даних використовується створений індекс і тип з’єднання ALL відсутній, отже - запит оптимізований.

4. Виконаємо аналіз виконання складного запиту з однієї з попередніх робіт використовуючи EXPLAIN та опцію STRAIGHT\_JOIN . **EXPLAIN SELECT dish.id, dish.name AS dish\_name,**

**COUNT(ingredient.id) AS amount**

**FROM dish INNER JOIN (ingredient\_dish INNER JOIN**

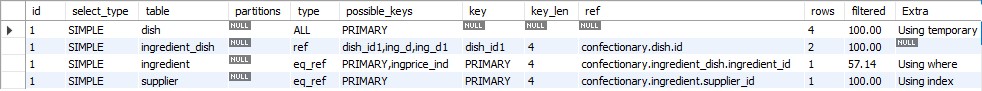
**(ingredient INNER JOIN supplier))**

**ON supplier.id=ingredient.supplier\_id**

**AND ingredient.id=ingredient\_dish.ingredient\_id**

**AND ingredient\_dish.dish\_id1=dish.id**

**WHERE ingredient.supplier\_id BETWEEN 1 AND 3 GROUP BY dish\_name;**



Запит не оптимізований, адже використовується тип з’єднання ALL та using temporary, що свідчить про створення тимчасової таблиці під час

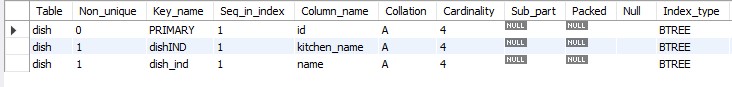
виконання запиту.

Для оптимізації створимо індекс для таблиці dish та використаємо опцію STRAIGHT\_JOIN:

CREATE INDEX dish\_ind ON dish(name);

Перевіримо:

**SHOW INDEX FROM dish;**



Ще раз проведемо аналіз запиту :

**EXPLAIN SELECT STRAIGHT\_JOIN dish.id, dish.name AS dish\_name, COUNT(ingredient.id) AS amount**

**FROM dish INNER JOIN (ingredient\_dish INNER JOIN**

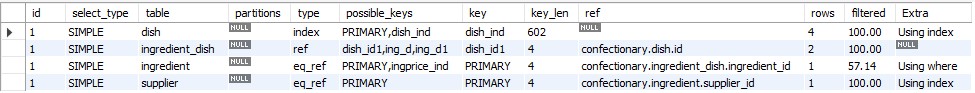
**(ingredient INNER JOIN supplier))**

**ON supplier.id=ingredient.supplier\_id**

**AND ingredient.id=ingredient\_dish.ingredient\_id**

**AND ingredient\_dish.dish\_id1=dish.id**

**WHERE ingredient.supplier\_id BETWEEN 1 AND 3 GROUP BY dish\_name;**



Тип з’єднання ALL – відсутній, using temporary теж. Запит оптимізовано.

**Висновок :** під час виконання даної лабораторної роботи я навчився аналізувати роботу СУБД та оптимізовувати виконання складних запитів на вибірку даних. Для аналізу запитів використовував директиву EXPLAIN та опцію STRAIGHT\_JOIN, а для їх оптимізації – створювала додаткові індекси.