# МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



## Лабораторна робота №13

## З дисципліни «Організація баз даних та знань»

***Виконав:***

*студент групи КН-210*

*Кухар Ярослав*

***Перевірив:***

*Кандидат тех. наук, ст. викладач Мельникова Н. І.*

Львів – 2020

**Мета:** 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 пришвидшення.

# **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*)

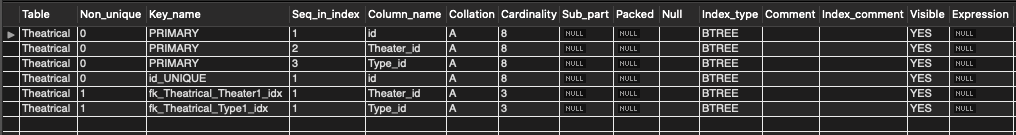
Створює індекс для одного або декількох полів таблиці. Одне поле може входити до кількох індексів. Якщо індекс оголошено як UNIQUE, то значення відповідних полів таблиці повинні бути унікальними. Таблиці MyISAM підтримують створення повнотекстових індексів (FULLTEXT) для полів типу TEXT, CHAR, VARCHAR.

# **Хід роботи**

1. Визначити індекси таблиці.
2. творити додаткові індекси для таблиці.
3. Дослідити процес виконання запитів за допомогою EXPLAIN.

1. За допомогою директиви SHOW INDEX визначимо наявні індекси для таблиць Theatrical i Genre.

show index from Theatrical;



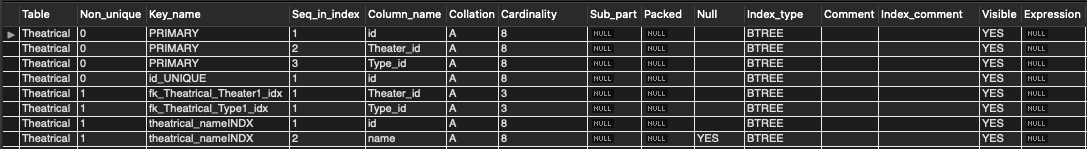
show index from Genre;



2. Створимо новий індекс для таблиці Theatrical i Genre. У БД є декілька запитів, які здійснюють вибірку даних за іменем (поле name), за датою (поле datetime) тощо. Cтворення індексів для цих полів повинно оптимізувати виконання запитів.

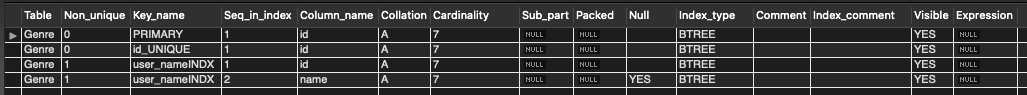
CREATE INDEX user\_nameINDX ON Theatrical (id, name);

SHOW INDEX FROM Theatrical;



CREATE INDEX user\_nameINDX ON Genre (id, name);

SHOW INDEX FROM Genre;



3. Виконаємо аналіз виконання складного запиту використовуючи EXPLAIN та опцію STRAIGHT\_JOIN.

explain select Theatrical.id as TheatricalId,

Theatrical.name as Name,

Type.name as Type,

Theater.name as Theater

from Theatrical

inner join Theater on Theater.id = Theatrical.Theater\_id

inner join Type on Type.id = Theatrical.Type\_id

where Theatrical.time > '2020-05-09 10:10:10';



Тепер застосуємо команду STRAIGHT\_JOIN:

explain select STRAIGHT\_JOIN Theatrical.id as TheatricalId,

Theatrical.name as Name,

Type.name as Type,

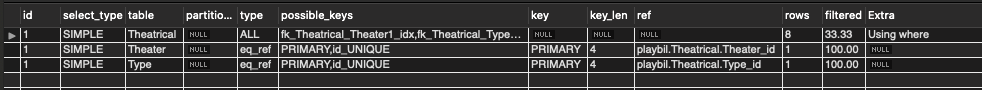
Theater.name as Theater

from Theatrical

inner join Theater on Theater.id = Theatrical.Theater\_id

inner join Type on Type.id = Theatrical.Type\_id

where Theatrical.time > '2020-05-09 10:10:10';



**Висновок**: На даній лабораторній роботі я навчився аналізувати і оптимізувати виконання запитів. Для аналізу запитів було використано директиву EXPLAIN, а для оптимізації – модифікація порядку з’єднання таблиць і створення додаткових індексів.