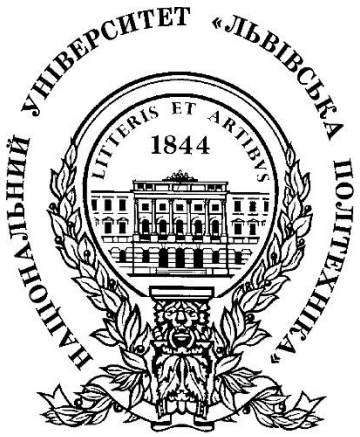
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
dish	0	PRIMARY	1	id	A	4	NULL	NULL		BTREE

2. 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;

id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	SIMPLE	dish	NULL	ALL	PRIMARY	NULL	NULL	NULL
1	SIMPLE	ingredient_dish	NULL	ref	dish_id1,ing_d,ing_d1	dish_id1	4	confectionary.dish.id
1	SIMPLE	ingredient	NULL	eq_ref	PRIMARY	PRIMARY	4	confectionary.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;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
dish	0	PRIMARY	1	id	A	4	NULL	NULL		BTREE
dish	1	dishIND	1	kitchen_name	A	4	NULL	NULL		BTREE

Ще раз проведемо аналіз запиту:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	SIMPLE	dish	NULL	ref	PRIMARY,dishIND	dishIND	302	const
1	SIMPLE	ingredient_dish	NULL	ref	dish_id1,ing_d,ing_d1	dish_id1	4	confectionary.dish.id
1	SIMPLE	ingredient	NULL	eq_ref	PRIMARY	PRIMARY	4	confectionary.ingredient_dish.ingredient_id

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;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
ingredient	0	PRIMARY	1	id	A	7	NULL	NULL		BTREE
ingredient	0	ingprice_ind	1	supplier_id	A	5	NULL	NULL		BTREE
ingredient	0	ingprice_ind	2	price_for_unit	A	7	NULL	NULL		BTREE

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;

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	SIMPLE	ingredient	NULL	index	PRIMARY,ingprice_ind	ingprice_ind	9	NULL	7
1	SIMPLE	supplier	NULL	eq_ref	PRIMARY	PRIMARY	4	confectionary.ingredient.supplier_id	1
1	SIMPLE	ingredient_dish	NULL	ref	dish_id1,ing_d,ing_d1	ing_d1	4	confectionary.ingredient.id	1
1	SIMPLE	dish	NULL	eq_ref	PRIMARY	PRIMARY	4	confectionary.ingredient_dish.dish_id1	1

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;

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	dish	NULL	ALL	PRIMARY	NULL	NULL	NULL	4	100.00	Using temporary
1	SIMPLE	ingredient_dish	NULL	ref	dish_id1,ing_d,ing_d1	dish_id1	4	confectionary.dish.id	2	100.00	NULL
1	SIMPLE	ingredient	NULL	eq_ref	PRIMARY,ingprice_ind	PRIMARY	4	confectionary.ingredient_dish.ingredient_id	1	57.14	Using where
1	SIMPLE	supplier	NULL	eq_ref	PRIMARY	PRIMARY	4	confectionary.ingredient.supplier_id	1	100.00	Using index

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;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
dish	0	PRIMARY	1	id	A	4	NULL	NULL		BTREE
dish	1	dishIND	1	kitchen_name	A	4	NULL	NULL		BTREE
dish	1	dish_ind	1	name	A	4	NULL	NULL		BTREE

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;

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	dish	NULL	index	PRIMARY,dish_ind	dish_ind	602	NULL	4	100.00	Using index
1	SIMPLE	ingredient_dish	NULL	ref	dish_id1,ing_d,ing_d1	dish_id1	4	confectionary.dish.id	2	100.00	NULL
1	SIMPLE	ingredient	NULL	eq_ref	PRIMARY,ingprice_ind	PRIMARY	4	confectionary.ingredient_dish.ingredient_id	1	57.14	Using where
1	SIMPLE	supplier	NULL	eq_ref	PRIMARY	PRIMARY	4	confectionary.ingredient.supplier_id	1	100.00	Using index

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.