МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
Кафедра систем штучного інтелекту

ЗВІТ 13

з курсу “ОБДЗ”

на тему:

“Аналіз та оптимізація запитів ”

Виконала:
студентка групи КН-211
Досяк Ірина
Викладач:
Якимишин Х.М.

Лабораторна №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 пришвидшення.

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.

SELECT BENCHMARK - 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.

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 - виводить інформацію про індекси таблиці.

CREATE [UNIQUE | FULLTEXT] INDEX - 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.

Завдання

Провести аналіз виконання запитів з попередніх робіт. Модифікувати найповільніші запити з метою їх пришвидшення.

Виконання

1. Проведемо аналіз виконання запиту з однієї з попередніх робіт використовуючи EXPLAIN :

EXPLAIN SELECT reader.name from reader, reader_book
WHERE reader_book.status='read' AND reader.id_reader=reader_book.id_reader

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	reader_book	NULL	ALL	id_reader_idx	NULL	NULL	NULL	6	25.00	Using where
	1	SIMPLE	reader	NULL	eq_ref	PRIMARY	PRIMARY	4	library.reader_book.id_reader	1	100.00	NULL

Як бачимо, reader_book не є оптимізованим , оскільки використовується найгірший тип з’єднання - ALL, що свідчить про те, що буде проходити сканування усієї таблиці. Щоб вирішити проблему, потрібно використати індекс. Індекси використовуються у MySQL для пошуку рядків з вказаними значеннями колонок, наприклад, з командою WHERE. Без індексів, MySQL має, починаючи з першого рядка, прочитати всю таблицю у пошуках потрібних значень.

Переглянемо наявні індекси в reader_book :

SHOW INDEX FROM reader_book

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
reader_book	0	PRIMARY	1	id	A	1	NULL	NULL		BTREE			YES	NULL
reader_book	1	id_reader_book_idx	1	id_reader_book	A	1	NULL	NULL		BTREE			YES	NULL
reader_book	1	id_reader_idx	1	id_reader	A	1	NULL	NULL		BTREE			YES	NULL

Серед наявних індексів немає такого, який допоміг би оптимізувати reader_book, а саме reader_book.status.

Створимо потрібний індекс:

CREATE INDEX reader_book_indx ON reader_book(status);

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	reader_book	NULL	ref	id_reader_idx,reader_book_indx	reader_book_indx	2	const	3	100.00	NULL
1	SIMPLE	reader	NULL	eq_ref	PRIMARY	PRIMARY	4	library.reader_book.id_reader	1	100.00	NULL

2. Проведемо аналіз виконання запиту з однієї з попередніх робіт використовуючи EXPLAIN :

EXPlAIN SELECT book.name as book, author.firstname as author FROM (book
INNER JOIN author)
INNER JOIN book_author ON book.id_book=book_author.id_book_author
AND book_author.id_author=author.id_author
ORDER BY book.name;

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	book	NULL	ALL	PRIMARY	NULL	NULL	NULL	5	100.00	Using filesort
1	SIMPLE	book_author	NULL	ref	id_author_idx,id_book_author_idx	id_book_author_idx	4	library.book.id_book	1	100.00	NULL
1	SIMPLE	author	NULL	eq_ref	PRIMARY	PRIMARY	4	library.book_author.id_author	1	100.00	NULL

book не є оптимізованим, створимо новий індекс:

CREATE INDEX book ON book(name);

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	book	NULL	index	NULL	book	135	NULL	5	100.00	Using index
1	SIMPLE	book_author	NULL	ref	id_author_idx,id_book_author_idx	id_book_author_idx	4	library.book.id_book	1	100.00	NULL
1	SIMPLE	author	NULL	eq_ref	PRIMARY	PRIMARY	4	library.book_author.id_author	1	100.00	NULL

Висновок: на лабораторній роботі я навчилась аналізувати і оптимізовувати виконання запитів. Для аналізу запитів було використано директиву EXPLAIN, а для оптимізації – створення додаткових індексів.