МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

КАФЕДРА СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

Звіт до лабораторної роботи №13
з дисципліни:
“ОБДЗ”
на тему:
“ **Аналіз та оптимізація запитів** ”

Підготувала:
студентка групи КН-209
Дипко Олександра
Викладач:
Мельникова Н.І.

Львів 2020

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. 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. Створити додаткові індекси для таблиці.
3. Дослідити процес виконання запитів за допомогою EXPLAIN.

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	17	25.00	Using where
1	SIMPLE	reader	NULL	eq_ref	PRIMARY	PRIMARY	4	lib.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	17	NULL	NULL		BTREE			YES	NULL
reader_book	1	id_reader_book_idx	1	id_reader_book	A	8	NULL	NULL		BTREE			YES	NULL
reader_book	1	id_reader_idx	1	id_reader	A	8	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);
```

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	17	NULL	NULL		BTREE			YES	NULL
reader_book	1	id_reader_book_idx	1	id_reader_book	A	8	NULL	NULL		BTREE			YES	NULL
reader_book	1	id_reader_idx	1	id_reader	A	8	NULL	NULL		BTREE			YES	NULL
reader_book	1	reader_book_indx	1	status	A	4	NULL	NULL	YES	BTREE			YES	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	author	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using temporary; Using filesort
1	SIMPLE	book_author	NULL	ALL	id_author_idx,id_book_author_idx	NULL	NULL	NULL	7	16.67	Using where; Using join buffer (Block
1	SIMPLE	book	NULL	eq_ref	PRIMARY	PRIMARY	4	lib.book_author.id_book_author	1	100.00	NULL

Таблиця **book** не є оптимізованою, створимо новий індекс:

```
CREATE INDEX book ON book(name);
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
book	0	PRIMARY	1	id_book	A	12	NULL	NULL		BTREE			YES	NULL
book	1	id_publish_house_idx	1	id_publish_house	A	4	NULL	NULL	YES	BTREE			YES	NULL
book	1	book	1	name	A	12	NULL	NULL		BTREE			YES	NULL

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

```
EXPLAIN SELECT DISTINCT count(DISTINCT genre_book.id_book) as book,
genre.name as genre, count(DISTINCT reader_book.id_reader) as readers
FROM (genre JOIN book)
JOIN genre_book ON book.id_book = genre_book.id_genre
AND genre_book.id_genre = genre.id_genre
JOIN reader_book ON genre_book.id_book = reader_book.id_reader_book
GROUP BY genre ORDER BY genre.name;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	genre_book	NULL	ALL	id_book_idx,id_genre_idx	NULL	NULL	NULL	21	100.00	Using temporary; Using filesort
1	SIMPLE	book	NULL	eq_ref	PRIMARY	PRIMARY	4	lib.genre_book.id_genre	1	100.00	Using index
1	SIMPLE	genre	NULL	eq_ref	PRIMARY	PRIMARY	4	lib.genre_book.id_genre	1	100.00	NULL
1	SIMPLE	reader_book	NULL	ref	id_reader_book_idx	id_reader_book_idx	4	lib.genre_book.id_book	2	100.00	NULL

Як бачимо, запит не є оптимізованим, оскільки використовується using filesort, using temporary. Використаємо STRAIGHT_JOIN, який покращить послідовність з'єднання таблиць і оптимізує запит в цілому:

```
EXPLAIN SELECT STRAIGHT_JOIN DISTINCT count(DISTINCT
genre_book.id_book) as book, genre.name as genre, count(DISTINCT
reader_book.id_reader) as readers
FROM (genre JOIN book)
JOIN genre_book ON book.id_book = genre_book.id_genre
AND genre_book.id_genre = genre.id_genre
JOIN reader_book ON genre_book.id_book = reader_book.id_reader_book
GROUP BY genre ORDER BY genre.name;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	genre	NULL	ALL	PRIMARY	NULL	NULL	NULL	14	100.00	Using temporary; Using filesort
1	SIMPLE	book	NULL	eq_ref	PRIMARY	PRIMARY	4	lib.genre.id_genre	1	100.00	Using index
1	SIMPLE	genre_book	NULL	ALL	id_book_idx,id_genre_idx	NULL	NULL	NULL	21	20.00	Using where; Using join buffer (Block
1	SIMPLE	reader_book	NULL	ref	id_reader_book_idx	id_reader_book_idx	4	lib.genre_book.id_book	2	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.