Лабораторна робота №13
з курсу “ОБДЗ”
на тему:
“Аналіз та оптимізація запитів”

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`.

Завдання на лабораторну роботу.

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

Хід роботи.

1. За допомогою директиви `SHOW INDEX` визначимо наявні індекси для таблиць Author і Message.

```
SHOW INDEX FROM author;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
author	0	PRIMARY	1	authorID	A	2	*NULL*	*NULL*		BTREE
author	1	roleID	1	roleID	A	2	*NULL*	*NULL*	YES	BTREE

```
SHOW INDEX FROM message;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
message	0	PRIMARY	1	messageID	A	9	*NULL*	*NULL*		BTREE
message	1	message_author	1	authorID	A	9	*NULL*	*NULL*		BTREE

2. Створимо новий індекс для таблиці Author і Message. У БД є декілька запитів, які здійснюють вибірку даних за логіном автора (поле login), за датою написання повідомлення (поле posted) тощо. Створення індексів для цих полів повинно оптимізувати виконання запитів.

```
CREATE INDEX authorINDX3 ON author (authorID, login);
SHOW INDEX FROM author;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
author	0	PRIMARY	1	authorID	A	2	*NULL*	*NULL*		BTREE
author	1	roleID	1	roleID	A	2	*NULL*	*NULL*	YES	BTREE
author	1	authorINDX3	1	authorID	A	2	*NULL*	*NULL*		BTREE
author	1	authorINDX3	2	login	A	2	*NULL*	*NULL*		BTREE

```
CREATE UNIQUE INDEX mposted_indx ON message (authorID, posted);
SHOW INDEX FROM message;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
message	0	PRIMARY	1	messageID	A	2	*NULL*	*NULL*		BTREE
message	0	mposted_indx	1	authorID	A	2	*NULL*	*NULL*		BTREE
message	0	mposted_indx	2	posted	A	2	*NULL*	*NULL*		BTREE

3. Виконаємо аналіз виконання складного запиту з однієї з попередніх робіт використовуючи `EXPLAIN` та опцію `STRAIGHT_JOIN`.

```
EXPLAIN SELECT cname AS tag, COUNT(message.messageID) AS amount
 FROM ((author INNER JOIN message)
 INNER JOIN message_category) INNER JOIN category
 ON author.login=name
 AND author.authorID=message.authorID
 AND message.messageID=message_category.messageID
 AND message_category.categoryID=category.categoryID
 WHERE message.posted BETWEEN '2008-01-01' AND '2009-05-05'
 GROUP BY tag;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	category	ALL	PRIMARY	*NULL*	*NULL*	*NULL*	18	Using temporary; Using filesort
1	SIMPLE	message_category	ref	UQ_Category_Message,cat_category	cat_category	4	mycms.category.categoryID	1	Using index
1	SIMPLE	message	eq_ref	PRIMARY,mposted_indx	PRIMARY	4	mycms.message_category.messageID	1	Using where
1	SIMPLE	author	eq_ref	PRIMARY,authorINDX3	PRIMARY	4	mycms.message.authorID	1	Using where

```
EXPLAIN SELECT STRAIGHT_JOIN cname AS tag,
 COUNT(message.messageID) AS amount
 FROM ((author INNER JOIN message)
 INNER JOIN message_category) INNER JOIN category
 ON author.login=name AND author.authorID=message.authorID
 AND message.messageID=message_category.messageID
 AND message_category.categoryID=category.categoryID
 WHERE message.posted BETWEEN '2008-01-01' AND '2009-05-05'
 GROUP BY tag;
```

select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
SIMPLE	author	ALL	PRIMARY,authorINDX3	*NULL*	*NULL*	*NULL*	16	Using where; Using temporary; Using filesort
SIMPLE	message	ref	PRIMARY,mposted_indx	mposted_indx	4	mycms.author.authorID	4	Using where; Using index
SIMPLE	message_category	ref	UQ_Category_Message,cat_category	UQ_Category_Message	4	mycms.message.messageID	2	Using index
SIMPLE	category	eq_ref	PRIMARY	PRIMARY	4	mycms.message_category.categoryID	1	

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.