МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота №13
З дисципліни «Організація баз даних та знань»

Виконав:
студент групи КН-210
Черещук Любомир

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. Створити додаткові індекси для таблиці.

3.Дослідити процес виконання запитів за допомогою EXPLAIN.

1. За допомогою директиви SHOW INDEX визначимо наявні індекси для таблиць user і order.

SHOW INDEX FROM sportproducts.user;

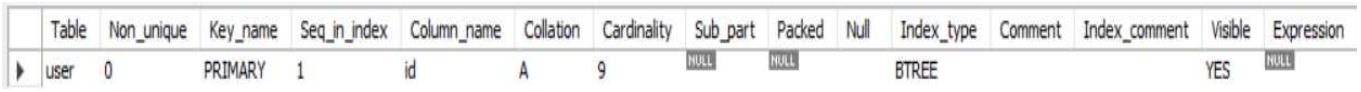


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
user	0	PRIMARY	1	id	A	9	NULL	NULL		BTREE			YES	NULL

SHOW INDEX FROM sportproducts.order;

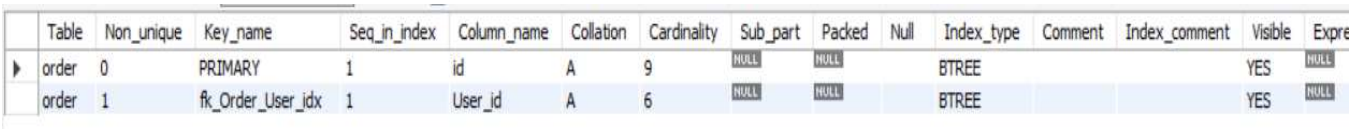


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expre
order	0	PRIMARY	1	id	A	9	NULL	NULL		BTREE			YES	NULL
order	1	fk_Order_User_idx	1	User_id	A	6	NULL	NULL		BTREE			YES	NULL

2. Створимо новий індекс для таблиці user і order. У БД є декілька запитів, які здійснюють вибірку даних за іменем користувача (поле name), за датою написання замовлення (поле datetime) тощо. Створення індексів для цих полів повинно оптимізувати виконання запитів.

CREATE INDEX user_nameINDX ON sportproducts.user (id, name);

SHOW INDEX FROM sportproducts.user;

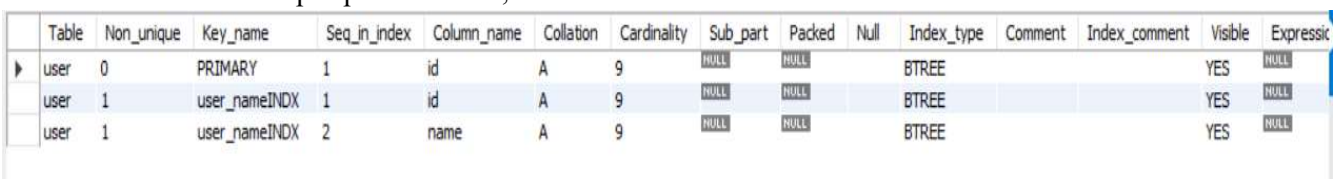


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expressic
user	0	PRIMARY	1	id	A	9	NULL	NULL		BTREE			YES	NULL
user	1	user_nameINDX	1	id	A	9	NULL	NULL		BTREE			YES	NULL
user	1	user_nameINDX	2	name	A	9	NULL	NULL		BTREE			YES	NULL

CREATE UNIQUE INDEX oorder_dateINDX ON sportproducts.order (User_id, datetime);

SHOW INDEX FROM sportproducts.order;

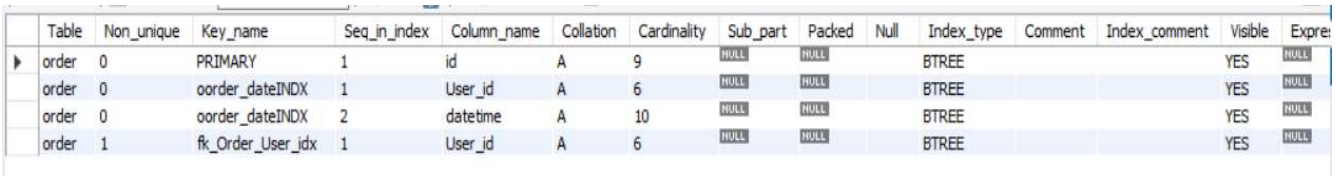
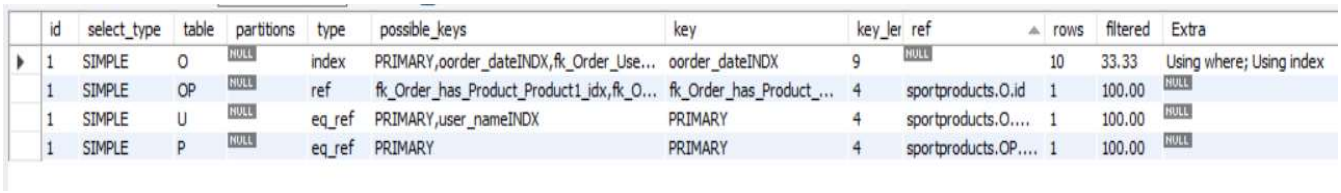


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expre
order	0	PRIMARY	1	id	A	9	NULL	NULL		BTREE			YES	NULL
order	0	oorder_dateINDX	1	User_id	A	6	NULL	NULL		BTREE			YES	NULL
order	0	oorder_dateINDX	2	datetime	A	10	NULL	NULL		BTREE			YES	NULL
order	1	fk_Order_User_idx	1	User_id	A	6	NULL	NULL		BTREE			YES	NULL

3. Виконаємо аналіз виконання складного запиту використовуючи EXPLAIN та опцію STRAIGHT_JOIN.

```
EXPLAIN SELECT U.id AS userID, U.name AS username,
                        O.datetime AS orderTime, P.price
FROM sportproducts.user U
INNER JOIN sportproducts.order O
        ON U.id = O.User_id
INNER JOIN sportproducts.order_product OP
        ON O.id = OP.Order_id
INNER JOIN sportproducts.product P
        ON OP.Product_id = P.id
```

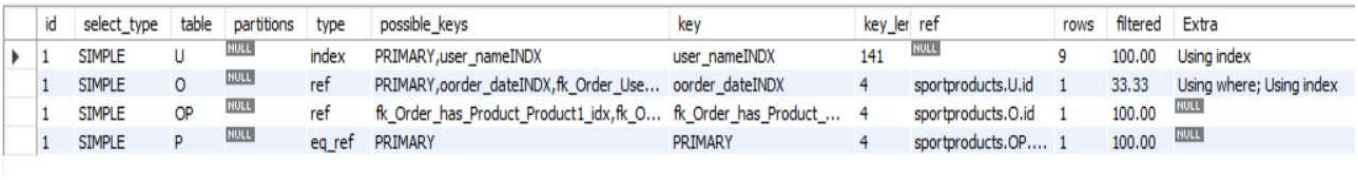
```
WHERE O.datetime > '2020-03-20 00:00:00';
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	O	NULL	index	PRIMARY,oorder_dateINDX,fk_Order_Use...	oorder_dateINDX	9	NULL	10	33.33	Using where; Using index
1	SIMPLE	OP	NULL	ref	fk_Order_has_Product_Product1_idx,fk_O...	fk_Order_has_Product_...	4	sportproducts.O.id	1	100.00	NULL
1	SIMPLE	U	NULL	eq_ref	PRIMARY,user_nameINDX	PRIMARY	4	sportproducts.O....	1	100.00	NULL
1	SIMPLE	P	NULL	eq_ref	PRIMARY	PRIMARY	4	sportproducts.OP....	1	100.00	NULL

Тепер застосуємо команду STRAIGHT_JOIN:

```
EXPLAIN SELECT STRAIGHT_JOIN U.id AS userID, U.name AS username,
                    O.datetime AS orderTime, P.price
FROM sportproducts.user U
INNER JOIN sportproducts.order O
ON U.id = O.User_id
INNER JOIN sportproducts.order_product OP
ON O.id = OP.Order_id
INNER JOIN sportproducts.product P
ON OP.Product_id = P.id
   WHERE O.datetime > '2020-03-20 00:00:00';
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	U	NULL	index	PRIMARY,user_nameINDX	user_nameINDX	141	NULL	9	100.00	Using index
1	SIMPLE	O	NULL	ref	PRIMARY,oorder_dateINDX,fk_Order_Use...	oorder_dateINDX	4	sportproducts.U.id	1	33.33	Using where; Using index
1	SIMPLE	OP	NULL	ref	fk_Order_has_Product_Product1_idx,fk_O...	fk_Order_has_Product_...	4	sportproducts.O.id	1	100.00	NULL
1	SIMPLE	P	NULL	eq_ref	PRIMARY	PRIMARY	4	sportproducts.OP....	1	100.00	NULL

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.