

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кафедра «Системи штучного інтелекту»



Звіт

до лабораторної роботи №13

З дисципліни «Організація баз даних та знань»

Виконав:

студент групи КН-211

Якимів Роман

Прийняла:

Якимишин Х. М.

Львів 2020

Лабораторна робота №13

“Аналіз та оптимізація запитів”

Мета роботи: Навчитися аналізувати роботу СУБД та оптимізувати 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.

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 consumer_order і café.

SHOW INDEX FROM consumer_order;

Result Grid											
		Filter Rows:			Export:			Wrap Cell Content:			
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
▶	consumer_order	0	PRIMARY	1	id	A	7	NULL	NULL		BTREE
	consumer_order	1	Order_fk0	1	consumer_id	A	2	NULL	NULL	YES	BTREE

SHOW INDEX FROM café;

Result Grid											
		Filter Rows:			Export:			Wrap Cell Content:			
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
▶	café	0	PRIMARY	1	id	A	3	NULL	NULL		BTREE

- Створимо новий індекс для таблиці consumer_order. У БД є декілька запитів, які здійснюють вибірку даних за номером, датою замовлення (поля id, date). Створення індексів для цих полів повинно оптимізувати виконання запитів.

```
CREATE INDEX orderINDX3 ON consumer_order (id, date);
SHOW INDEX FROM consumer_order;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	C
consumer_order	0	PRIMARY	1	id	A	7	NULL	NULL		BTREE	
consumer_order	1	Order_fk0	1	consumer_id	A	2	NULL	NULL	YES	BTREE	
consumer_order	1	orderINDX3	1	id	A	7	NULL	NULL		BTREE	
consumer_order	1	orderINDX3	2	date	A	7	NULL	NULL		BTREE	

- Виконаємо аналіз виконання складного запиту з однієї з попередніх робіт використовуючи EXPLAIN.

```
EXPLAIN SELECT
c.name as consumer,
count(o.id) as orders,
sum(o.price) as summary,
f.name AS cafe
FROM consumer c INNER JOIN consumer_order o on c.id = o.consumer_id
INNER JOIN order_cocktail b on o.id = b.order_id
INNER JOIN cocktail d on b.cocktail_id = d.id
INNER JOIN cafe_cocktail e on d.id = e.cocktail_id
INNER JOIN cafe f on f.id = e.cafe_id
WHERE month(o.date) = 4
GROUP BY 1
ORDER BY 3 desc;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	f	NULL	ALL	PRIMARY	NULL	NULL	NULL	3	100.00	Using temporary
1	SIMPLE	d	NULL	index	PRIMARY	PRIMARY	4	NULL	8	100.00	Using index
1	SIMPLE	e	NULL	ref	Cafe_Cocktail_fk0,Cafe_Cocktail_fk1	Cafe_Cocktail_fk0	5	drinmix.d.id	1	33.33	Using where
1	SIMPLE	b	NULL	ref	Order_Cocktail_fk0,Order_Cocktail_fk1	Order_Cocktail_fk1	5	drinmix.d.id	2	100.00	Using where
1	SIMPLE	o	NULL	eq_ref	PRIMARY,Order_fk0,orderINDX3	PRIMARY	4	drinmix.b.order_id	1	100.00	Using where
1	SIMPLE	c	NULL	eq_ref	PRIMARY	PRIMARY	4	drinmix.o.consumer_id	1	100.00	Using where

Як бачимо, щоб отримати ім'я кафе, нам потрібно сканувати всю таблицю кафе. Оскільки ім'я кафе нам досить часто буде потрібне, то створюємо індекс для поля ім'я кафе.

```
CREATE INDEX cafeINDX3 ON cafe (name);
SHOW INDEX FROM cafe;
```

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
►	cafe	0	PRIMARY	1	id	A	3	NULL	NULL		BTREE
	cafe	1	cafeINDX3	1	name	A	3	NULL	NULL		BTREE

Виконаємо селект знову:

Result Grid Filter Rows: Export: Wrap Cell Content: IA												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
►	1	SIMPLE	f	NULL	index	PRIMARY	cafeINDX3	1022	NULL	3	100.00	Using index
	1	SIMPLE	d	NULL	index	PRIMARY	PRIMARY	4	NULL	8	100.00	Using index
	1	SIMPLE	e	NULL	ref	Cafe_Cocktail_fk0,Cafe_Cocktail_fk1	Cafe_Cocktail_fk0	5	drinmix.d.id	1	33.33	Using where
	1	SIMPLE	b	NULL	ref	Order_Cocktail_fk0,Order_Cocktail_fk1	Order_Cocktail_fk1	5	drinmix.d.id	2	100.00	Using where
	1	SIMPLE	o	NULL	eq_ref	PRIMARY,Order_fk0,orderINDX3	PRIMARY	4	drinmix.b.order_id	1	100.00	Using where
	1	SIMPLE	c	NULL	eq_ref	PRIMARY	PRIMARY	4	drinmix.o.consumer_id	1	100.00	Using where

Тепер запит буде виконуватись швидше.

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.