МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра систем штучного інтелекту

Лабораторна робота №13

з дисципліни “ ОБДЗ ”

На тему: «Аналіз та оптимізація запитів»

Виконав:
ст. гр. КН-211
Ільків Андрій
Викладач:
Якимишин Х. М.

Львів – 2020

Мета роботи: Навчитися використовувати механізм транзакцій у СУБД MySQL. Розробити SQL запити, які виконуються як єдине ціле в рамках однієї транзакції.

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:

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 order, customer та order_item.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Student	0	PRIMARY	1	student_id	A	7	NULL	NULL		BTREE			YES	NULL
Student	1	Student_group	1	group_id	A	4	NULL	NULL		BTREE			YES	NULL

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Teacher	0	PRIMARY	1	teacher_id	A	4	NULL	NULL		BTREE			YES	NULL
Teacher	1	Teacher_car	1	car_id	A	4	NULL	NULL	YES	BTREE			YES	NULL

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Category	0	PRIMARY	1	category_id	A	3	NULL	NULL		BTREE			YES	NULL

```
show index from student;

show index from teacher;

show index from category;
```

2. Створимо новий індекс для таблиці teacher, student та category. У БД є декілька запитів, які здійснюють вибірку даних за логіном студента (поле login), за id категорії(category_id). Створення індексів для цих полів повинно оптимізувати виконання запитів.

```
create index studentindx on student (group_id, first_name, last_name);

create index groupidx on studentsgroup(group_id, category_id, lectionId);

create index categoryidx on category(category_id, category_description);
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
student	0	PRIMARY	1	student_id	A	6	NULL	NULL		BTREE			YES	NULL
student	1	studentindx	1	group_id	A	4	NULL	NULL		BTREE			YES	NULL
student	1	studentindx	2	first_name	A	6	NULL	NULL		BTREE			YES	NULL
student	1	studentindx	3	last_name	A	6	NULL	NULL		BTREE			YES	NULL

studentsgroup	0	PRIMARY	1	group_id	A	4	NULL	NULL		BTREE			YES	NULL
studentsgroup	1	Group_teacher	1	teacher_id	A	3	NULL	NULL		BTREE			YES	NULL
studentsgroup	1	Group_category	1	category_id	A	3	NULL	NULL		BTREE			YES	NULL
studentsgroup	1	groupidx	1	group_id	A	4	NULL	NULL		BTREE			YES	NULL
studentsgroup	1	groupidx	2	category_id	A	4	NULL	NULL		BTREE			YES	NULL
studentsgroup	1	groupidx	3	lectionId	A	4	NULL	NULL		BTREE			YES	NULL

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

Перший запит було реалізовано з навмисно неправильним порядком з'єднання таблиць, і результат показує, що таке з'єднання не використовує створені індекси і є досить повільним.

```
explain SELECT
    student.first_name,
    student.last_name,
    studentsgroup.lectionId,
    category.category_description
    as statk
FROM (student INNER JOIN category) INNER JOIN studentsgroup
    ON student.group_id = studentsgroup.group_id
    AND studentsgroup.category_id = category.category_id
    group by category_description;
```

id	select_type	table	partitio...	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	category	NULL	index	PRIMARY,categoryidx	categoryidx	1027	NULL	3	100.00	Using index; Using temporary
1	SIMPLE	studentsgroup	NULL	ref	PRIMARY,Group_category,groupidx	Group_category	4	bass.category.category_id	1	100.00	NULL
1	SIMPLE	student	NULL	ref	studentindx	studentindx	4	bass.studentsgroup.group_id	1	100.00	Using index

Використовуючи директиву straight_join, порядок з'єднань таблиць став оптимальним, і створені індекси майже втричі покращили швидкість запиту.

```
explain SELECT straight_join
    student.first_name,
    student.last_name,
    studentsgroup.lectionId,
    category.category_description
    as statk
FROM (student INNER JOIN category) INNER JOIN studentsgroup
    ON student.group_id = studentsgroup.group_id
    AND studentsgroup.category_id = category.category_id
    group by category_description;
```

id	select_type	table	partitio...	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	student	NULL	index	studentindx	studentindx	2048	NULL	6	100.00	Using index; Using temporary
1	SIMPLE	category	NULL	index	PRIMARY,categoryidx	categoryidx	1027	NULL	3	100.00	Using index; Using join buffer (Block Nested Loop)
1	SIMPLE	studentsgroup	NULL	eq_ref	PRIMARY,Group_category,groupidx	PRIMARY	4	bass.student.group_id	1	33.33	Using where

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 - straight_join.