МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

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

Виконав:
студент групи КН-210
Дойков Вадим

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.

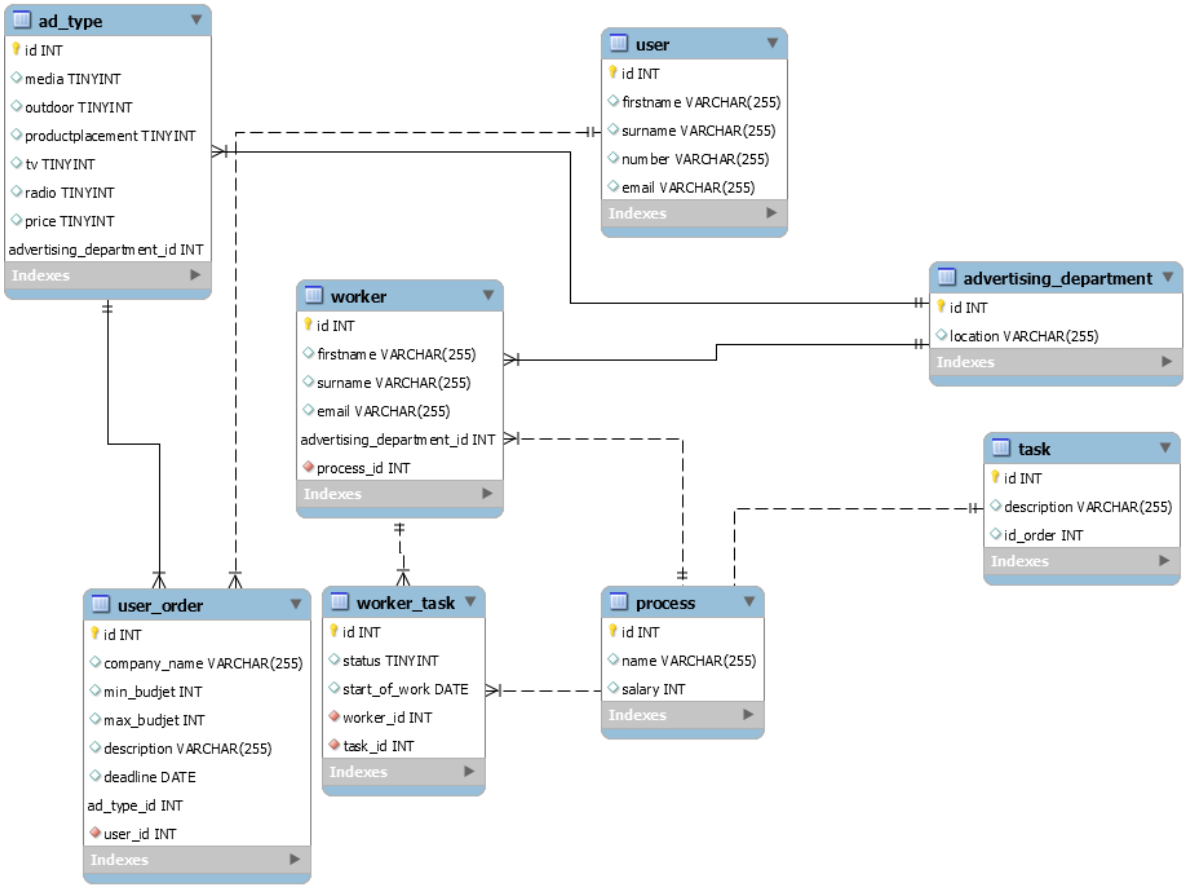

Рис 1. ER-діаграма

1.За допомогою директиви SHOW INDEX визначимо наявні індекси для таблиць user_order і ad_type.

```
3 •  show index from user_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	Comment	Index_comment	Visible	Expression
user_order	0	PRIMARY	1	id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	0	PRIMARY	2	ad_type_id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	1	fk_order_ad_type1_idx	1	ad_type_id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	1	fk_order_user1_idx	1	user_id	A	7	NULL	NULL		BTREE			YES	NULL

```
3 •  show index from ad_type;
```

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	Comment	Index_comment	Visible	Expression
ad_type	0	PRIMARY	1	id	A	9	NULL	NULL		BTREE			YES	NULL
ad_type	0	PRIMARY	2	advertising_department_id	A	9	NULL	NULL		BTREE			YES	NULL
ad_type	1	fk_ad_type_advertising_department_idx	1	advertising_department_id	A	4	NULL	NULL		BTREE			YES	NULL

2. Створимо новий індекс для таблиці user_order і ad_type. Індексів для цих таблиць повинні оптимізувати виконання запитів.

```
3 •  CREATE INDEX user_orderINDX ON user_order (id, company_name);
4 •  SHOW INDEX FROM user_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	Comment	Index_comment	Visible	Expression
user_order	0	PRIMARY	1	id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	0	PRIMARY	2	ad_type_id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	1	fk_order_ad_type1_idx	1	ad_type_id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	1	fk_order_user1_idx	1	user_id	A	7	NULL	NULL		BTREE			YES	NULL
user_order	1	user_orderINDX	1	id	A	10	NULL	NULL		BTREE			YES	NULL
user_order	1	user_orderINDX	2	company_name	A	10	NULL	NULL	YES	BTREE			YES	NULL

```
3 •  CREATE INDEX ad_typeINDX ON ad_type (id, advertising_department_id);
4 •  SHOW INDEX FROM ad_type;
```

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	Comment	Index_comment	Visible	Expression
ad_type	0	PRIMARY	1	id	A	9	NULL	NULL		BTREE			YES	NULL
ad_type	0	PRIMARY	2	advertising_department_id	A	9	NULL	NULL		BTREE			YES	NULL
ad_type	1	fk_ad_type_advertising_department_idx	1	advertising_department_id	A	4	NULL	NULL		BTREE			YES	NULL
ad_type	1	ad_typeINDX	1	id	A	9	NULL	NULL		BTREE			YES	NULL
ad_type	1	ad_typeINDX	2	advertising_department_id	A	9	NULL	NULL		BTREE			YES	NULL

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

```
7 •  explain select STRAIGHT_JOIN  u.id, concat(u.firstname, ' ', u.surname) as fullname, uo.company_name
8    from user as u
9    inner join  user_order as uo on uo.user_id = u.id
10   inner join ad_type as ad on ad.id = uo.ad_type_id
11   inner join advertising_department as d on d.id = ad.advertising_department_id
12   order by u.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	u	NULL	index	PRIMARY	PRIMARY	4	NULL	25	100.00	NULL
1	SIMPLE	uo	NULL	ref	fk_order_ad_type1_idx,fk_order_user1_idx	fk_order_user1_idx	4	mydb.u.id	1	100.00	NULL
1	SIMPLE	ad	NULL	ref	PRIMARY,fk_ad_type_advertising_department...	PRIMARY	4	mydb.uo.ad_type_id	1	100.00	Using index
1	SIMPLE	d	NULL	eq_ref	PRIMARY	PRIMARY	4	mydb.ad.advertising_d...	1	100.00	Using index

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.