МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота 13

з організації баз даних та знать

Виконав:

Студент групи КН-208

Воробель Адріан

Викладач:

Якимишин Х. М.

Львів – 2020р.

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 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`.

Хід роботи.

1. Визначити індекси таблиці.

```
show indexes from car;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
car	0	PRIMARY	1	id	A	18	NULL	NULL		BTREE

```
show indexes from distributor;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
distributor	0	PRIMARY	1	id	A	4	NULL	NULL		BTREE

2. Створити додаткові індекси для таблиці.

В таблиці car є поля mark і model, до яких дуже часто звертаються, також в таблиці distributor є поле distrib_name до якого активно посилають запити, створимо для цих полів додаткові індекси щоб оптимізувати вибірку.

```
create index carINDX2 on car (id, mark, model);
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
car	0	PRIMARY	1	id	A	18	NULL	NULL		BTREE
car	1	carINDX2	1	id	A	18	NULL	NULL		BTREE
car	1	carINDX2	2	mark	A	11	NULL	NULL		BTREE
car	1	carINDX2	3	model	A	18	NULL	NULL		BTREE

```
create index distributorINDX on distributor (id,
distrib_name);
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
distributor	0	PRIMARY	1	id	A	4	NULL	NULL		BTREE
distributor	1	distributorINDX	1	id	A	4	NULL	NULL		BTREE
distributor	1	distributorINDX	2	distrib_name	A	4	NULL	NULL		BTREE

3. Дослідити процес виконання запитів за допомогою `EXPLAIN`

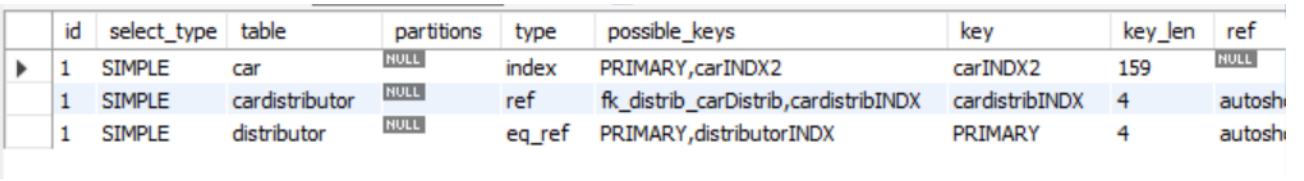
Виберемо найскладніший запит з попередніх лабораторних.

```
explain SELECT car.mark, car.model,
distributor.distrib_name as distributorID FROM car

INNER JOIN cardistributor ON car.id = cardistributor.car_id
```

```
INNER JOIN distributor ON cardistributor.distrib_id = distributor.id

WHERE car.car_type NOT IN ("Pickup","Crossover");
```



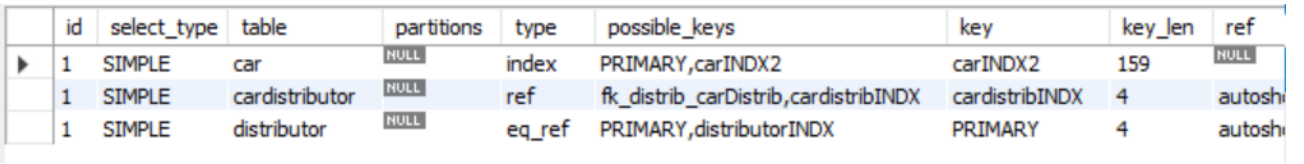
id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	SIMPLE	car	NULL	index	PRIMARY,carINDX2	carINDX2	159	NULL
1	SIMPLE	cardistributor	NULL	ref	fk_distrib_carDistrib,cardistribINDX	cardistribINDX	4	autosh
1	SIMPLE	distributor	NULL	eq_ref	PRIMARY,distributorINDX	PRIMARY	4	autosh

```
explain SELECT straight_join car.mark, car.model, distributor.distrib_name as distributorID FROM car

INNER JOIN cardistributor ON car.id = cardistributor.car_id

INNER JOIN distributor ON cardistributor.distrib_id = distributor.id

WHERE car.car_type NOT IN ("Pickup","Crossover");
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	SIMPLE	car	NULL	index	PRIMARY,carINDX2	carINDX2	159	NULL
1	SIMPLE	cardistributor	NULL	ref	fk_distrib_carDistrib,cardistribINDX	cardistribINDX	4	autosh
1	SIMPLE	distributor	NULL	eq_ref	PRIMARY,distributorINDX	PRIMARY	4	autosh

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.