Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій

Кафедра «Системи штучного інтелекту»

Лабораторна робота №13

З дисципліни:

«Організація баз даних та знань»

Виконав:
Студент групи КН-208
Лукаш О.В.
Викладач:
Мельникова Н.І.

Львів-2020

Тема: Аналіз та оптимізація запитів.

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 пришвидчення.

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.

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 *user* і *service_centrum*.
   SHOW INDEX FROM *car_service.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_user	A	3	NULL	NULL		BTREE			YES	NULL
user	0	email	1	email	A	3	NULL	NULL		BTREE			YES	NULL
user	0	phone	1	phone	A	3	NULL	NULL		BTREE			YES	NULL

   SHOW INDEX FROM *car_servise.service_centrum*;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
service_centrum	0	PRIMARY	1	id_centrum	A	4	NULL	NULL		BTREE			YES	NULL
service_centrum	0	phone	1	phone	A	4	NULL	NULL		BTREE			YES	NULL
service_centrum	0	email	1	email	A	4	NULL	NULL	YES	BTREE			YES	NULL

2. Створимо новий індекс для таблиці *user* і *service_centrum*. Це мало б оптимізувати виконання запитів.

   CREATE INDEX userINDX4 **ON** *car_servise.user* (id_user, name);

   SHOW INDEX FROM *car_servise.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_user	A	3	NULL	NULL		BTREE			YES	NULL
user	0	email	1	email	A	3	NULL	NULL		BTREE			YES	NULL
user	0	phone	1	phone	A	3	NULL	NULL		BTREE			YES	NULL
user	1	userINDX4	1	id_user	A	3	NULL	NULL		BTREE			YES	NULL
user	1	userINDX4	2	name	A	3	NULL	NULL		BTREE			YES	NULL

   CREATE INDEX servis_centrumINDX4 **ON** *car_servise.service_centrum* (id_centrum, name);

   SHOW INDEX FROM *car_servise.service_centrum*;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
service_centrum	0	PRIMARY	1	id_centrum	A	4	NULL	NULL		BTREE			YES	NULL
service_centrum	0	phone	1	phone	A	4	NULL	NULL		BTREE			YES	NULL
service_centrum	0	email	1	email	A	4	NULL	NULL	YES	BTREE			YES	NULL
service_centrum	1	servis_centrumINDX4	1	id_centrum	A	4	NULL	NULL		BTREE			YES	NULL
service_centrum	1	servis_centrumINDX4	2	name	A	4	NULL	NULL		BTREE			YES	NULL

3. Виконаємо аналіз виконання запиту використовуючи EXPLAIN.

```
EXPLAIN select  concat(user.name,"  ",user.surname) as name,
service_centrum.name as centrum,
sum(service.price) as price,
discount(sum(service.price)) as discount_price
FROM user inner join (
        car inner join (
                service_centrum inner join (
                        repair_work inner join (
                                check_list inner join service
                                on check_list.id_service = service.id_service)
                        on repair_work.id_work = check_list.id_repair)
        on service_centrum.id_centrum = repair_work.id_centrum)
         on car.id_car = repair_work.id_car)
on user.id_user = car.id_user
group by name, centrum
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	service	NULL	ALL	PRIMARY	NULL	NULL	NULL	1	100.00	Using temporary
1	SIMPLE	check_list	NULL	ref	repair_work_fk,service_fk	service_fk	4	car_servise.service.id_service	1	100.00	NULL
1	SIMPLE	service_centrum	NULL	index	PRIMARY,servis_centrumINDX4	servis_centrumINDX4	126	NULL	4	100.00	Using index; Using join buffer (Block Nested Loop)
1	SIMPLE	repair_work	NULL	eq_ref	PRIMARY,service_centrum_fk,car_fk	PRIMARY	4	car_servise.check_list.id_repair	1	33.33	Using where
1	SIMPLE	car	NULL	eq_ref	PRIMARY,user_fk	PRIMARY	4	car_servise.repair_work.id_car	1	100.00	NULL
1	SIMPLE	user	NULL	eq_ref	PRIMARY,userINDX4	PRIMARY	4	car_servise.car.id_user	1	100.00	NULL

Використовуючи **STRAIGHT_JOIN**:

```
EXPLAIN STRAIGHT_JOIN select  concat(user.name,"  ",user.surname) as name,
service_centrum.name as centrum,
sum(service.price) as price,
discount(sum(service.price)) as discount_price
FROM user inner join (
        car inner join (
                service_centrum inner join (
                        repair_work inner join (
                                check_list inner join service
                                on check_list.id_service = service.id_service)
                        on repair_work.id_work = check_list.id_repair)
        on service_centrum.id_centrum = repair_work.id_centrum)
         on car.id_car = repair_work.id_car)
on user.id_user = car.id_user
group by name, centrum
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	user	NULL	ALL	PRIMARY,userINDX4	NULL	NULL	NULL	3	100.00	Using temporary
1	SIMPLE	car	NULL	ref	PRIMARY,user_fk	user_fk	4	car_servise.user.id_user	2	100.00	Using index
1	SIMPLE	service_centrum	NULL	index	PRIMARY,servis_centrumINDX4	servis_centrumINDX4	126	NULL	4	100.00	Using index; Using join buffer (Block Nested Loop)
1	SIMPLE	repair_work	NULL	ALL	PRIMARY,service_centrum_fk,car_fk	NULL	NULL	NULL	6	16.67	Using where; Using join buffer (Block Nested Lo…
1	SIMPLE	check_list	NULL	ref	repair_work_fk,service_fk	repair_work_fk	4	car_servise.repair_work.id_work	3	100.00	NULL
1	SIMPLE	service	NULL	ALL	PRIMARY	NULL	NULL	NULL	1	100.00	Using where; Using join buffer (Block Nested Lo…

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.