Кафедра систем штучного інтелекту

Лабораторна робота №13

з дисципліни

«Організація баз даних та знань»

Виконав:

студент групи КН-208

Горностай Б.Я.

Викладач:

к.т.н.

Мельникова Н.І.

Львів – 2020 р.

Аналіз та оптимізація запитів

Мета роботи: Розробити SQL запити, які моделюють роботу тригерів: каскадне знищення, зміна та доповнення записів у зв’язаних таблицях.

Хід роботи

1. Визначити індекси таблиці.

Спочатку потрібно за допомогою дерективи **SHOW INDEX** подивитись для яких таблиць потрібно добавити додатковий індекс. Так я в таблиці **order_goods** не було конкретного індексу, то я обрав її

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
order_goods	0	PRIMARY	1	order_id	A	10	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_fk0	1	staff_id	A	8	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_fk1	1	customer_id	A	7	NULL	NULL		BTREE			YES	NULL

Далі добавляємо новий індекс, який буде виконувати швидші записи для індексованих полів **order_id, staff_id, customer_id, quantity_of_goods;**

2. Створити додаткові індекси для таблиці.

```
1 •   CREATE INDEX order_goods_index
2     ON order_goods(order_id, staff_id, customer_id, quantity_of_goods);
```

Переглянемо чи індекс добавився

```
1 •   show index from order_goods
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expressior
order_goods	0	PRIMARY	1	order_id	A	10	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_fk0	1	staff_id	A	8	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_fk1	1	customer_id	A	7	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_index	1	order_id	A	11	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_index	2	staff_id	A	11	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_index	3	customer_id	A	11	NULL	NULL		BTREE			YES	NULL
order_goods	1	order_goods_index	4	quantity_of_goods	A	11	NULL	NULL		BTREE			YES	NULL

3.Дослідити процес виконання запитів за допомогою EXPLAIN.

Виконуємо запит за допомогою директиви **EXPLAIN** та **STRAIGHT_JOIN** (якщо оптимізатор вибирає не найкращу послідовність з’єднання таблиць)

для таблиць **order_goods, goods, client,** який буде виводити ім’я, прізвище клієнта, ім’я, прізвище працівника номер замовлення, та кількість товарів в замовленні

Виконуємо спочатку запит без застосування додаткових індексів

```
EXPLAIN SELECT STRAIGHT_JOIN
customer.name, customer.surname, staff.name, staff.surname, order_goods.order_id, order_goods.quantity_of_goods
from
customer inner join order_goods
on customer.customer_id = order_goods.customer_id
inner join staff
on order_goods.staff_id = staff.staff_id;
```

Як бачимо воно перебирає всі доступні ключі

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	customer	NULL	ALL	PRIMARY	NULL	NULL	NULL	14	100.00	NULL
1	SIMPLE	order_goods	NULL	ref	order_goods_fk0,order_goods_fk1	order_goods_fk1	8	lab.customer.customer_id	1	100.00	NULL
1	SIMPLE	staff	NULL	eq_ref	PRIMARY	PRIMARY	8	lab.order_goods.staff_id	1	100.00	NULL

Тепер після створення індексу воно буде обирати тільки наш індекс

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	order_goods	NULL	index	order_goods_fk0,order_goods_fk1	order_goods_index	32	NULL	11	100.00	Using index
1	SIMPLE	staff	NULL	eq_ref	PRIMARY	PRIMARY	8	lab.order_goods.staff_id	1	100.00	NULL
1	SIMPLE	customer	NULL	eq_ref	PRIMARY	PRIMARY	8	lab.order_goods.customer_id	1	100.00	NULL

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.