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: «ОБДЗ»

Виконав студент
групи КН-208
Жеребецький Олег
Прийняла:
асистент
Якимишин Х.М.

Львів-2020

Хід роботи.

1. Розглянемо які індекси є у таблички user

```
show index from user;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
user	0	PRIMARY	1	user_id	A	7	NULL	NULL		BTREE	
user	0	user_id	1	user_id	A	7	NULL	NULL		BTREE	

Index_comment	Visible	Expression
	YES	NULL
	YES	NULL

```
#non_uniq 0 - унікальне
# key_name - назва ключа
# Seq_in_index - позиція колонки в індексі
# collaction сортування A-зростає D-протилежне Null -нічо
# cardinality - кількість унікальних значення
# sub_part -  префікс індексу якшо весь - null, число - деякі індексовані
# packed - пакується ключ , нал якшо ніяк
# Null - так якшо може бути нал
# index_tye - метод індекс який використовуєть ся
# comment disable -якщо вимкнений
# index-comment - комент при створенні
# visible -видимий оптимізатору
# expression -... функції кейпартс
```

2. Глянемо індекси які уже є в табличці active_goods

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
active_goods	0	PRIMARY	1	active_goods_id	A	6	NULL	NULL		BTREE			YES	NULL
active_goods	0	comment_id	1	comment_id	A	3	NULL	NULL	YES	BTREE			YES	NULL
active_goods	1	active_goods_fk1	1	delivery_service_id	A	3	NULL	NULL		BTREE			YES	NULL

3. Знайдемо замовлення за певними умовами використовуючи стандартні індекси. І подивимось наскільки **ефективно** вони використовуються

```
EXPLAIN SELECT product_price, product_name, storage_adress_city
FROM (active_goods INNER JOIN product)
INNER JOIN storage
ON product.product_price >10
AND product.product_id = active_goods.product_id
AND product.storage_id = storage.storage_id
WHERE active_goods.activate_time BETWEEN '2020-04-01' AND '2020-12-01'
GROUP BY product_price;
```

Отримаємо **індекси** які використовуються для пошуку і усілякі параметри.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	SIMPLE	active_goods	NULL	ALL	active_goods_indx	NULL	NULL	NULL
1	SIMPLE	product	NULL	eq_ref	PRIMARY,product_fk0	PRIMARY	4	internet_shop_1.active_goods.product_id
1	SIMPLE	storage	NULL	eq_ref	PRIMARY	PRIMARY	4	internet_shop_1.product.storage_id

rows	filtered	Extra
6	16.67	Using where; Using temporary
1	33.33	Using where
1	100.00	NULL

Бачимо що пошук в таблиці active_goods дуже **не ефективний** бо використовує усі 6 атрибутів таблиці для роботи. Отже потрібно його **оптимізувати**.

4. Потрібно вибирати такі атрибут, які є унікальні та по яким найчастіше проводиться відбір.

5. Створимо індекс який буде підв’язаний до унікального поля по якому ми найчастіше задаємо умови - active_goods.

```
create index active_goods_indx2 on active_goods (activate_time);
```

6. Перевіримо чи все успішно створилось

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
active_goods	0	PRIMARY	1	active_goods_id	A	6	NULL	NULL		BTREE
active_goods	0	comment_id	1	comment_id	A	3	NULL	NULL	YES	BTREE
active_goods	1	active_goods_fk1	1	delivery_service_id	A	3	NULL	NULL		BTREE
active_goods	1	active_goods_indx2	1	activate_time	A	6	NULL	NULL		BTREE

7. Ось тепер повторивши **дію** вище де ми шукали замовлення але уже використовуючи з’єднання таблиць **у порядку їх прописання в коді** (Straight_join) побачимо таку картину.

```
EXPLAIN SELECT STRAIGHT_JOIN product_price, product_name, storage_adress_city
FROM (active_goods INNER JOIN product)
INNER JOIN storage
ON product.product_price >10
AND product.product_id = active_goods.product_id
AND product.storage_id = storage.storage_id
WHERE active_goods.activate_time BETWEEN '2020-04-01' AND '2020-12-01'
GROUP BY product_price;
```

При дії використовувались обидві створені нами індекси і тепер для пошуку використовуються всього **два атрибути** цієї таблички

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	SIMPLE	active_goods	NULL	range	active_goods_indx2	active_goods_indx2	5	NULL	2
1	SIMPLE	product	NULL	eq_ref	PRIMARY,product_fk0,product_indx1	PRIMARY	4	internet_shop_1.active_goods.product_id	1
1	SIMPLE	storage	NULL	eq_ref	PRIMARY	PRIMARY	4	internet_shop_1.product.storage_id	1

filtered	Extra
100.00	Using index condition; Using where; Using temp…
40.00	Using where
100.00	NULL

8. Це досить **оптимізовано**)

9. Тепер повторимо усі ті ж дії для автоматичного вибору порядку join табличок.

10. Глянемо **індекси** які використовуються для пошуку і їх параметри.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	product	NULL	ALL	PRIMARY,product_fk0	NULL	NULL	NULL	5	33.33	Using where; Usi
1	SIMPLE	active_goods	NULL	ref	active_goods_indx,active_goods_indx2	active_goods_indx	5	internet_shop_1.product.product_id	1	33.33	Using where
1	SIMPLE	storage	NULL	eq_ref	PRIMARY	PRIMARY	4	internet_shop_1.product.storage_id	1	100.00	NULL

Як бачимо – пошук по табличці product дуже не ефективний а саме тип **ALL**

11. Добавимо індекс на поле **product_price** в табличці product, бо ми по ньому і шукаємо при виконанні цього селекту(найчастіше).

```
create index product_indx1 on product (product_price);
```

12. Виведемо індекси в табличці product.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
product	0	PRIMARY	1	product_id	A	5	NULL	NULL		BTREE			YES	NULL
product	1	product_fk0	1	storage_id	A	4	NULL	NULL		BTREE			YES	NULL
product	1	product_indx1	1	product_price	A	5	NULL	NULL		BTREE			YES	NULL

13. А тепер знову попробуємо виконати цей же селект в автоматичному виборі порядку join табличок.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	active_goods	NULL	range	active_goods_indx,active_goods_indx2	active_goods_indx2	5	NULL	2	100.00
1	SIMPLE	product	NULL	eq_ref	PRIMARY,product_fk0,product_indx1	PRIMARY	4	internet_shop_1.active_goods.product_id	1	40.00
1	SIMPLE	storage	NULL	eq_ref	PRIMARY	PRIMARY	4	internet_shop_1.product.storage_id	1	100.00

тепер це тип **eq_ref** – по одному полю для таблички product.

Запам’ятаємо, що велика кількість індексів значно збільшує час додавання видалення і оновлення, адже крім полів документа потрібно проводити роботу і з відповідними індексами

- **Тому не потрібно створювати зайві індекси**

14. Видалення індексів

```
drop index active_goods_indx on active_goods;
drop index active_goods_indx2 on active_goods;
drop index active_goods_fk1 on active_goods;
drop index product_indx1 on product;
```

+

Якщо ми захочемо видалити індекс який привязаний до **зовнішнього ключа** то потрібно буде спочаку видалити його зв’язок і аж тоді видаляти індекс і не забути вернути зв’язок!

```
alter table active_goods
    drop foreign key `active_goods_fk0`;
```

Висновок: на цій лабораторній роботі я навчився працювати з індексами. Та розібрався у значенні їх параметрів. Та як оптимізувати базу даних.