МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра систем штучного інтелекту

Лабораторна робота №13
із дисципліни
Бази даних

Виконав:
Ст. групи КН-207
Романко С.А.
Прийняв:
Мельникова Н.І.

Львів – 2018 р.

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

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.

Хід роботи.

1. Переглянемо наявні індекси в appointment та worker.

```
195 • show index from worker;
```

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	Express
worker	0	PRIMARY	1	idWorker	A	3	NULL	NULL		BTREE			YES	NULL
worker	0	PRIMARY	2	Appointment_idAppointment	A	3	NULL	NULL		BTREE			YES	NULL
worker	0	PRIMARY	3	office_idOffice	A	3	NULL	NULL		BTREE			YES	NULL
worker	0	idWorker_UNIQUE	1	idWorker	A	3	NULL	NULL		BTREE			YES	NULL
worker	1	fk_Worker_Appointment1_idx	1	Appointment_idAppointment	A	2	NULL	NULL		BTREE			YES	NULL
worker	1	fk_worker_office1_idx	1	office_idOffice	A	2	NULL	NULL		BTREE			YES	NULL

```
195 • show index from appointment;
```

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
appointment	0	PRIMARY	1	idAppointment	A	3	NULL	NULL		BTREE			YES	NULL
appointment	0	idAppointment_UNIQUE	1	idAppointment	A	3	NULL	NULL		BTREE			YES	NULL

2. Створимо індекс для Appointment.Appointment та Appointment.Salary

```
195 • create index appointmentINDX on appointment (appointment, salary);
196 • show index from appointment;
```

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
appointment	0	PRIMARY	1	idAppointment	A	3	NULL	NULL		BTREE			YES	NULL
appointment	0	idAppointment_UNIQUE	1	idAppointment	A	3	NULL	NULL		BTREE			YES	NULL
appointment	1	appointmentINDX	1	Appointment	A	3	NULL	NULL	YES	BTREE			YES	NULL
appointment	1	appointmentINDX	2	Salary	A	3	NULL	NULL	YES	BTREE			YES	NULL

3. Виконаємо аналіз з Explain та straight_join

```
195 • create index appointmentINDX on appointment (appointment, salary);
196
197 • explain select surname as Worker, count(appointment.idappointment)
198   from Worker inner join appointment
199   on worker.appointment_idAppointment = appointment.idAppointment
200   where appointment.salary > 100
201   group by Name;
```

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	appointment	NULL	index	PRIMARY,idAppointment_UNIQUE	appointmentINDX	188	NULL	3	33.33	Using where; Using index; Using temporary
1	SIMPLE	Worker	NULL	ALL	fk_Worker_Appointment1_idx	NULL	NULL	NULL	4	50.00	Using where; Using join buffer (Block Nested Lo...

```
197 • explain select straight_join surname as Worker, count(appointment.idappointment)
198   from Worker inner join appointment
199   on worker.appointment_idAppointment = appointment.idAppointment
200   where appointment.salary > 100
201   group by Name;
```

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	Worker	NULL	ALL	fk_Worker_Appointment1_idx	NULL	NULL	NULL	5	100.00	Using temporary
1	SIMPLE	appointment	NULL	eq_ref	PRIMARY,idAppointment_UNIQUE	PRIMARY	4	db1.Worker.Appointment_idAppointment	1	33.33	Using where

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.