PHẦN 2

I. Khắc phục sự cố về hiệu năng với EXPLAIN

1. Index tất cả các cột dùng trong lệnh 'where', 'order by' và 'group by'

Ngoài việc đảm bảo nhận dạng duy nhất các bản ghi, MySQL Indexes còn giúp máy chủ MySQL lấy bản ghi từ CSDL nhanh hơn. Index cũng rất có ích khi sắp xếp các bản ghi. MySQL Indexes có thể chiếm nhiều dung lượng hơn và giảm hiệu suất khi insert, delete và update. Tuy nhiên, nếu bảng của bạn có nhiều hơn 10 records, nó thể giảm đáng kể thời gian thực hiện truy vấn. Xét trường hợp chạy truy vấn SQL sau từ CSDL có 500 records không có Index:

```
mysql> select customer_id, customer_name from customers
where customer_id='160343';
```

Truy vấn trên sẽ buộc máy chủ MySQL tiến hành quét toàn bộ bảng để truy xuất bản ghi mà ta muốn tìm. MySQL có một câu lệnh EXPLAIN mà ta có thể sử dụng cùng với các lệnh SELECT, INSERT, UPDATE, DELETE để phân tích truy vấn của mình. Khi bạn đặt EXPLAIN vào trước câu lệnh SQL, MySQL sẽ hiển thị thông tin từ trình tối ưu hóa về kế hoạch thực hiện dự định:

```
mysql> explain select customer_id, customer_name from
customers where customer_id='160343';

+----+-------------+----------+------------+------+-------
--------+------+---------+------+------+----------+--------
-----+

| id | select_type | table     | partitions | type |
possible_keys | key  | key_len | ref  | rows | filtered |
Extra       |
```

```
+----+-------------+-----------+------------+------+---------------
--------+------+---------+------+------+----------+---------------
-----+

|  1 | SIMPLE      | customers | NULL       | ALL  | NULL
| NULL | NULL    | NULL |  500 |    10.00 | Using where |

+----+-------------+-----------+------------+------+---------------
--------+------+---------+------+------+----------+---------------
-----+
```

Như bạn thấy, trình tối ưu hóa đã hiển thị thông tin rất quan trọng có thể giúp ta tinh chỉnh lại bảng CSDL. Đầu tiên, rõ ràng rằng MySQL sẽ tiến hành quét toàn bộ bảng vì cột key là 'NULL'. Thứ hai, máy chủ MySQL đã chỉ rõ rằng nó sẽ tiến hành quét toàn bộ 500 hàng trong CSDL.

Để tối ưu hóa truy vấn trên, chúng ta chỉ cần thêm Index vào trường 'customer_id' bằng cú pháp dưới đây:

```
mysql> CREATE INDEX customer_id ON customers (customer_id);

Query OK, 0 rows affected (0.02 sec)

Records: 0  Duplicates: 0  Warnings: 0
```

Sau đó, ta chạy lại EXPLAIN và xem kết quả:

```
mysql> Explain select customer_id, customer_name from
customers where customer_id='160343';

+----+-------------+-----------+------------+------+---------------
--------+-------------+---------+-------+----------+-------+
-------+

| id | select_type | table     | partitions | type |
possible_keys | key         | key_len | ref   | rows |
filtered | Extra |
```

```
+----+-------------+----------+------------+-----+-------
--------+-------------+---------+-------+-----+----------+
-------+

|  1 | SIMPLE      | customers | NULL       | ref  |
customer_id   | customer_id | 13      | const |    1 |
100.00 | NULL  |

+----+-------------+----------+------------+-----+-------
--------+-------------+---------+-------+-----+----------+
-------+
```

Từ output trên, rõ ràng máy chủ MySQL sẽ sử dụng Index (customer_Id) để tìm kiếm. Ta có thể thấy rõ số lượng rows cần quét chỉ là 1, dù ta chạy truy vấn trên trong một bảng có 500 records. Indexes có thể rất có ích khi bạn truy vấn một tập dữ liệu lớn (ví dụ: một bảng có hàng triệu rows)

1. Tối ưu lệnh LIKE với UNION

Đôi khi, bạn có thể muốn chạy truy vấn bằng toán tử so sánh OR trên các cột khác nhau trong một bảng cụ thể. Khi từ khóa OR được sử dụng quá nhiều trong mệnh đề WHERE, nó có thể khiến trình tối ưu hóa MySQL chọn quét toàn bộ bảng để lấy bản ghi.

Mệnh đề UNION có thể làm cho truy vấn chạy nhanh hơn, đặc biệt nếu bạn có Index cho cột trong mệnh đề WHERE

Ví dụ, xét trường hợp bạn đang chạy truy vấn bên dưới với ‘first_name’ và ‘last_name’ được đánh index:

```
mysql> select * from students where first_name like 'Le%'
or last_name like 'Hau%' ;
```

Truy vấn trên có thể chạy chậm hơn nhiều so với truy vấn bên dưới sử dụng toán tử UNION kết hợp 2 kết quả của các truy vấn riêng biệt để tận dụng Index.

```
mysql> select  from students where first_name like  'Le%'
union all select from students where last_name like 'Hau%'
;
```

2. Tránh dùng LIKE Expressions với ký tự đại diện ở đầu

MySQL không thể sử dụng các Index khi có một ký tự đại diện ở đầu trong một truy vấn. Ta lấy ví dụ ở trên trên bảng students, tìm kiếm như thế này sẽ khiến MySQL thực hiện quét toàn bộ bảng ngay cả khi bạn đã đánh index trường 'first_name' trên bảng students.

```
mysql> select * from students where first_name like  '%nam'
;
```

Ta có thể thấy điều đó khi chạy lệnh EXPLAIN:

```
mysql> explain select * from students where first_name like
'%nam'  ;

+----+-------------+----------+------------+------+--------
-------+------+---------+------+------+----------+---------
----+

| id | select_type | table    | partitions | type |
possible_keys | key  | key_len | ref  | rows | filtered |
Extra       |

+----+-------------+----------+------------+------+--------
-------+------+---------+------+------+----------+---------
----+

|  1 | SIMPLE      | students | NULL       | ALL  | NULL
| NULL | NULL    | NULL |  500 |    11.11 | Using where |

+----+-------------+----------+------------+------+--------
-------+------+---------+------+------+----------+---------
----+
```

Như bạn thấy, MySQL phải duyệt qua 500 rows trong bảng students, điều đó làm chậm truy vấn của ta đi rất nhiều.

3. Sử dụng MySQL Full-Text Search

Nếu bạn gặp phải tình huống cần tìm kiếm dữ liệu bằng ký tự đại diện và bạn không muốn CSDL của mình chạy chậm, bạn nên cân nhắc sử dụng tìm kiếm toàn văn bản – MySQL Full-Text Search (FTS) vì nó nhanh hơn nhiều so với truy vấn sử dụng ký tự đại diện.

Hơn nữa, FTS cũng có thể mang lại kết quả tốt hơn và phù hợp hơn khi bạn tìm kiếm trong một CSDL lớn.

Để thêm Index tìm kiếm toàn văn vào bảng students, ta có thể sử dụng lệnh MySQL bên dưới:

```
mysql>ALTER TABLE students ADD FULLTEXT (first_name,
last_name);

mysql>SELECT * FROM students WHERE MATCH(first_name,
last_name) AGAINST ('Hau');
```

Trong ví dụ trên, ta đã chỉ định các cột mà ta muốn khớp (first_name và last_name) theo từ khóa tìm kiếm (‘Hau’). Chạy lệnh EXPLAIN cho truy vấn trên, ta có kết quả tối ưu như dưới đây:

```
mysql> EXPLAIN SELECT * FROM students WHERE
MATCH(first_name, last_name) AGAINST ('Hau');

+----+-------------+----------+------------+----------+----
----------+------------+---------+-------+------+---------
-+------------------------------+

| id | select_type | table    | partitions | type     |
possible_keys | key        | key_len | ref   | rows |
filtered | Extra                        |
```

```
+----+-------------+----------+------------+----------+----
-----------+------------+---------+-------+------+---------
-+------------------------------+

|  1 | SIMPLE      | students | NULL       | fulltext |
first_name    | first_name | 0       | const |    1 |
100.00 | Using where; Ft_hints: sorted |

+----+-------------+----------+------------+----------+----
-----------+------------+---------+-------+------+---------
-+------------------------------+
```

II. Tổng kết

Bằng việc nhìn vào id/select_type, chúng ta sẽ biết "trình tự" access vào các bảng như là vào bảng nào để lấy ra gì, sau đó dùng kết quả đó để kết hợp với bảng khác ra sao.

Bằng việc nhìn vào type/key/ref/rows, chúng ta sẽ biết với mỗi bảng sẽ có những thông tin gì được fetch ra, access vào bảng nào sẽ bị nặng, qua đó tuning index cho việc access vào bảng đó

Bằng việc nhìn vào Extra, chúng ta sẽ biết được hành vi của optimizer , biết là khi access vào bảng nào thì optimizer sẽ dự định làm gì. Qua Extra chúng ta sẽ có một cái nhìn tổng quát về query đó.