Lab Number : 03

Topic : Query Optimization

Name : Dharmathilaka A.L.V.H.

Reg. No. : E/16/086

02.

1. Use explain to analyze the outputs of following two simple queries which use only one table access.

I. SELECT * FROM departments WHERE deptname = 'Finance';

```
13    -- Question 01 part 1
14 •  EXPLAIN SELECT * FROM departments WHERE dept_name = "Finance";
15
```

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	departments	NULL	ALL	NULL	NULL	NULL	NULL	9	11.11	Using where

II. SELECT * FROM departments WHERE deptno ='d002';

```
16    -- Question 01 part II
17 •  EXPLAIN SELECT * FROM departments WHERE dept_no ="d002";
18
```

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	departments	NULL	const	PRIMARY	PRIMARY	16	const	1	100.00	NULL

What conclusions you can draw from the results?

There is a no possible key for indexing in the first query. So that, the compiler needs to go through 9 rows to displayed the results. But in the where clause of the second query, department number which is a primary key of the department table, is used. In the explanation tables, the type has been changed to "const" from "ALL". It means compiler will not go through each and every record of the table and it directly found the required one. So, it is only 1 row instead of 9 rows as in first query. As a result we can say the second query do 100% filtering.

Considering the all comparisons, it proves that using the primary key for selecting rows will increase the performance of the query execution.

2. Explain is especially important for join analysis because joins have much potential to increase the amount of server processing

Start by creating the initial tables emplist and titleperiod as follows. These derived tables need to contain only the columns involved in the query.

I. create table emplist select emp_no, first_name from employees;

II. create table titleperiod select emp_no, title, datediff(to_date, from_date) as period FROM titles;

```
19      -- Question 02
20 •    create table emplist select emp_no, first_name from employees;
21 •    create table titleperiod select emp_no, title, datediff(to_date, from_date) as period FROM titles;
22
23
```

Now write the query that gives the desired information in the required format.

SELECT first_name, period from emplist inner join titleperiod on titleperiod.emp_no=emplist.emp_no where period>4000;

Analyze the output of applying EXPLAIN to the above query explaining each value.

```
24 •    EXPLAIN SELECT first_name, period from emplist inner join titleperiod
25      on titleperiod.emp_no = emplist.emp_no
26      where period>4000;
```

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	titleperiod	NULL	ALL	NULL	NULL	NULL	NULL	441754	33.33	Using where
1	SIMPLE	emplist	NULL	ALL	NULL	NULL	NULL	NULL	299388	10.00	Using where; Using join buffer

Result Grid

Form Editor

The table titleperiod needs to map 441754 and emplist table needs to get 299388 tuples. As type mentioned as “ALL”, which means compiler needs to go through each record.

Note that the tables are in their initial unindexed state. What could be the number of row combinations that MySQL would need to check?

As MYSQL will need to go through every record, the number of row combinations will be around 299388 x 441754. This is a comparatively large number.

3. Good columns to index are those that you typically use for searching, grouping, or sorting records. The query does not have any GROUP BY or ORDER BY clauses, but it does use columns for searching: ● The query uses emplist.emp_no and titleperiod.emp_no to match records between tables.

● The query uses titleperiod.period to cut down records that do not satisfy the condition.

I. Create indexes on the columns used to join the tables. In the emplist table, emp_no can be used as a primary key because it uniquely identifies each row.

II. In the titleperiod table, emp_no must be a non-unique index because multiple employees can share the same title:

III. Analyse the outputs of EXPLAIN After creating the indexes.

Is it possible to optimize the query execution further? If so, what can be done?