CO527 – Lab 03

E/17/407

Wijesooriya H.D

(Q1)

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

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

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

What conclusions you can draw from the results?

(i)

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

- The EXPLAIN plan for the query (I) contained one row because in the query only the table 'departments' involved in the process. The query (I) does not contain any subqueries or unions, so therefore the id for the row was 1, as there is actually only 1 query.
- The select type of the query was simple as it does not contain any subqueries or unions.
- The column 'type' defines how the tables are accessed or joined. Here the access type was 'ALL' which means that the MySQL scanned the entire table.
- MySQL did not use indexes in that query therefore the column 'key' was NULL.
- During the process it examined 9 rows.

(ii)

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

- The EXPLAIN plan for the query (II) contained one row because in the query only the table 'departments' involved in the process. The query (II) does not contain any subqueries or unions, so therefore the id for the row was 1, as there is actually only one query.

- The select type of the query was simple as it does not contain any subqueries or unions.
- The column 'type' defines how the tables are accessed or joined. Here the access type was 'const' which means that the MySQL did not scan the entire table.
- Here MySQL used dept_no, the primary key of the 'departments' table as an index. During the process it examined a single row. Therefore we can say that query (II) is an optimized query.

(Q2)

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;

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

```
EXPLAIN SELECT first_name , period
FROM emplist , titleperiod
WHERE emplist.emp_no=titleperiod.emp_no AND period > 4000;
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titleperiod	NULL	ALL	NULL	NULL	NULL	NULL	442211	33.33	Using where
1	SIMPLE	emplist	NULL	ALL	NULL	NULL	NULL	NULL	299823	10.00	Using where; Using join buffer (hash join)

- The above EXPLAIN plan contains two rows because in the query, two tables 'titleperiod' and 'emplist' involved in the process. T
- he select type of the query was simple as it does not contain any subqueries or unions.
- The column 'type' defines how the tables are accessed or joined.
- Here the access type is 'ALL' which means that the MySQL scanned the two tables entirely. As you can see in the EXPLIAN the table 'titleperiod' was the first accessed table using the ALL access type. Then the table 'emplist' was accessed using the ALL accessed type.
- In the above table the column 'key' is NULL in both rows, therefore it is clear that MySQL did not use any index in the query execution.
- It scanned 442211 number of rows in the 'titleperiod' table and 299823 number of rows in the 'emplist' table. The values in the column 'filtered' contains the amount of rows unfiltered by the conditions in the WHERE clause.

- However this is not an optimized query because it scanned large number of rows in both tables. To optimize this query we create indexes on those tables.

What could be the number of row combinations that MySQL would need to check?

442211 x 299823

(Q3)

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:

```
CREATE UNIQUE INDEX emp_no_uni_index ON
Company.emplist(emp_no);

CREATE INDEX emp_no_index ON
Company.titleperiod(emp_no);
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titleperiod	NULL	ALL	emp_no_index	NULL	NULL	NULL	442211	33.33	Using where
1	SIMPLE	emplist	NULL	eq_ref	emp_no_uni_index	emp_no_uni_index	4	company.titleperiod.emp_no	1	100.00	NULL

- The above EXPLAIN plan contains two rows because in the query, two tables ‘titleperiod’ and ‘emplist’ involved in the process.
- The select type of the query was simple as it does not contain any subqueries or unions.

- The column ‘type’ defines how the tables are accessed or joined. As you can see in the EXPLIAN, the table ‘titleperiod’ was the first accessed table using the ALL access type. Then the table ‘emplist’ was accessed using the eq_ref accessed type.The database will access one row from this table for each combination of rows from the previous table.
- The column ‘key’ indicates the actual index that MySQL decided to use. In that query execution, MySQL used ‘emp_no_uni_index’ as the index which we created on the column ‘emp_no’ of the ‘emplist’ table.
- This is an optimized query as it did not scan the entire table ‘emplist’.

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

Yes it is. We can add an index on the column ‘period’ .