CO527 – Advanced Database Systems

Lab 03 - Query Optimization

Karunachandra R.H.I.O.

E/17/153

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

a) SELECT * FROM departments WHERE dept_name = 'Finance';

```
9 •   EXPLAIN SELECT * FROM departments WHERE dept_name = "Finance";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	departments	ALL	NULL	NULL	NULL	NULL	9	Using where

b) SELECT * FROM departments WHERE deptno ='d002';

```
11 •   EXPLAIN SELECT * FROM departments WHERE dept_no ="d002";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	departments	const	PRIMARY	PRIMARY	16	const	1	

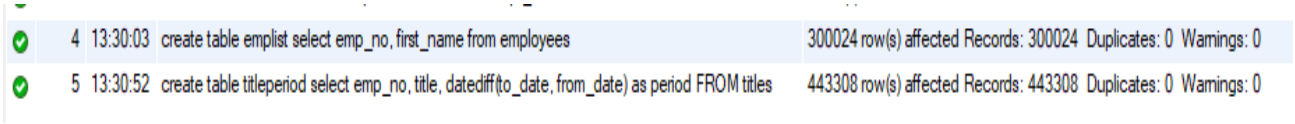
What conclusions you can draw from the results?

In the first query, there is a no possible key for indexing. So, the compiler has gone through 9 rows and displayed the results. But in the second query where, department number is used in the where clause, the select query has used the primary key as an index. There, type has been changed to "const" from "ALL" which means compiler will not go through each and every record of the table. When we consider the number of rows, it directly found the required one. So, it's only 1 row instead of 9 rows as in previous one. It's a 100% filtering. So, as a conclusion we can say that using the primary key for selecting rows will increase the query execution performance.

2. The query is to ask the question ”Which employees have worked for more than 4000 days for an assigned title” and to display for each employee his/her first name and the number of days he/she has worked on a particular title.

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;



✓	4	13:30:03	create table emplist select emp_no, first_name from employees	300024 row(s) affected Records: 300024	Duplicates: 0 Warnings: 0
✓	5	13:30:52	create table titleperiod select emp_no, title, datediff(to_date, from_date) as period FROM titles	443308 row(s) affected Records: 443308	Duplicates: 0 Warnings: 0

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

Analyze the output of applying EXPLAIN to the above query explaining each value. Note that the tables are in their initial unindexed state.

```
16 •  EXPLAIN SELECT first_name, period from emplist inner join titleperiod
17    on titleperiod.emp_no = emplist.emp_no
18    where period>4000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	emplist	ALL	NULL	NULL	NULL	NULL	299642	
1	SIMPLE	titleperiod	ALL	NULL	NULL	NULL	NULL	442734	Using where; Using join buffer (flat, BNL join)

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

Here we can see the type is “ALL“ which means MYSQL will need to go through every record. So, the number of row combinations will be around 299642 x 442734 which is a very large number.

3. 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.

```
20 •    ALTER TABLE `company`.`emplist`
21      ADD PRIMARY KEY (`emp_no`);
```

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

```
23 •    ALTER TABLE `company`.`titleperiod`
24      ADD INDEX `empNoidx` (`emp_no` ASC);
```

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

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	titleperiod	ALL	empNoidx	NULL	NULL	NULL	442734	Using where
1	SIMPLE	emplist	eq_ref	PRIMARY	PRIMARY	4	company.titleperiod.emp_no	1	

Here we can see now there are two indexes available as empnoidx and primary key. So, the number of row combinations have been decreased to 442734 and it has optimized the query in a great scale.

Type of the second operation has changed from ALL to eq_ref, that way in the emplist table MYSQL will select one row for an employee without going through all the rows. This is due to MYSQL using indexing in the that table.

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

One operation is not using its available index as a key so if we force to use it. That will optimize the execution further. Also, by adding an index for the period column, the query might be optimized more.