

Lab Number : 02

Topic : Indexing

Name : Dharmathilaka A.L.V.H.

Reg. No. : E/16/086

## Lab Tasks

01. Assuming no indexes are used, record the query execution time for retrieving all the employees by first name in ascending order.

The screenshot shows the SQL Developer interface with a query window titled 'E16086\_Answers\_lab02'. The query executed is:

```
1 • use company;
2
3 -- The Session Profiler captures the state of the
4 • set session profiling=1;
5
6 -- Question 1
7 • select * from employees order by first_name ASC;
8 • show profiles;
9
```

The 'Result Grid' displays the following data:

| emp_no | birth_date | first_name | last_name   | sex | hire_date  |
|--------|------------|------------|-------------|-----|------------|
| 11935  | 1963-03-23 | Aamer      | Jayawardene | M   | 1996-10-26 |
| 73804  | 1954-04-16 | Aamer      | Zalocco     | M   | 1993-08-13 |
| 283280 | 1962-10-21 | Aamer      | Bahl        | F   | 1985-11-22 |
| 22279  | 1959-01-30 | Aamer      | Kornyak     | M   | 1985-02-25 |
| 82975  | 1964-06-25 | Aamer      | Ghemri      | M   | 1989-08-23 |
| 293445 | 1959-10-10 | Aamer      | Jansen      | F   | 1991-08-25 |

The screenshot shows the SQL Developer interface with the 'show profiles;' query executed. The 'Result Grid' displays the following data:

| Query_ID | Duration   | Query   |
|----------|------------|---|
| 6        | 0.48893700 | select * from employees order by first_name AS... |
| 7        | 0.50931875 | select * from employees order by first_name AS... |
| 8        | 0.07524175 | drop index fname_index on employees               |
| 9        | 0.04337050 | drop index emp_ix on employees                    |
| 10       | 0.49336050 | select * from employees order by first_name AS... |

The row for Query\_ID 10 is highlighted with a red border, indicating the duration of the query execution.

Duration : 0.49336050 seconds

**02. Create an index called fname\_index on the first\_name of the employee table. Retrieve all the employees by first name and record the query execution time. Observe the performance improvement gained when accessing with index.**

```

10 -- Question 2
11 • create index fname_index on employees(first_name);
12 • select * from employees order by first_name ASC;
13 • show profiles;
14

```

| Query_ID | Duration   | Query   |
|----------|------------|---|
| 10       | 0.49336050 | select * from employees order by first_name AS... |
| 11       | 2.19497475 | create index fname_index on employees(first_...   |
| 12       | 0.52299575 | select * from employees order by first_name AS... |
| 13       | 0.51093550 | select * from employees order by first_name AS... |
| 14       | 0.46928950 | select * from employees order by first_name AS... |

**Duration :** 0.46928950 seconds

**Observation:** Query execution time reduced. Since this table is large, indexing will improve the performance.

**03. Which indexing technique has been used when creating the above index? Hint: You can use SHOW INDEX FROM [mytable]; to see details of your indexes.**

```

15 -- Question 3
16 • show index from employees;
17 • show profiles;
18
19
20

```

| Table     | Non_unique | Key_name    | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
|-----------|------------|-------------|--------------|-------------|-----------|-------------|----------|--------|------|------------|
| employees | 0          | PRIMARY     | 1            | emp_no      | A         | 299478      | NULL     | NULL   |      | BTREE      |
| employees | 1          | fname_index | 1            | first_name  | A         | 1262        | NULL     | NULL   | YES  | BTREE      |

It is non unique index type BTREE.

**04. Create a unique index on emp\_no, first\_name and last\_name of employees table. Retrieve all the employees by emp\_no, first\_name and last\_name. : 28th November 2022 before 11:50 PM Observe if there is any performance improvement with respect to question1. If not, explain any possible reason.**

**Duration :** 0.46114550seconds

The performance is improved since in question 1,2 it took longer duration than in question 4.

```

19  -- Question 4
20  • create unique index emp_ix on employees(emp_no, first_name, last_name);
21  • select * from employees order by first_name ASC;
22  • show profiles;
23
24

```

| Query_ID | Duration   | Query   |
|----------|------------|---|
| 13       | 0.51093550 | select * from employees order by first_name AS... |
| 14       | 0.46928950 | select * from employees order by first_name AS... |
| 15       | 0.00154175 | show index from employees                         |
| 16       | 1.73020850 | create unique index emp_ix on employees(emp...    |
| 17       | 0.46114550 | select * from employees order by first_name AS... |

Result 27 x Read Only

05. Take the following 3 queries.

A. select distinct emp\_no from dept\_manager where from\_date>= '1985-01-01' and dept\_no>= 'd005';

B. select distinct emp\_no from dept\_manager where from\_date>= '1996-01-03' and dept\_no>= 'd005';

C. select distinct emp\_no from dept\_manager where from\_date>= '1985-01-01' and dept\_no<= 'd009';

I. Choose one single simple index(i.e index on one attribute) that is most likely to speed up all 3 queries giving reasons for your selection.

All three queries use project emp\_no attribute as it is primary key. So indexing is not done manually since we have only 9 departments. But from\_date is a non-unique attribute which is used in every query. So decided to get the from\_date attribute to have good indexing.

II. For each of the 3 queries, check if MySQL storage engine used that index. If not, give a short explanation why not. You can prefix your select queries with EXPLAIN EXTENDED or with EXPLAIN to display a query execution plan.

query A

```

28  -- Query A
29  • EXPLAIN select distinct emp_no from dept_manager where from_date>='1985-01-01' and dept_no >= 'd005';
30

```

| id | select_type | table        | partitions | type  | possible_keys        | key     | key_len | ref  | row | filtered | Extra                      |
|----|-------------|--------------|------------|-------|----------------------|---------|---------|------|-----|----------|----------------------------|
| 1  | SIMPLE      | dept_manager | HULL       | range | PRIMARY,emp_no,frmDt | PRIMARY | 16      | HULL | 14  | 83.33    | Using where; Using tempora |

According to the result, the query A used the frmDt index. It used it as the key here.

query B

```

31  -- Query B
32  • EXPLAIN select distinct emp_no from dept_manager where from_date>= '1996-01-03' and dept_no >= 'd005';
33

```

| id | select_type | table        | partitions | type  | possible_keys        | key   | key_len | ref  | row | filtered | Extra                       |
|----|-------------|--------------|------------|-------|----------------------|-------|---------|------|-----|----------|-----------------------------|
| 1  | SIMPLE      | dept_manager | NULL       | range | PRIMARY,emp_no,frmDt | frmDt | 20      | NULL | 2   | 58.33    | Using where; Using index; U |

According to the result, the query B used the frmDt index. It used it as the key here.

query C

```

34  -- Query C
35  • EXPLAIN select distinct emp_no from dept_manager where from_date>='1985-01-01' and dept_no <= 'd009';
36
37

```

| id | select_type | table        | partitions | type  | possible_keys        | key     | key_len | ref  | row | filtered | Extra                      |
|----|-------------|--------------|------------|-------|----------------------|---------|---------|------|-----|----------|----------------------------|
| 1  | SIMPLE      | dept_manager | NULL       | range | PRIMARY,emp_no,frmDt | PRIMARY | 16      | NULL | 24  | 100.00   | Using where; Using tempora |

According to the result, the query C used the frmDt index. It used it as the key here.

### Conclusion:

As all three queries used the index frmDTt, it can be considered as a good choice which can increase the performance.

**06. Consider the queries you wrote for questions 2 - 10 in Lab 01 assignment. Give with short explanations, which attributes on which relations should be used for creating indexes that could speed up your queries.**

#### Question 2

index can be used for last\_name as we consider last names and their counts from the table 'employees'.

#### Question 3

As we select engineers with the use of title attribute, title attribute can used for index. If we consider it to the current date, indexing can create for index for title with to\_date in titles table.

#### Question 4

Multiple indexes can be created.

- An index on title and to\_date fields of 'titles' table.
- An index on sex field of 'employees' table.

Since we consider titles department managers, senior engineers and female employees.

### Question 5

We can create multiple indexes on different tables.

- An index on salary and to\_date fields of 'salary' table
- An index on to\_date field of 'dept\_emp' table
- An index on title field of 'titles' table

As the main target of that query was to find the departments and titles of employees who have a salary greater than 115000 and the above index will speed up the query.

### Question 6

We can create multiple indexes on different tables.

- An index on hire\_date and birth\_date fields of 'employees' table
- An index on to\_date field of 'dept\_emp' table

As in the where clause of that query I have considered about the hire\_date, birth\_date and to\_date of an employee.

### Question 7

As we categorize based on department name, attribute dept\_name can be used for indexing.

### Question 8

We can create multiple indexes on different tables.

- An index on salary field of the 'salaries' table
- An index on dept\_name field of the 'departments' table

As we have to find the names of all employees in the database who earn more than every employee in the Finance department. Creating these indexes will speed up the query.

### Question 9

We can create an index on salary and to\_date fields of 'salaries' table. As in the where clause we have to compare the salary and to\_date of an employee with another employee's those attributes.

### Question 10

We can create multiple indexes in order to speed up the query execution time. It is as follows,

- An index on salary field of the table 'salaries'
- An index on title field of the table 'titles'

As in that query I have calculated the difference between the average salary of a Senior Engineer and the average salary of all employees. Having above indexes will speed up the process.

**07. Assume that most of the queries on a relation are insert/update/delete. What will happen to the query execution time if that relation has an index created.**

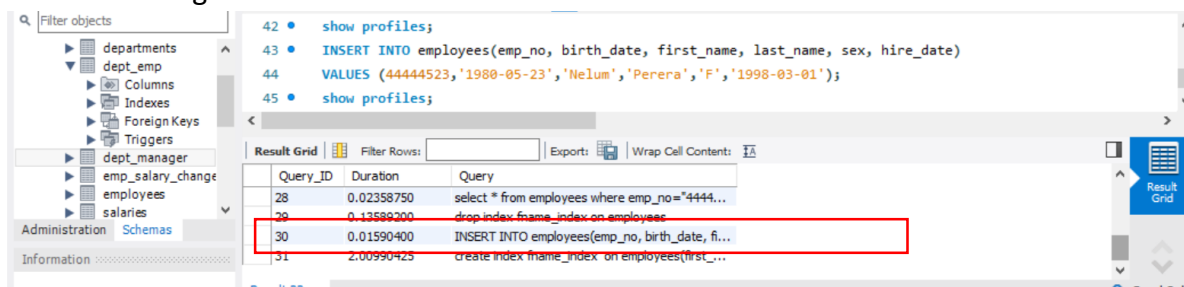
### Insert:

Example scenario: A tuple is added to the employees table before and after adding indexing to first name:

Used query:

```
INSERT INTO employees(emp_no, birth_date, first_name, last_name, sex, hire_date)  
VALUES (44444523, '1980-05-23', 'Nelum', 'Perera', 'F', '1998-03-01');
```

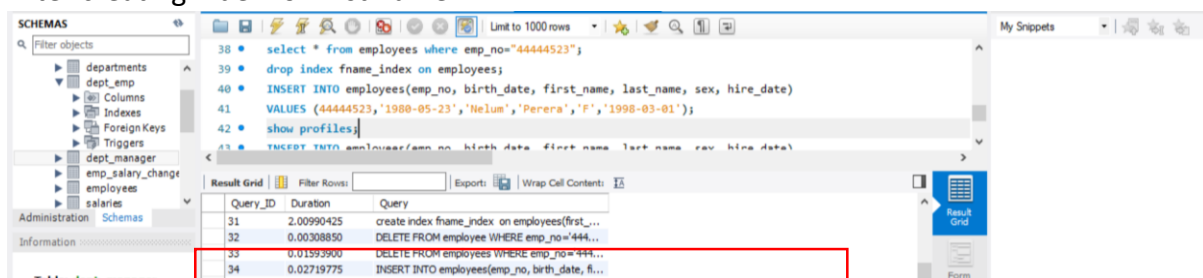
Before creating index for first name:



| Query_ID | Duration   | Query   |
|----------|------------|---|
| 28       | 0.02358750 | select * from employees where emp_no="4444...   |
| 29       | 0.13589200 | drop index fname_index on employees;            |
| 30       | 0.01590400 | INSERT INTO employees(emp_no, birth_date, fi... |
| 31       | 2.00990425 | create index fname_index on employees(first_... |

Id=30 has the execution time duration. It is 0.01590400 seconds.

After creating index for first name:



| Query_ID | Duration   | Query   |
|----------|------------|---|
| 38       | 0.02358750 | select * from employees where emp_no="4444...   |
| 39       | 0.13589200 | drop index fname_index on employees;            |
| 40       | 0.01590400 | INSERT INTO employees(emp_no, birth_date, fi... |
| 41       | 0.01590400 | DELETE FROM employees WHERE emp_no="444...      |

For same query, Id=34 has execution time duration. It is 0.02719775 seconds.

According to that, the query execution time was increased after creating index.

### Update:

Example scenario: A tuple first name is updated in the employees table before and after adding indexing to first name:

Used query:

```
UPDATE employees SET first_name='Nelumi' where emp_no = 44444523;
```

Before creating index for first name:

| Query_ID | Duration   | Query   |
|----------|------------|---|
| 34       | 0.02719775 | INSERT INTO employees(emp_no, birth_date, fi... |
| 35       | 0.00170725 | create index fname_index on employees(first...  |
| 36       | 0.01007100 | DELETE FROM employees WHERE emp_no='444...      |
| 37       | 0.00943275 | INSERT INTO employees(emp_no, birth_date, fi... |
| 38       | 0.01517675 | UPDATE employees SET first_name='Nelumi' wh...  |

The process Id=38 has the execution time. It is 0.01517675 seconds.

After creating index for first name:

| Query_ID | Duration   | Query   |
|----------|------------|---|
| 39       | 0.01176400 | DELETE FROM employees WHERE emp_no='444...      |
| 40       | 0.00964250 | INSERT INTO employees(emp_no, birth_date, fi... |
| 41       | 0.00249200 | create index fname_index on employees(first...  |
| 42       | 0.01086950 | UPDATE employees SET first_name='Nelumi' wh...  |
| 43       | 0.00048200 | UPDATE employees SET first_name='Nelumi' wh...  |

The process Id=42 has the execution time 0.01086950 seconds

According to the result query execution time is decreased when creating a index related to it.

**Delete:**

**DELETE FROM employees WHERE emp\_no='44444523' and first\_name="Nelum";**

| Query_ID | Duration   | Query   |
|----------|------------|---|
| 53       | 0.00945200 | INSERT INTO employees(emp_no, birth_date, fi... |
| 54       | 0.01030075 | DELETE FROM employees WHERE emp_no='444...      |
| 55       | 2.27217300 | create index fname_index on employees(first...  |
| 56       | 0.01223500 | INSERT INTO employees(emp_no, birth_date, fi... |
| 57       | 0.01209825 | DELETE FROM employees WHERE emp_no='444...      |

The query id =54 execution time duration is 0.00945200seconds for the delete operation which has done before creating index for first name. After creating an index for first name, the execution time duration is 0.01209825seconds.

According to the results, for delete operation the execution time is increased after creating index for first name. This is because after deleting, if indexing is there, those entries should be deleted from indexing too.

## SQL Queries:

```
use company;
```

```
-- The Session Profiler captures the state of the
```

```
set session profiling=1;
```

```
-- Question 1 -----
```

```
select * from employees order by first_name ASC;
```

```
show profiles;
```

```
-- Question 2 -----
```

```
create index fname_index on employees(first_name);
```

```
select * from employees order by first_name ASC;
```

```
show profiles;
```

```
-- Question 3 -----
```

```
show index from employees;
```

```
show profiles;
```

```
-- Question 4 -----
```

```
create unique index emp_ix on employees(emp_no, first_name, last_name);
```

```
select * from employees order by first_name ASC;
```

```
show profiles;
```

```
-- Question 5 Part I -----
```

```
create index frmDt on dept_manager(from_date);
```

```
-- Question 5 Part II -----
```

```
-- Query A
```

```
EXPLAIN select distinct emp_no from dept_manager where from_date>='1985-01-01' and dept_no  
>= 'd005';
```



-- Query B

```
EXPLAIN select distinct emp_no from dept_manager where from_date >= '1996-01-03' and dept_no >= 'd005';
```

-- Query C

```
EXPLAIN select distinct emp_no from dept_manager where from_date >= '1985-01-01' and dept_no <= 'd009';
```

-- Question 7 -----

-- check for unique index

```
select * from employees where emp_no="44444523";
```

-- remove index for first name

```
drop index fname_index on employees;
```

-- insert values for employee table

```
INSERT INTO employees(emp_no, birth_date, first_name, last_name, sex, hire_date)
VALUES (44444523, '1980-05-23', 'Nelum', 'Perera', 'F', '1998-03-01');
```

-- update values in employees table

```
UPDATE employees SET first_name='Nelumi' where emp_no = 44444523;
```

-- delete values in employees table

```
DELETE FROM employees WHERE emp_no='44444523' and first_name="Nelum";
```

```
show profiles;
```