Lab 02: Indexing

CO527: Advanced Database Systems

June 30, 2016

1 Introduction

In this lab session, we will be focusing on an important technique called *indexing* which is used for improving performance of query processing. At the end of the lab session, you should be familiarised with indexing and query efficiency with the understanding of right index selection.

An index on an attribute A of relation R allows the database to find quickly all tuples in R with a given value of attribute A. This index is useful if a value of A is specified by your query in the where clause. Index selection is critical to performance in databases. You will use one or more indexes to build.

1.1 SQL *CREATE INDEX* Syntax

Creating an index involves the CREATE INDEX statement, which allows you to name the index to specify the table and which column or columns to index.

CREATE INDEX [name of index] ON [table name] ([attribute-list]);

Example:

CREATE INDEX income_ix ON Company.salaries(salary);

would create an index on the salary field of salaries table in Company database. And the following query uses this income_ix index when returning the related result.

```
SELECT emp_no
FROM Salaries
WHERE salary >15000;
```

1.2 SQL *CREATE UNIQUE INDEX* Syntax

Indexes can be created such that they prevent duplicate entries in the indexing field/fields.

CREATE UNIQUE INDEX name_ix ON Company.employees (first_name, last_name);

1.3 SQL *DROP INDEX*

You can drop a created index with *DROP INDEX [index_name];*

2 Exercises

Refer to the same Company database you created for the previous lab and answer the following questions.

1. Assuming no indexes are used, record the query execution time for retrieving all the employees by first name in ascending order.
2. 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.
3. 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.
4. Create 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*. Observe if there is any performance improvement with respect to question 1. If not, explain any possible reason.
5. 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.

   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.
   (Note that in MySQL InnoDB engine uses a clustered index usually on the primary key of the table, by default. We only care about the index you create.)
6. 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.
7. 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?

3 Submission

Submissions have to be done within one week from the lab class. Submit a single E/1X/XXX_lab02.txt file including all your .sql scripts and explanations. You can discuss among your classmates, but the submitted assignments should be your own work. Please note that committing **plagiarism** or (assisting others to commit plagiarism) will result in zero marks.