Lab 4

Out: 11/24/15, Due 12/15/15

In this lab, you will play with PostgreSQL as a black box. You will not be required to write code or modify anything in the PostgreSQL layers. Instead, you will do some experiments with large dataset and study the performance of different indexing structures, and different query variants.

The lab consists of two parts. In the first part, you are required to come up with some queries showing the effect of using various indexing structures. In the second part, you are required to provide two variants of each input query, and explain the difference in performance between them. In both parts, you should analyze the results you got in a report.

Lab Setup

Recall the following database schema, it is a reduced version from the schema used in lab 2 (keys are underlined):

- student(sid, sname, sex, age, year, gpa)
- dept(dname, numphds)
- major(dname, sid)

We have included several text files with data that you need to insert into these tables. Don't worry, you will be given PostgreSQL scripts to generate the tables, and load the data in them. Once you have done so, you can proceed to do the lab requirements.

Part 1: The effect of using indexes

1. Choose three different attributes from the student relation, such that each attribute has no index (and of course not a primary key). Then, provide a SELECT query on each attribute with any filtering criteria you determine. Report both the SELECT query and the execution time for each attribute, and analyze the results you got.
2. Create a B-tree index for each of the three attributes. Then, run the same queries in step 1 and report the time and your analysis again. The syntax to create a B-tree index on an attribute is "CREATE INDEX *index_name* ON *table_name* (*attribute_name*)".
3. Create a hash index for each of the three attributes. Then, run the same queries in step 1 and report the time and your analysis again. You should remove the B-tree index before you create the hash index. The syntax to create a hash index on an attribute is "CREATE INDEX *index_name* ON *table_name* USING hash (*attribute_name*)".
4. Focus on one indexed attribute only, and provide three SELECT queries with different filtering criteria on this attribute (e.g. age > 1, age > 10, and age > 50). Is there a difference between the three execution times? Why? You should do this step twice; Once with hash index and once with B-tree index.
5. Using the same indexed attribute from step 4, provide an equality query (e.g. age = 20), and a range query (e.g. 10 < age < 20). Try each query with both B-tree and hash indexes. Then, report the execution time and your analysis.

6. Choose two indexed attributes, and do a composite search with any filtering criteria that combine these attributes together (e.g. age > 15 and sex = ‘male’). Try the different indexing combinations; both attributes are non-indexed, one of them is only indexed with either B-tree or hash index, both of them indexed with similar indexes, or both of them indexed with different indexes. Report the execution time and your analysis for each indexing combination.

Part 2: Different variants of SQL queries

1. Write a query to select all students with age < 30, and record the execution time. Then, use the DISTINCT keyword. Is there a difference in the execution time? Why?
2. Provide two queries that can show the difference in performance between having a condition in WHERE clause, and a condition in HAVING clause. Both queries should have the same output.
3. Write a query to get the names of departments that have one or more majors who are under 30 years old. Provide two variants of this query; the first variant uses JOIN and WHERE, and the second variant uses IN and a nested query. Do they have the same execution time? If not, which variant is efficient? Why?

Submit a single zip file, via Moodle, containing your SQL file for the above queries, as well as a short report describing your queries and explaining the results you got. A query without analysis will get **ZERO** point. Please, **DOUBLE CHECK** your SQL syntax, and make sure that all queries are correctly running on PostgreSQL. As before, all files must be in text-only format. This lab should be done in groups of two (It is preferable to keep your group from Lab3).