# **CSCI X370: Database Management**

**Spring 2017**

## **Project 4: Performance Tuning of SQL Queries**

## Due: April 12 (11:59 pm)

In this project, we want to test the two DBMS to see which one is faster using the queries given below.

Consider the below schema and tables:

1. Student [ id, name, address, status ]
2. Professor [ id, name, deptId ]
3. Course [ crsCode, deptId, crsName, descry ]
4. Transcript [ studId, crsCode, semester, grade ]
5. Teaching [ profId, crsCode, semester ]

**Tuple counts** : Student : 10000, Professor: 1000, Course: 2000, Teaching: 5000, Transcript: 5000

You can assume any reasonable datatypes or you can refer to the tables given in Chapter 3 of textbook.

Write queries on the given schema as below:

1. List the name of the student with id equal to v1 (id).
2. List the names of students with id in the range of v2 (id) to v3 (inclusive).
3. List the names of students who have taken course v4 (crsCode).
4. List the names of students who have taken a course taught by professor v5 (name).
5. List the names of students who have taken a course from department v6 (deptId), but not v7.
6. List the names of students who have taken all courses offered by department v8 (deptId).

In these queries, v1, v2, ... stands for value1, value2, ...etc.

To be able to test the two database systems, which are MySQL and PostgreSQL, you need to analyze query plans generated by the DBMS and tune these queries to be able to run them faster. For tuning these queries, links to tuning guides for each DBMS are given below:

MySQL: http://dev.mysql.com/doc/refman/5.6/en/optimization.html

PostgreSQL: http://www.postgresql.org/docs/9.1/interactive/internals.html

You will need to run the queries, look at the query plans, and use some of hints given in the tuning guides of each DBMS above (removal of redundant parenthesis in the query, rewriting queries etc.). Also, you can add indexes to try to speed it up. Then you need to write *a report about each query,* which is short that indicates what you did to speed it up. For doing this, you need to obtain initial timing results and what the timing result is after you tuned it.

Furthermore, you need to create a database, which you can run those queries on. You'll create a sample database used in the textbook, and you can find the details of this database, which is used throughout the book (database on students, courses etc.). However, if you just create a database with small number of tuples it is going to take 0 times, and you will not be able to make any comparison. Hence you need to insert enough tuples as per the given tuple counts to get the proper timing result and can make a comparison in between. To do so, you need to use Tuple Generator that you have used earlier. You will retarget the Tuple Generator to be able to quickly populate your database.

You need to turn in before and after .sql files and query plans for six queries (given in English, see above). Do this for both the MySQL (MySQL workbench) and PostgreSQL DBMSs. Also, please analyze the queries using explain plan and write your observations and performance cost and time taken to execute before and after the tuning.

**The following documents should be submitted**:

1. Schema file (TeamName\_DatabaseName\_MySQL.sql, TeamName\_DatabaseName\_Postgres.sql)
2. Queries before tuning (TeamName\_BefTuning\_MySQL.sql, TeamName\_BefTuning\_Postgres.sqL)
3. Queries after tuning (TeamName\_AftTuningMySQL.sql, TeamName\_AftTuningPostgres.sql)
4. Document with explain plans for each of the queries and your observations about how the performance has improved after tuning the queries and the screen shots of the explain plans for each of the queries.(Two separate documents for each database)
5. Readme file
6. Team Leader report

**Note : The .sql files should include the create database and use database names statements and comments added infront of them.**

Please add comments in the database file before each query related to the question number and query.

e.g -- create database TeamName

-- use database TeamName

test