MCDA 5540 – Assignment 3 and 4

==

Date assigned: Oct 23 Time due: Nov 12 at 11:59pm

==

This is a TEAM assignment, where teams are expected to be the same as the PROJECT teams.

PART 1 – **Assignment-3**

This part is related to the examples done in the Tutorial and refers to the files in the subfolder assignment4 in the resources part of the course website. Regarding the scripts and data files on the Brightspace, note the following.

1. The MySQL table "person" is created in the script "make_person_table.sql" using 20,000 person lines from the file "persons.tsv".

2. The python script "make_person_history.py" can be used to generate any number of events about 20,000 persons (technically about persons with id's between 1 and 20,000). If N is the number of events given as input, the script generates N event lines and saves them in the file "phistory.tsv". Note that the data files "cities.txt" and "countries.txt" are used by the script to generate city and country data.

3. The MySQL table "history" is created in the script "make_hist_table.sql" using the event lines from the file "phistory.tsv".

4. In the Tutorial, the history file "phistory.tsv" was created with N = 1,000,000 events and then the effect of an index on the attribute history(city) was tested. In particular, the **test** was to make a search in the join of the tables person and history on person id using history.city as a search attribute. The search was first done without an index on the search attribute history(city), and then after creating an index on history(city).

Your first task is to repeat the above test for several increasing values of N and make a table in which you record, for each N, the milliseconds for the search with and without an index. Do this for at least 10 values of N, where the increase should be as large as possible.

Your second task is to explore the effect of an index when inserting new data in a table. Make a second history file "phistory2.tsv" of 1,000 event lines. Make a transaction that inserts the 1,000 event lines when the index exists, and a transaction when it does not exist, and record the times required for these insertions. Do that for the values of N you have chosen above. In all cases rollback the transactions so that, in the end, the insertions have no effect.

Of course, it makes sense to **do** the two **tasks above together**. As the amount of data and computation time might be large, you are **not** supposed to do the above tasks on the server `dev.cs.smu.ca`. Do them **on your laptops**.

YOUR REPORT: write a report that contains (i) your measurements in table form (ii) a few paragraphs explaining your findings about the performance of the index (iii) snapshots of the execution of scripts/mySQL-statements used in the above tasks.

PART 2 – Assignment-4

The main goal of this part is to process a set of data files such that they are inserted appropriately into a Mongodb collection called **newsitems** that would allow one to perform text search for news relevant to a given topic. Each data file contains a news article. In particular, for each data file, the first line is the headline, the second line is the list of reporters and/or the reporting agency, and the rest of the lines constitute the article content. Once all data get inserted into Mongodb, a user should be able to query for relevance as follows: I want to find the top 2 articles relevant to a given topic. Relevance should be based on the scores that Mongodb assigns to documents with respect to the given topic. I should be able to enter the query:

```
>
> db.newsitems.find(
    {$text: {$search: "Trump"}},
    {_id:0, title:1, score: {$meta: "textScore"}}).sort({score:
{$meta: "textScore"}}).limit(2)
>
```

WRITE a bash script that reads all data files in the directory **data_files** and creates the required Mongodb collection **newsitems** together with its text index. Your script will be tested on the server `dev.cs.smu.ca`. See Brightspace for available data and scripts.

IMPORTANT COMMENTS

1. Late submissions will not be accepted.
2. Submission procedure: Only **one** submission per team. Produce a zip file, called Axx.zip, where Axx is the A-number of one team member. The file should contain:
   a. Your report for the 1st part in **pdf** format with the **names** and **A**-numbers of the team members at the top of the report.
   b. Your bash script for the 2nd part.

   Submit Axx.zip on Brightspace.